M Series

Programmable Controller

M-PLC Instruction User Manual

NEXT Level SOLUTION

The contents of the manual will be revised as the version changes, and this version may not be the final version. Please go to www.fatek.com technical support area to download the latest version of the manual.

FATEK AUTOMATION CORP.

INDEX

7 Advanced Function Instructions

8 Step Instruction Description

9 Real Time Clock (RTC)

Amendment Record

Precautions on Using the Product

Compliance with the application-related conditions
The user shall evaluate the suitability of FATEK product and shall install the product in the well-designed equipment or system. The user needs to check if the system, machinery or device currently used is compatible with the FATEK product. If the user fails to confirm the compatibility or the suitability, then FATEK shall not be liable for the suitability of the product. When required by the customer, FATEK shall provide correlated third party certification to define the value rating and the application restrictions that will be applicable for the product. However, the aforesaid certification message shall not be considered as sufficient to determine the suitability of the FATEK product, the final product, the machine, the system and other applications or relevant combinations. Described below are certain applications that should be cautiously treated by the user. In spite of this, the content described below shall neither be considered as having included all of the intended product purposes nor suggesting that all of the following purposes shall be entirely suitable for the product. For example, outdoors use, use in an area subjected to potential chemical contamination or electrical interference or used under conditions or functions not mentioned in this Manual or used with the system, machine and equipment that may create risks to life or properties. Before working with the product, the user will be required to check if the entire system is marked with a hazard sign and shall select the design that can ensure the safety such as the backup design, etc. Otherwise, the user shall not be allowed to use the product in the application that will present personnel and the property safety concerns. In no event shall FATEK be liable for the specifications, statutory regulations or restrictions that will be used by the customer in the product combination or the product operations. When using the product, FATEK shall not be liable for the programs edited by the user or the resulting consequences.

Disclaimers

Dimensions and weight
The dimensions and the weight specified in the manual are nominal values only. Even if provided with the tolerance, they cannot be used in the manufacturing purposes.

Performance data
The data specified in this Manual mean that the performance data obtained under FATEKf provided with the tolerance, they cannot be used in the manufacturing purposes. afety such as the backup design, etc. Otherwise, the user shall not be allowed to use the actual performance shall be defined according to the content of the guarantee and the limit of responsibilities established by FATEK.

Errors and negligence
The content of this Manual is provided through careful checking process and is considered as correct. However, FATEK shall not be liable for the errors or the negligence that may be found in the text, printing content and proofreading.

Change of specifications
The product specifications and accessories may be subject to change along with the technical improvement or other reasons. In the event that the published specifications or performance need to be changed or where significant structural change is required, FATEK will change the model number of the product accordingly. If certain specifications of the product have changed, then FATEK will not give the notice under the following situation: when it is required to use a special model number or create particular specifications in order to support the customer’s application according to the instructions given by the customer. To confirm actual specifications of the product to be purchased, please contact the local FATEK distributor.

1

PLC Ladder Diagram and the Coding Rules of Mnemonic

In this chapter, we would like to introduce you the basic principles of ladder diagram.

1-1 The Operation Principle of Ladder Diagram

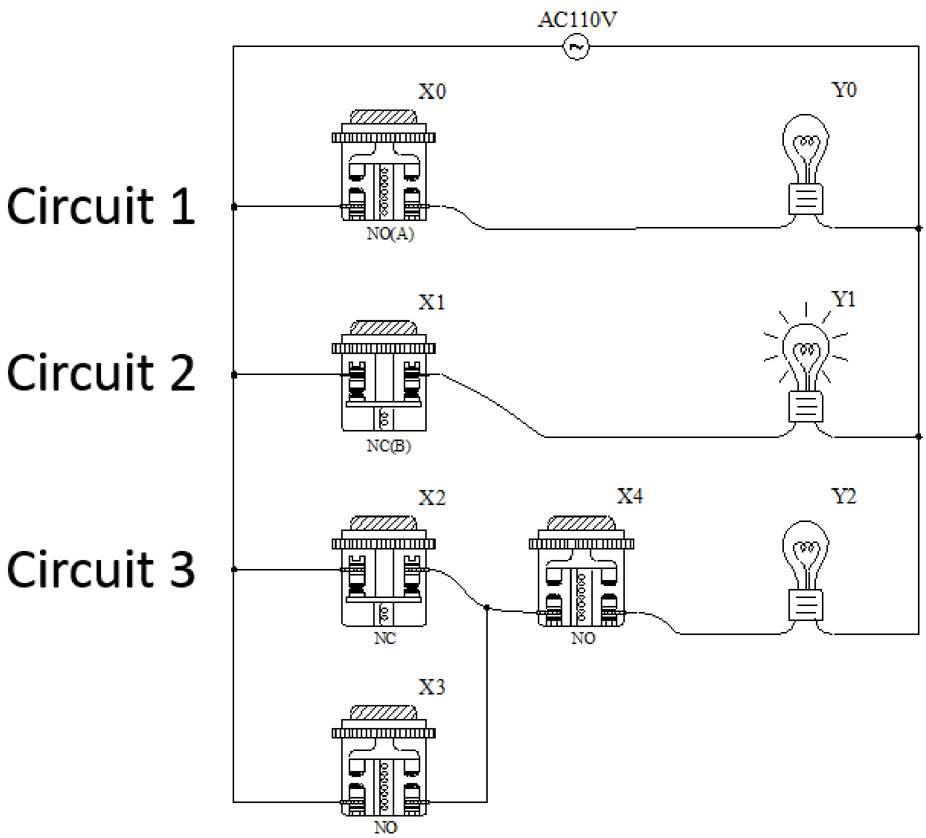
Ladder Diagram is a type of graphic language for automatic control systems it had been used for a long period since World War II. Until today, it is the oldest and most popular language for automatic control systems. Originally there are only few basic elements available such as A-contact (Normally ON), B contact (Normally OFF), output Coil, Timers and Counters.

Not until the appearance of microprocessor-based PLC, more elements for Ladder Diagram, such as differential contact, retentive coil (refer Table 2) and other instructions that a conventional system cannot provide, became available.

The basic operation principle for both conventional and PLC Ladder Diagram is the same. The main difference between the two systems is that the appearance of the symbols for conventional Ladder Diagram are closer to the real devices, while for PLC system, symbols are simplified for computer display. There are two types of logic system available for Ladder Diagram logic, namely combination logic and sequential logic. Detailed explanations for these two logics are discussed below:

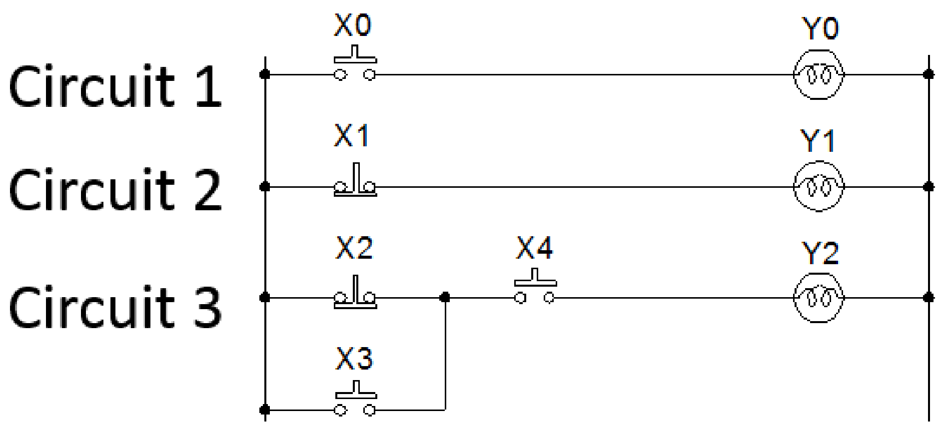
1-1-1 Combination Logic

Combination logic of the Ladder Diagram is a circuit that combines one or more input elements in series or parallel and then send the results to the output elements, such as Coils, Timers/Counters, and other application instructions.

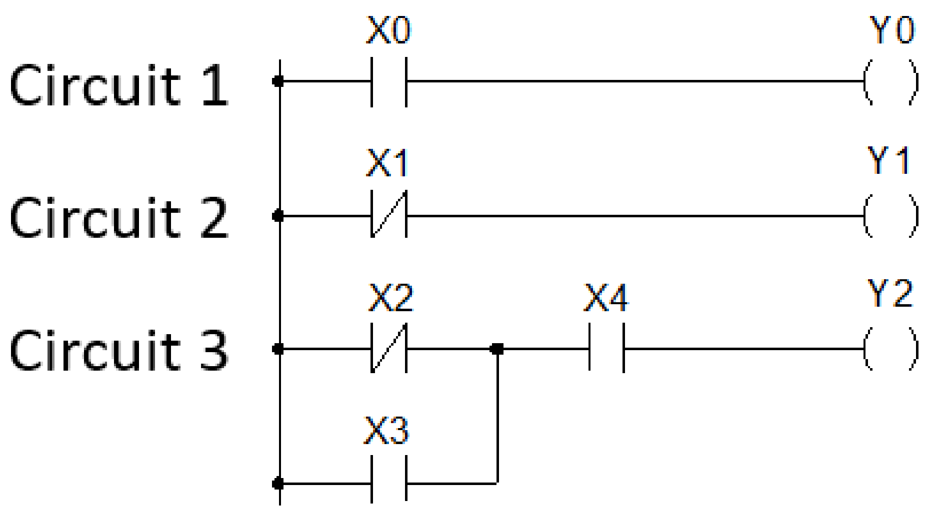


Combination logic_Actual wiring diagram

The example illustrated the combination logic using the actual wiring diagram, conventional Ladder Diagram, and PLC Ladder Diagram. Circuit 1 uses a NO (Normally Open) switch that is also called "A" switch or contact. Under normal condition (switch is not pressed), the switch contact is at OFF state and the light is off. If the switch is pressed, the contact status turns ON and the light is on.
In contrast, circuit 2 uses a NC (Normally Close) switch that is also called "B" switch or contact. Under normal condition, the switch contact is at ON state and the light is on. If the switch is pressed, the contact status turns OFF and the light also turns off.
Circuit 3 contains more than one input element. Output Y2 light will turn on under the condition when X2 is closed or X3 switches to ON, and X4 must switch ON too.



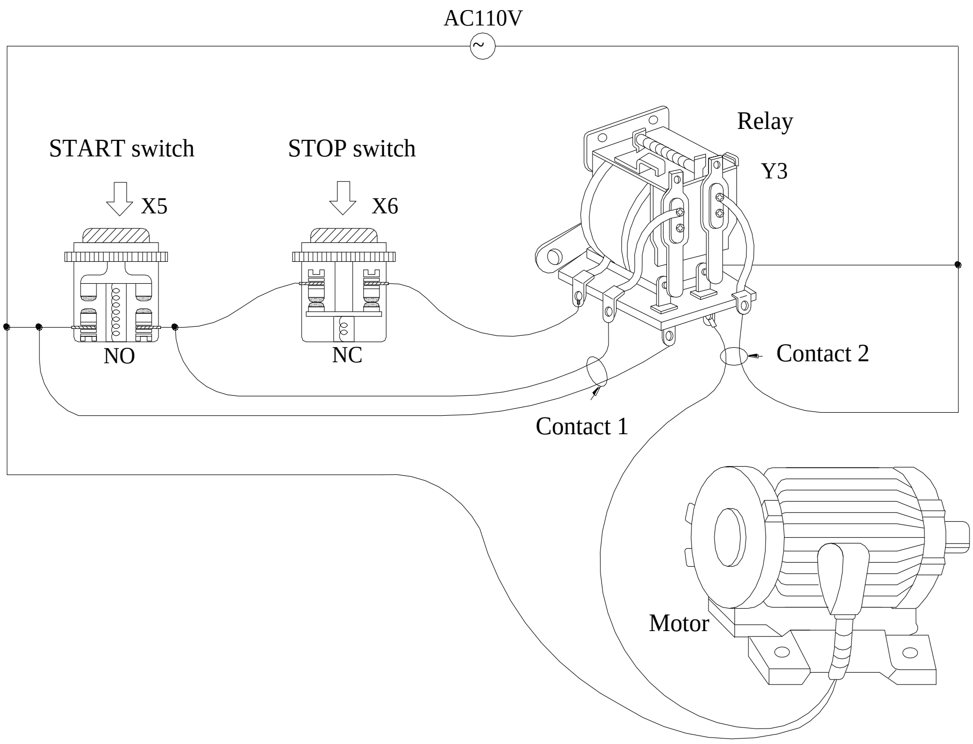
Combination logic_Conventional Ladder Diagram



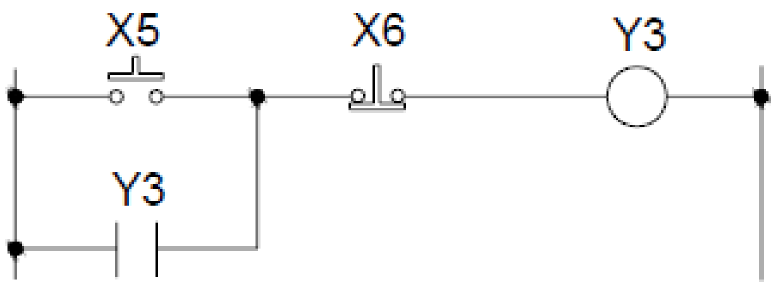
Combination logic_PLC Ladder Diagram

1-1-2 Sequential Logic

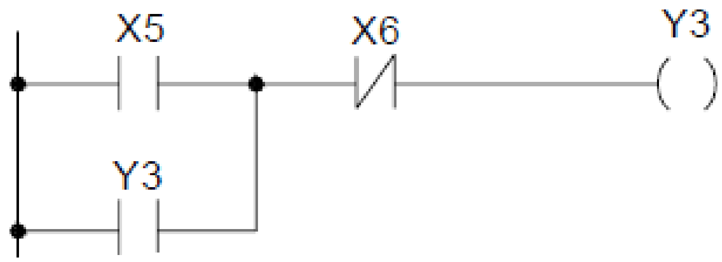
The sequential logic is a circuit with feedback control; that is, the output of the circuit will be feedback as an input to the same circuit. The output result remains in the same state even if the input condition changes to the original position. This process can be best explained by the ON/OFF circuit of a latched motor driver as shown in below.



Sequential logic_Actual wiring diagram



Sequential logic_ Conventional Ladder Diagram



Sequential logic_ PLC Ladder Diagram

When we first connect this circuit to the power source, X6 switch is ON but X5 switch is OFF, therefore the relay Y3 is OFF. The relay output contacts 1and 2 are OFF because they belong to A contact (ON when relay is ON). Motor does not run. If we press down the switch X5, the relay turns ON as well as contacts 1and 2 are ON and the Motor starts. Once the relay turns ON, if we release the X5 switch (turns OFF), relay can retain its state with the feedback support from contact 1 and it is called Latch Circuit. The following table shows the switching process of the example we have discussed above：

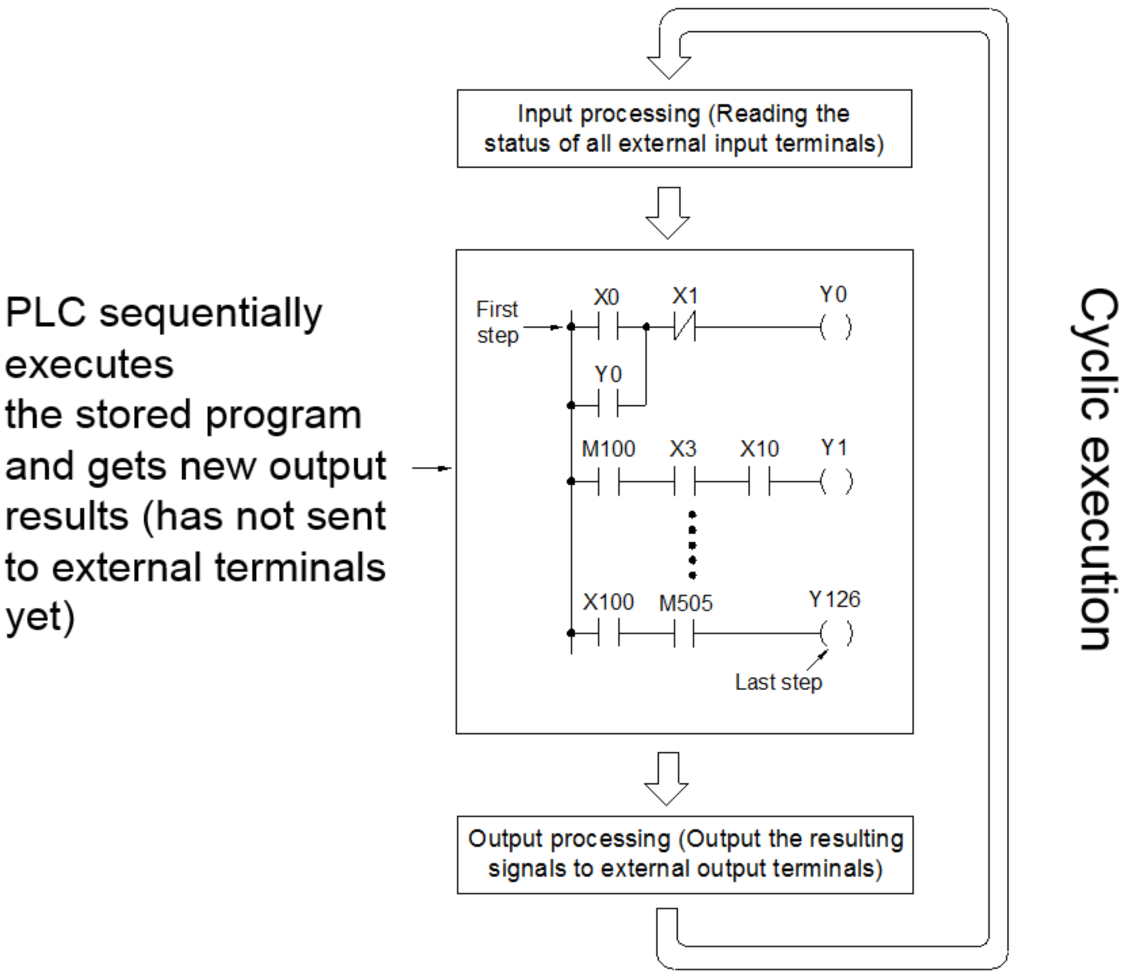
	X5 Switch (NO)	X6 Switch (NC)	Motor (Relay) Status
① ↓	Released	Released	OFF
② ↓	Pressed	Released	ON
③ ↓	Released	Released	ON
④ ↓	Released	Pressed	OFF
⑤	Released	Released	OFF

Sequential logic_Action

From the above table we can see that under different stages of sequence, the results can be different even the input statuses are the same. For example, X5 and X6 switches are both released, but the Motor is ON (running) at status ③ and is OFF (stopped) at status ①. This sequential control with the feedback of the output to the input is a unique characteristic of Ladder Diagram circuit. Sometimes we call the Ladder Diagram a "Sequential Control Circuit" and the PLC a "Sequencer”. In this section, we only use the A/B contacts and output coils as the example. For more details on sequential instructions please refer to Chapter 5 "Introduction of Sequential Instructions”.

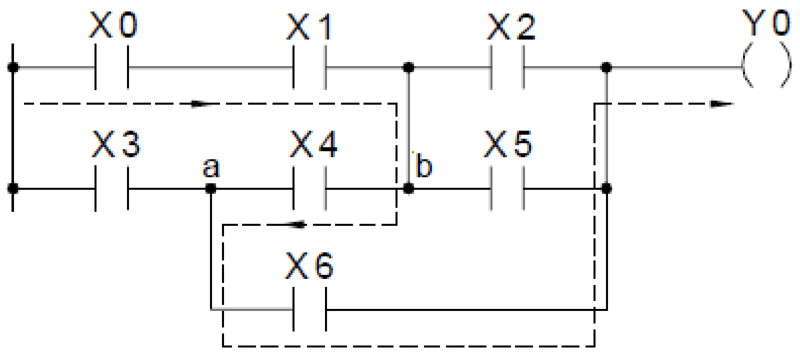
1-2 Differences Between Conventional and PLC Ladder Diagram

Although the basic operation principle for both conventional and PLC Ladder Diagram are the same, but in reality, PLC uses the CPU to emulate the conventional Ladder Diagram operations; that is, PLC uses scanning method to monitor the statuses of input elements and output coils, then uses the Ladder Diagram program to emulate the results which are the same as the results produced by the conventional Ladder Diagram logic operations. There is only one CPU, so the PLC has to sequentially examine and execute the program from its first step to the last step, then returns to the first step again and repeats the operation (cyclic execution). The duration of a single cycle of this operation is called the scan time. The scan time varies with the program size. If the scan time is too long, then input and output delay will occur. Longer delay time may cause big problems in controlling fast response systems. At this time, PLCs with short scan time are required. Therefore, scan time is an important specification for PLCs. Due to the advance in microcomputer and ASIC technologies nowadays the scan speed has been enhanced a great deal. M SERIES PLC takes approximately 1us for IK steps of contact under the condition of continuous address reading, and 5us under the condition of discrete address. The following diagram illustrates the scanning process of a PLC Ladder Diagram.



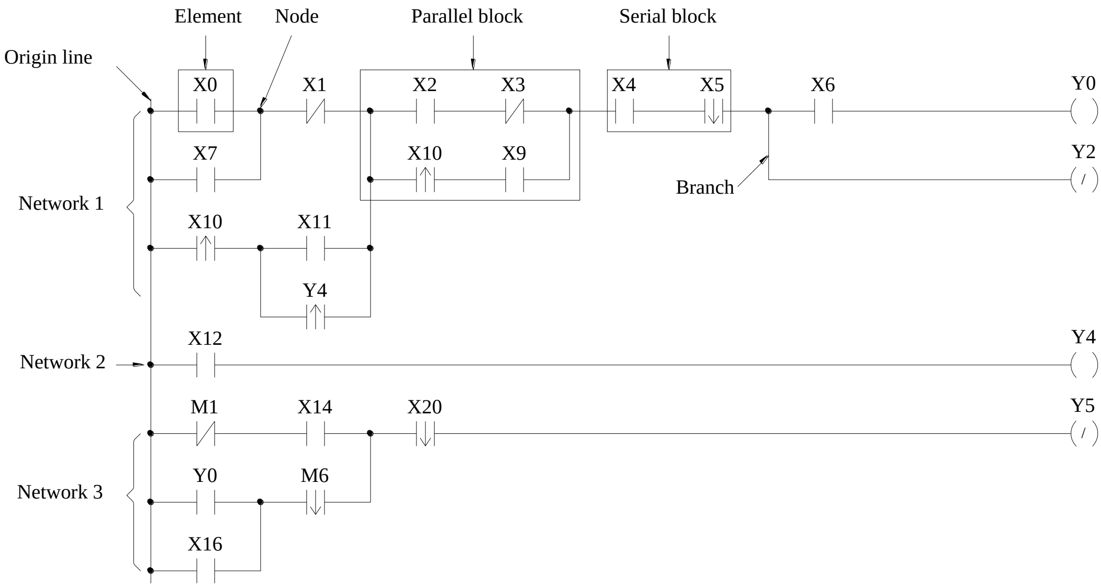
Schematic diagram of PLC ladder diagram program scan

Besides the time scan difference mentioned above, the other difference between the conventional and PLC Ladder Diagram is “Reverse Flow”. As shown in the diagram below, if X0, X1, X4 and X6 are ON, and the remaining elements are OFF: In a conventional Ladder Diagram circuit, a reverse flow route for output Y0 can be defined by the dashed line and Y0 will be ON; while PLC scans from left to right and from top to bottom when the PLC CPU is calculating the result of the ladder diagram program. Under the same input conditions, the state of point “a” in this illustration is considered OFF by the CPU because X3 contact is OFF. Although point a is connected to point “b” via X4 and both are ON, because the PLC ladder diagram only scans from left to right, the CPU Unable to detect, so Y0 output is OFF.



Reverse flow of conventional Ladder diagram

1-3 Ladder Diagram Structure and Terminolog



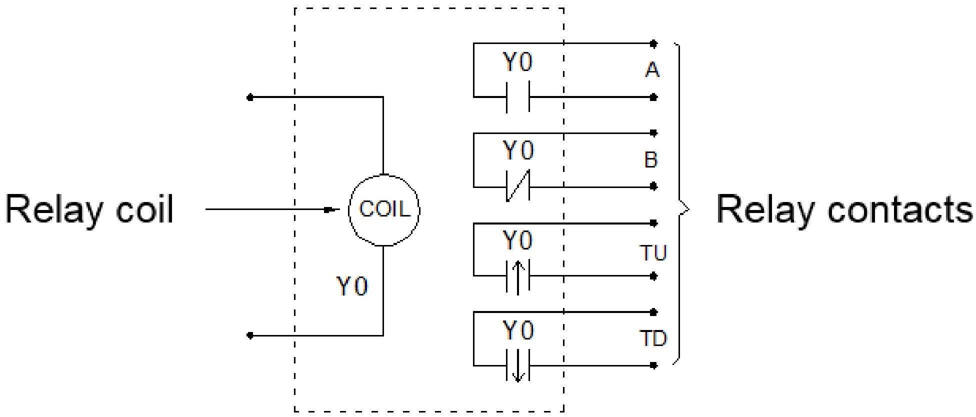
Ladder Diagram Program Example

Note: The maximum size of M SERIES PLC network is 22 columns X 16 rows.

As shown above, the Ladder Diagram can be divided into many small cells. There is total 88 cells (8 rows X 11 columns) for this example Ladder Diagram. One cell can accommodate one element. A completed Ladder Diagram can be formed by connecting all the cells together according to the specific requirements. The terminologies related to Ladder Diagram are illustrated below：

①. Contact
Contact is an element with open or short status. One kind of contact is called "Input Contact"(reference number prefix with X) and its status reference from the external signals (the input signal comes from the input terminal block). Another one is called "Relay contact" and its status reflects the status of relay coil (please refer to ②). The relation between the reference number and the contact status depends on the contact type. The 6 contact elements provided by M series PLC include: A contact, B contact, Up/Down Differential (TU/TD) contacts and Open/Short contacts. Please refer to ④.

②. Relay
Same as the conventional relay, it consists of a Coil and a Contact as shown in the diagram below.



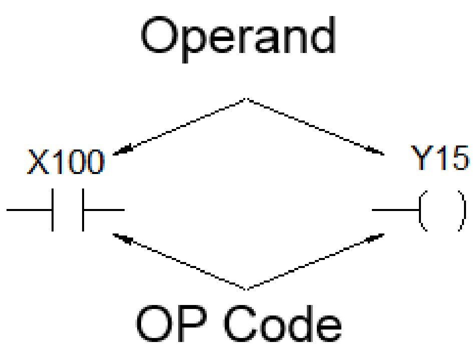
Relay_Coil and Contact

As shown in the figure, the relay must have a coil. To make the relay act, the coil must be driven (by OUT command). After the coil is driven, the state of its contacts will be affected
As shown in the example, if Y0 is driven with 1 (make it ON), then the A contact of the relay is 1, the B contact is 0, the TU contact is only ON for one scan time, and the TD contact is 0.
When Y0 turns OFF, the A contact is 0, the B contact is 1, the TU contact is 0, and the TD contact is only ON for one scan time (for the actions of A, B, TU, and TD contacts, please refer to Chapter 4 "Sequential Instructions").

There are four types of M SERIES PLC relays, namely Y△△△ (output relay), M△△△△(internal relay), S△△△ (step relay) and TR△△ (register relay). The status of output relays will be sent to the output point of terminal block.

③. Origin
The starting line at the left side of the Ladder Diagram.

④. Element
Element is the basic unit of a Ladder Diagram.
An element consists of two parts as shown in the diagram below. One is the element symbol which is called “OP Code” and another is the reference number part which is called “Operand”.



Element

The components of M SERIES PLC have the following 8 types:

Element type	Symbol	Note
A Contact (Normally OPEN)	□△△△△△ ─┤ ├─	□ can be X、Y、M、S 、T 、C (please refer to section 2.2)
B Contact (Normally CLOSE)	□△△△△△ ─┤/├─	
Up Differential Contact	□△△△△△ ─┤↑├─	□ can be X、Y、M、S
Down Differential Contact	□△△△△△ ─┤↓├─	
Open Circuit Contact	– – ∘ ∘	
Short Circuit Contact	──── ● ●	
Output Coil	□△△△△△ ─( )─	□ can be Y、M、S
Inverse Output Coil	□△△△△△ ─(/)─	

M SERIES PLC Elements

Note: Please refer to section 2.2 for the ranges of X, Y, M, S, T and C contacts or coils. Please refer to section 2.2 for the element characteristics.

There is a special sequential instruction: FOn, which is also one of the elements. Please refer to section 5.1.4 “Function Output FO”.

⑤. Node

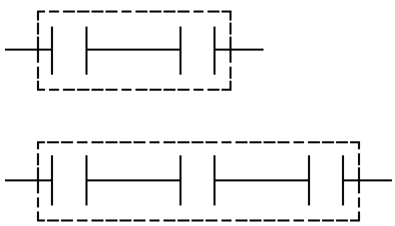
The connection point between two or more elements.

⑥. Block

A circuit consists of two or more elements.

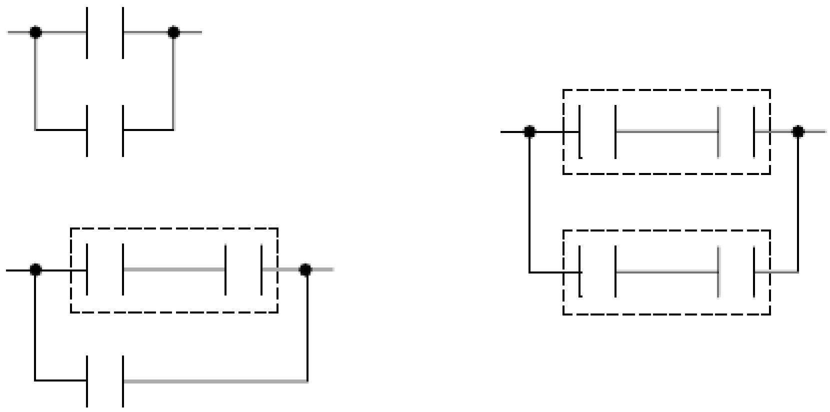
There are two basic types of blocks：

- Serial Block: Two or more elements are connected in series to form a single row circuit.



Serial Block

- Parallel Block: A parallel (rectangular) closed circuit composed of components or series blocks connected in parallel.

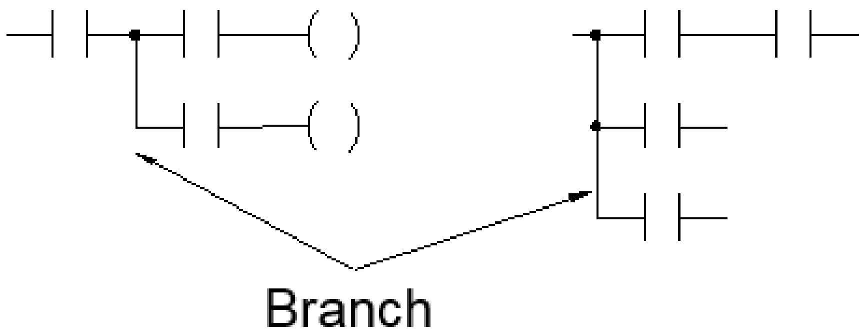


Parallel Block

Note: Complicated block can be formed by the combination of the single element, serial blocks and parallel blocks. When designing a Ladder Diagram with mnemonic entry, it is necessary to break down the circuits into element, serial, and parallel blocks.

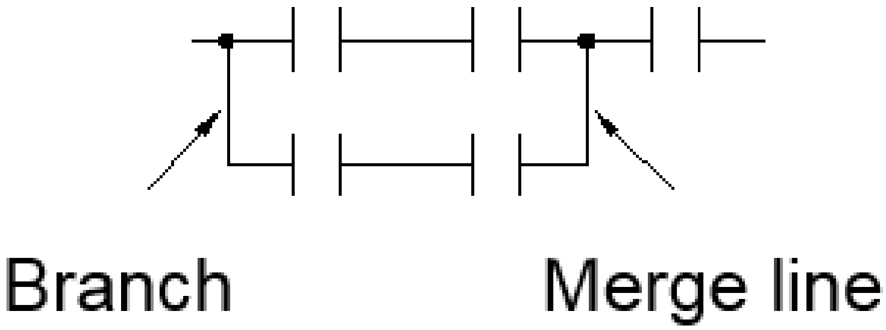
⑦. Branch

If there are two or more loops connected to the right of the vertical line in any network, this is a branch, and this vertical line is called a branch line.



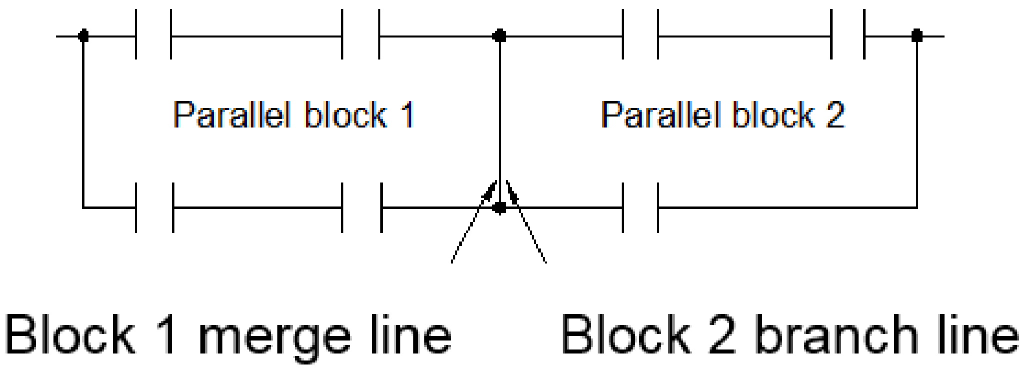
Branch

If there is another vertical line on the right side of the branch line to merge the two branch columns of circuits (this vertical line is called the merging line), then this circuit will form a closed circuit (forming a parallel block), and this circuit is a non-branching circuit.



Branch line and Merge line

If both the right and the left sides of the vertical line are connected with two or more rows of circuits, then it is both a branch line and a merge line as shown in the example below：



For both branch and merge lines

⑧. Network

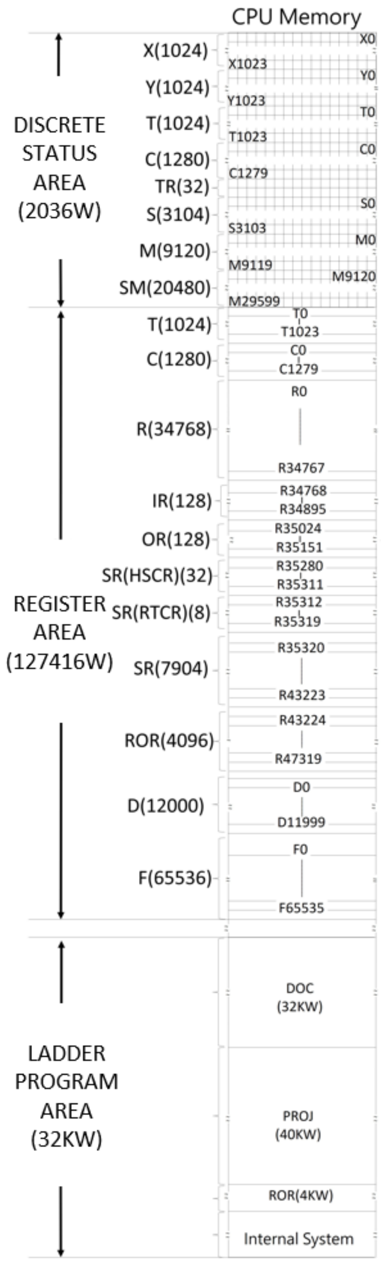
A loop that can perform specific functions is composed of elements, branches, and blocks, which is called a network. A network is the basic unit that can perform complete functions in a ladder diagram program, and a ladder diagram program is composed of a series of networks. The beginning of the network must start from the busbar, and any two columns of circuits without a vertical line connection belong to two different networks (the ones connected by a vertical line belong to the same network). According to this rule, such as the ladder diagram program example, it can be divided into three networks: network 1~3.

2

Details of Memory Configuration, Single Point (Digital) and Register in PLC

※※Being designed with very broad flexibility range, the M-Serial PLC allows the user to access ordinary register field (containing 34768 counts of words) by the indirect addressing method. However, it may easily lead to false data writing issues if the indirect addressing parameters are improperly used. When operated in the Read-only Register Field-ROR (containing 4096 counts of words), the M-Serial PLC does not allow the user to access the register by the indirect addressing method. If the user needs to create important parameter values, it is recommended that the ROR (Read-only Register) Field should be used in order to execute the desired reading and writing according to the respective program commands. The main purpose is to avoid the issues that may be generated due to the incorrect parameters required for the indirect addressing.

2-1 M SERIES PLC Memory Configuration



PLC memory configuration diagram

2-2 Digital and Register Configuration

- This configuration is the factory setting:

Item				Specifications	Note
Single Point (BTI State)	X	Input contact (DI) (Max. point count: 2048 points)		X0 ~ X1023 (1024)	Corresponding to external digital input
	Y	Output relay (DO) (Max. point count: 2048 points)		Y0 ~ Y1023 (1024)	Corresponding to external digital output
	TR	Temporary relay		TR0 ~ TR31(32) (Reserved for system operations)	
	M	Internal relay		M0 ~ M9119 (9120)	M0~M9119 can be configured as retentive or non-retentive relay.
		Special Relay		M9120 ~ M29599 (20480)	
	S	Step Relay		S0 ~ S3103 (3104)	S0 ~ S3103 Can be configured as retentive or non-retentive relay.
	T	Timer "Time-Up" status contact		T0 ~ T1023 (1024)	
	C	Counter "Counter-Up" status contact		C0 ~ C1279 (1280)	
Register (WORD Data)	TMR	Timer current value register	0.001S Time Base	T0 ~ T255 (256) *	T0 ~ T1023 numbers for each time base can be adjusted.
			0.01S Time Base	T256 ~ T511 (256) *	
			0.1S Time Base	T512 ~ T767 (256) *	
			1S Time Base	T768 ~ T1023 (256) *	
	CTR	Counter current value register	16-bit	C0 ~ C1023 (1024)	Can be configured as non-retentive or retentive.
			32-bit	C1024 ~ C1279 (256)	Can be configured as non-retentive or retentive.

	HR DR	Data Register	Retentive	R0 ~ R14999 (15000) * Can be configured as non-retentive D0 ~ D11999 (12000)	
			Non-Retentive	R15000 ~ R34767 (19768)	
	HR ROR		Retentive	R43224 ~ R47319 (4096) * When not configured as ROR, it can serve normal register (for read/write)	
			Read Only Register (ROR)	R43224 ~ R47319 can be set as ROR ~ default setting is "0" *	ROR is stored in special ROR area and not occupy program space
			File Register	F0 ~ F65535(65536) * Save/retrieved via dedicated instruction	
	IR	Input Register (AI)		R34768 ~ R34895 (128)	Corresponding to external analog input
	OR	Output Register (AO)		R35024 ~ R35151 (128)	Corresponding to external analog output
	SR	Special System Register		R35280 ~ R43223 (7944)	

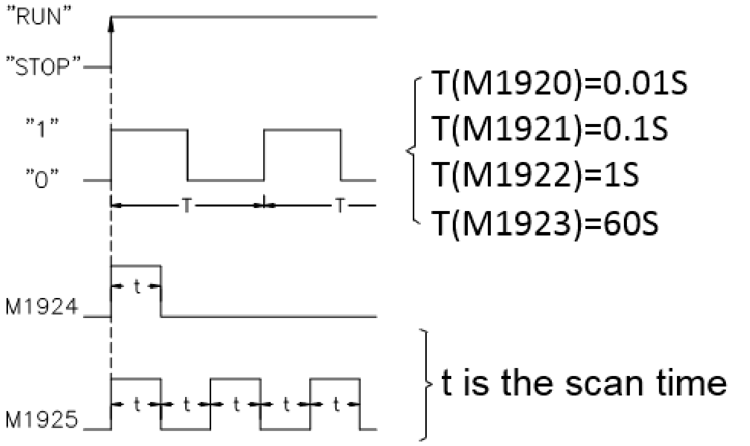
<table>
<tr><td rowspan="9"></td><td rowspan="7">〈Special Register〉</td><td>0.1ms HST Register</td><td colspan="4">R35451~R35466(16)</td><td rowspan="5"></td></tr>
<tr><td>1ms STM Register</td><td colspan="4">R35435~R35442(8)</td></tr>
<tr><td>10ms LTM Register</td><td colspan="4">R35443~R35450(8)</td></tr>
<tr><td>0.1ms HSTA Circulation Counter Register</td><td colspan="4">DR35467</td></tr>
<tr><td>High-Speed Counter Register</td><td colspan="4">R35280~R35311(32)</td></tr>
<tr><td rowspan="2">Calendar Registers</td><td>R35312
(Second)</td><td>R35313
(Minute)</td><td>R35314
(Hour)</td><td>R35315
(Day)</td><td rowspan="2"></td></tr>
<tr><td>R35316
(Month)</td><td>R35317
(Year)</td><td>R35318
(Week)</td><td>R35319
(Hour +
Minute)</td></tr>
<tr><td>FR</td><td>File Register</td><td colspan="4">F0~F65535(65536)</td><td></td></tr>
<tr><td>XR</td><td>Index Register</td><td colspan="4">V: R43214
Z: R43216
P0 ~ P9 :
R43194, R43196, R43198, R43200, R43202,
R43204, R43206, R43208, R43210, R43212</td><td></td></tr>
</table>

Digital and Register Configuration

Note: During power up or changing operation mode from STOP→RUN, all contents in non-retentive relays or registers will be cleared to 0; the retentive relays or registers will remain the same state as before.

2-3 CPU Special Relay Details

Relay No.	Function/TAG Symbol	Description
1. Stop, Prohibit Control		
M9120	Emergency Stop control EMERGENCY_STOP_CTRL	If 1, PLC will be stopped.
2. Disable, Clear Control		
M9121	Reserve	
M9122	Disable Status Retent Select DISABLE_STATUS_RETENT_CTR	Disabled when at 1
M9123	Clear Non-Retentive Relays CLR_NON_RETENT_RELAY	Cleared when at 1
M9124	Clear Retentive Relays CLR_RETENT_RELAY	Cleared when at 1
M9125	Clear Non-Retentive Registers CLR_NON_RETENT_REG	Cleared when at 1
M9126	Clear Retentive Registers CLR_RETENT_REG	Cleared when at 1

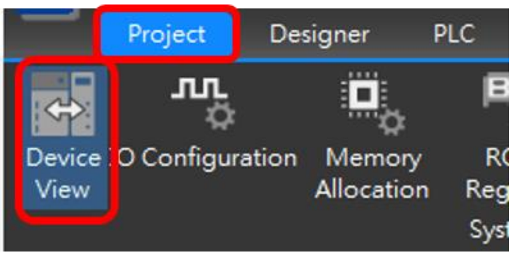
3. Pulse Signals		
M9127 M9218 M9129 M9130	0.01S Clock pulse CLK_PULSE_0_01S 0.1 S Clock pulse CLK_PULSE_0_1S 1 S Clock pulse CLK_PULSE_1S 60 S Clock pulse CLK_PULSE_60S ①	
M9131	Initial Pulse (First Scan) CLK_PULSE_INIT	=0, PLC working at STOP Mode =1, PLC working at RUN Mode
M9132	Scan Cyclic Pulse ③ CLK_PULSE_SCAN	
M9133	PLC Working Mode PLC_WORKING_MODE	
4. Error Messages		
M9134	System Error Warning CPU_ABNL_WARNING	1: Indicating no expansion unit or exceed the limit on number of I/O points
5. Port1~Port2 Controls		
M9135	Port1 Work Indicator COM_BUSY_P1	0: Port 1 Busy 1: Port 1 Ready
M9136	Port 1 Work Indicator COM_DN_P1	1: Complete all communication transactions of FUN151 (CLINK), only one scan is ON.
M9137	Port 1 Communication Status COM_STATUS_P1	Port 1 has received and transmitted a message
M9138	Port 2 Work Indicator COM_BUSY_P2	0: Port 2 Busy 1: Port 2 Ready
M9139	Port 2 Work Indicator COM_DN_P2	1: Complete all communication transactions of FUN151 (CLINK), only one scan is ON.
M9140	Port 2 Communication Status COM_STATUS_P2	1: Port 2 has received and transmitted a message

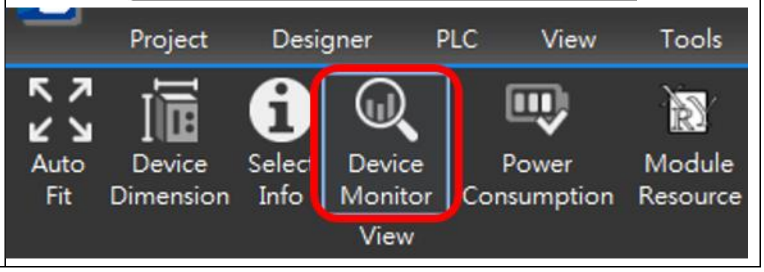
6. HSC0～HSC7 Controls		
M9141	HSC0 Software Mask HSC0_MSK	1: Mask
M9142	HSC0 Software Clear HSC0_CLR	1: Clear
M9143	HSC1 Software Mask HSC1_MSK	1: Mask
M9144	HSC1 Software Clear HSC1_CLR	1: Clear
M9145	HSC2 Software Mask HSC2_MSK	1: Mask
M9146	HSC2 Software Clear HSC2_CLR	1: Clear
M9147	HSC3 Software Mask HSC3_MSK	1: Mask
M9148	HSC3 Software Clear HSC3_CLR	1: Clear
M9149~ M9157	Reserved	

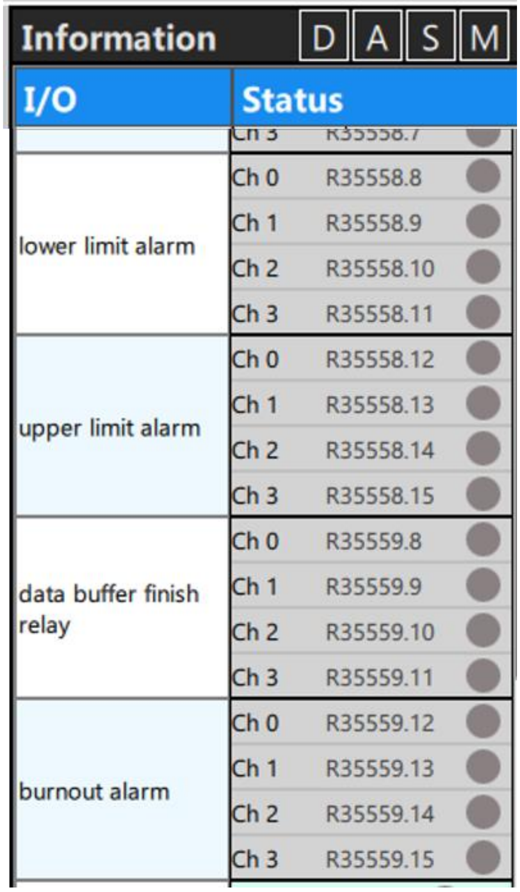
7. Communication/Timing/Counting Controls		
M9158	The CV value control after the timer “Time-Up” HST_TIME_UP_MODE	0: The CV value will continue timing until the upper limit is met after “Time-Up”. 1: The CV value will stop at the PV value after “Time-Up” (User may control M9158 within the program to control the individual timer)
M9159	The CV value control after the counter “Count-Up” HSC_COUNT_UP_MODE	0: The CV value will continue counting up to the upper limit after “Count-Up”. 1: The CV value will stop at the PV value after “Count-Up” (User may control M9159 within the program to control the individual counter)
M9160	CAM Function Cross 0 Degree Selection CAM_FUNC_SELECT	M9160=1: When the upper limit value of the FUN 112 (BKCMP) command is less than the lower limit value, it can be executed (for example, the upper limit value is 10°, the lower limit value is 350°, when the current angle is 350°~10°, the comparison bit is 1). M9160=0： If among the upper and lower limit setting values, the upper limit value is less than the lower limit value, the limit value error flag The number "ERR" is set to 1, and the comparison output of this group is 0.
M9161	High-Speed Pulse Output Stop Selection HSPSO_STOP_SELECT	
M9162	Update MODBUS Planning MODBUS_UPDATE	
M9163	Update COM Setting COM_UPDATE	
M9164	Reboot Network Interface ETH_UPDATE	
M9165	Enable DHCP ETH_DHCP_ENABLE	
M9166	1ms Timer STM 0 Control STM0_CTRL	
M9167	1ms Timer STM 1 Control STM1_CTRL	

M9168	1ms Timer STM 2 Control STM2_CTRL	
M9169	1ms Timer STM 3 Control STM3_CTRL	
M9170	10ms Timer LTM 0 Control LTM0_CTRL	
M9171	10ms Timer LTM 1 Control LTM1_CTRL	
M9172	10ms Timer LTM 2 Control LTM2_CTRL	
M9173	10ms Timer LTM 3 Control LTM3_CTRL	
M9174	0.1 ms HST 0 Control HST0_CTRL	
M9175	0.1ms HST 1 Control HST1_CTRL	
M9176	0.1ms HST 2 Control HST2_CTRL	
M9177	0.1ms HST 3 Control HST3_CTRL	
M9178	0.1ms HSTA Circulation Counter Control HSTA_CTRL	
8. RTC Control		
M9179 M9180 M9181 M9182	RTC Setting RTC_UPDATE 30 S Adjustment RTC_30S_ADJUSTMENT RTC Installation Checking RTC_INSTALL_CHK Set Value Error RTC_SET_VALUE_ERROR	
9. PS0~7 Control		
M9183 M9184	PSO0 Indicator PSO0_BUSY PSO1 Indicator PSO1_BUSY	

M9185 M9186	PSO2 Indicator PSO2_BUSY PSO3 Indicator PSO3_BUSY	
M9187 M9188 M9189 M9190	PSO0 Done PSO0_DN PSO1 Done PSO1_DN PSO2 Done PSO2_DN PSO3 Done PSO3_DN	
M9191 M9192 M9193 M9194	PSO4 Indicator PSO4_BUSY PSO5 Indicator PSO5_BUSY PSO6 Indicator PSO6_BUSY PSO7 Indicator PSO7_BUSY	
M9195 M9196 M9197 M9198	PSO4 Done PSO4_DN PSO5 Done PSO5_DN PSO6 Done PSO6_DN PSO7 Done PSO7_DN	

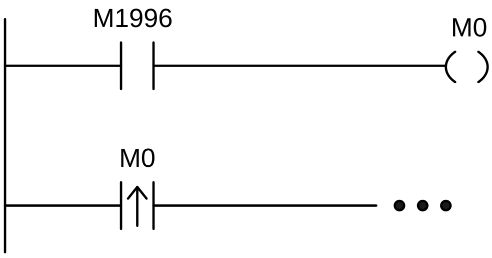
<table>
<tr><td colspan="3">10. Expansion Module Operation Field</td></tr>
<tr><td>M9199~
M10511</td><td>Please refer to the respective Expansion Module User Manual.</td><td>Because the number of special registers is related to the expansion module that will be set by the user, the sequence is not set with a fixed number order. Therefore, it will be learned through the following method:
The number of Special Register can be displayed by clicking on the following profile: “Project-> Device View->Device Monitor
->select desired module.”
The data indicated below are explained by using Data Buffer Relay as the example. The Data Buffer Relay will be started with the same method as the Triggering Data Buffer Relay.



</td></tr>
</table>

<table>
<tr><td></td><td></td><td>
</td></tr>
<tr><td>M10512~
M16095</td><td>For Motion related special relays</td><td></td></tr>
</table>

CPU Module special relay list

※All special relays do not provide Up/Down differential contact commands TU. If it is necessary to perform differential action on the special relay, it can be replaced by an indirect method. (Refer to the picture below)



Differential Action Connection of Special Relay

Note: All special relays or registers attached with "◤" symbol shown in the above table are write prohibited. At the same time, this type of relay still prohibits/disables control and mandatory setting, and does not provide TU and TD contacts.

2-4 CPU Special Registers Details

Register No./ System Tag Code	Function/System Tag Symbol	Description
R35280	HSC0 current value Low word HSC0_CV	
R35281	HSC0 current value High word HSC0_CV	
R35282	HSC0 preset value Low word HSC0_PV	
R35283	HSC0 preset value High word HSC0_PV	
R35284	HSC1 current value Low word HSC1_CV	
R35285	HSC1 current value High word HSC1_CV	
R35286	HSC1 preset value Low word HSC1_PV	
R35287	HSC1 preset value High word HSC1_PV	
R35288	HSC2 current value Low word HSC2_CV	
R35289	HSC2 current value High word HSC2_CV	
R35290	HSC2 preset value Low word HSC2_PV	
R35291	HSC2 preset value High word HSC2_PV	
R35292	HSC3 current value Low word HSC3_CV	
R35293	HSC3 current value High word HSC3_CV	
R35294	HSC3 preset value Low word HSC3_PV	
R35295	HSC3 preset value High word HSC3_PV	

Register No./ System Tag Code	Function/System Tag Symbol	Description
R35296	HSC4 current value Low word HSC4_CV	
R35297	HSC4 current value High word HSC4_CV	
R35298	Reserved	
R35299	Reserved	
R35300	HSC5 current value Low word HSC5_CV	
R35301	HSC5 current value High word HSC5_CV	
R35302	Reserved	
R35303	Reserved	
R35304	HSC6 current value Low word HSC6_CV	
R35305	HSC6 current value High word HSC6_CV	
R35306	Reserved	
R35307	Reserved	
R35308	HSC7 current value Low word HSC7_CV	
R35309	HSC7 current value High word HSC7_CV	
R35310	Reserved	
R35311	Reserved	
R35312	Second of calendar RTC_SECOND	
R35313	Minute of RTC RTC_MINUTE	

R35314	Hour of RTC RTC_HOUR	
R35315	Date of RTC RTC_DAY	
R35316	Month of RTC RTC_MONTH	
R35317	Year of RTC RTC_YEAR	
R35318	Week of RTC RTC_DAY_OF_WEEK	
R35319	Hour (High byte) + Minute (Low byte) RTC_HOUR_MINUTE	
R35320	Error code of PSO0 PSO0_ERR_CODE	
R35321	Error code of PSO1 PSO1_ERR_CODE	
R35322	Error code of PSO2 PSO2_ERR_CODE	
R35323	Error code of PSO3 PSO3_ERR_CODE	
R35324	Completed step number of positioning program for PSO0 PSO0_DN_STEP_NUM	
R35325	Completed step number of positioning program for PSO1 PSO1_DN_STEP_NUM	
R35326	Completed step number of positioning program for PSO2 PSO2_DN_STEP_NUM	

R35327	Completed step number of positioning program for PSO3 PSO3_DN_STEP_NUM	
R35328	Output frequency for Low Word of PSO0 PSO0_CUR_FREQ	-
R35329	Output frequency for High Word of PSO0 PSO0_CUR_FREQ	-
R35330	Output frequency for Low Word of PSO1 PSO1_CUR_FREQ	-
R35331	Output frequency for High Word of PSO1 PSO1_CUR_FREQ	-
R35332	Output frequency for Low Word of PSO2 PSO2_CUR_FREQ	-
R35333	Output frequency for High Word of PSO2 PSO2_CUR_FREQ	-
R35334	Output frequency for Low Word of PSO3 PSO3_CUR_FREQ	-
R35335	Output frequency for High Word of PSO3 PSO3_CUR_FREQ	-
R35336	Current pulse position for Low Word of PSO0 PSO0_CUR_POS	
R35337	Current pulse position for High Word of PSO0 PSO0_CUR_POS	

R35338	Current pulse position for Low Word of PSO1 PSO1_CUR_POS	
R35339	Current pulse position for High Word of PSO1 PSO1_CUR_POS	
R35340	Current pulse position for Low Word of PSO2 PSO2_CUR_POS	
R35341	Current pulse position for High Word of PSO2 PSO2_CUR_POS	
R35342	Current pulse position for Low Word of PSO3 PSO3_CUR_POS	
R35343	Current pulse position for High Word of PSO3 PSO3_CUR_POS	
R35344	Pulse count remaining for output for Low Word of PSO0 PSO0_REMAINING_COUNT	
R35345	Pulse count remaining for output for High Word of PSO0 PSO0_REMAINING_COUNT	
R35346	Pulse count remaining for output for Low Word of PSO1 PSO1_REMAINING_COUNT	
R35347	Pulse count remaining for output for High Word of PSO1 PSO1_REMAINING_COUNT	
R35348	Pulse count remaining for output for Low Word of PSO2 PSO2_REMAINING_COUNT	

R35349	Pulse count remaining for output for High Word of PSO2 PSO2_REMAINING_COUNT	
R35350	Pulse count remaining for output for Low Word of PSO3 PSO3_REMAINING_COUNT	
R35351	Pulse count remaining for output for High Word of PSO3 PSO3_REMAINING_COUNT	
R35352	COM1 Communication Parameters Setting COM_PARAM_P1	Set Baud Rate, Data bit… of Port 1
R35353	COM2 Communication Parameters Setting COM_PARAM_P2	Set Baud Rate, Data bit… of Port 2
R35354	COM1 & COM2 connection setting COM_STN_CHK_P1 COM_STN_CHK_P2	● Low Byte of R35354: =1, Port 1 without station number checking for FATEK's external communication protocol (communicating with MMI/SCADA) ≠1,Port 1 checks station number, it allows multi-drop network for data acquisition ● High Byte of R35354: =1, Port 2 without station number checking for FATEK's external communication protocol (communicating with MMI/SCADA) ≠1,Port 2 checks station number, it allows multi-drop network for data acquisition.

R35355	Communication protocol setting for COM1 and COM2 COM_PROTOCOL	Set Port1 and Port2 as the FATEK or Modbus RTU/ASCII communication protocol
R35356	Reserved	
R35357	Transmission delay and reception error detection time setting when COM1 is used as the master station COM_TX_DELAY_P1	
R35358	Transmission delay and reception error detection time setting when COM2 is used as the master station COM_TX_DELAY_P2	
R35359	Reserved	

R35360	System error indication (Need to confirm with LED) CPU_ERROR	see table below

Item	ERR1/ERR 2 LED	SR/MASK
Out of Memory	ON/ON	Reserved
Initialization Error	ON/ON	Reserved
Systen Error	ON/ON	Reserved
System Stack Error	ON/ON	R35361/0x0200
System Check Code Error Indication	ON/ON	R35361/0x0004
Power-on detection is power-off for trial operation	ON/ON	R35360/0x0400
System Check Code Error Indication	ON/ON	R35362/0x3E00
Expansion Module Detection Error	ON/OFF	R35360/0x0001
Expansion Module Configuration File Error	ON/OFF	R35360/0x0002
The number of expansion modules does not match the host project	ON/OFF	R35360/0x0004
Expansion module I/O points out of range	ON/OFF	R35360/0x0008
The number of expansion modules exceeds the range	ON/OFF	R35361/0x0100
Motion Control Unit Queue Error	ON/OFF	R35360/0x0080
Motion Control Unit Overflow Error	ON/OFF	R35360/0x0100
Motion Control Unit Emergency Stop	ON/OFF	R35360/0x0200
Watchdog Reset Check	ON/OFF	R35361/0x0010
Invalid Memory Card Detection Indication	OFF/ON	R35360/0x0010
Memory Card Operation Error Indication	OFF/ON	R35360/0x0020
PLC ID does not match PROG ID	OFF/ON	R35360/0x0040
The application exceeds the capabilities of this CPU	OFF/ON	R35361/0x0800
System Service Error Indication	OFF/ON	R35361/0x8000

R35361 ~R35362	CPU Status Indication CPU_STATUS	BIT0: CPU RUN or Stop BIT1: Resvered BIT2: System Check Code Error BIT3: Memory Card Ready display BIT4: Watch-Dog Error BIT5: Motion Control Unit Detection BIT6: PLC ID Protection BIT7: Emergency Stop BIT8: Number of expansion module exceeds the scope BIT9: System STACK Error BIT10: Resvered BIT11: Function(s) existed that CPU does not support BIT12: Resvered BIT13: Resvered BIT14: RTC Ready Indicator BIT15: System Service Error Indicator BIT16: PLC ID Setting State BIT17: Program ID Setting State BIT18: Mian Program Password Setup State BIT19: Subroutine Password Setup State BIT20: PLC Upload Password Setup State BIT21: PLC Download Password Setup State BIT22: CIC Setup State BIT23: Resvered BIT24: Resvered BIT25~29: System Check Code Error Indicator BIT30: Switch State BIT31: Resvered

R35363	PLC station number display or setup PLC_STATION_NUM	● If high byte is not equal 55H, R35363 will show the station number of this PLC. ● When the high byte of register R35363 is equal to 55H, the low byte of R35363 is used to set the station number of this PLC.
R35364	PLC OS Version (MAJOR NO.) PLC_OS_VER_MAJOR	
R35365	PLC OS Version (MINOR NO + PATCH NO) PLC_OS_VER_MINOR PLC_OS_VER_PATCH	
R35366	Host model information (UNIT ID + MODEL) MAIN_UNIT_MODEL	
R35367	Power ON Delay (0.01s unit) POWER_ON_DELAY	● PLC is ready for I/O service after this delay time while power up. The unit is in 0.01S. The default value is 100.
R35368	Power Off Counter POWER_OFF_COUNTER	
R35369	Reserved	
R35370	Current Scan Time SCAN_TIME_CURRENT	1. Error < ±1ms 2. Re-calculate when PLC changes from STOP to RUN
R35371	Maximum Scan Time SCAN_TIME_MAX	
R35372	Minimum scan time SCAN_TIME_MIN	
R35373	Fixed Scan Time SCAN_TIME_SETTING	-
R35374	Expansion Module Heart Beat Detection (Rack 1) EXP_HEARTBEAT_RACK1	
R35375	Expansion Module Heart Beat Detection (Rack 2) EXP_HEARTBEAT_RACK2	
R35376	Expansion Module Heart Beat Detection (Rack 3) EXP_HEARTBEAT_RACK3	

R35377	Expansion Module Heart Beat Detection (Rack 4) EXP_HEARTBEAT_RACK4	
R35378	Number of expansion AI points EXP_AI_POINTS	
R35379	Number of expansion AO points EXP_AO_POINTS	
R35380	Number of expansion DI points EXP_DI_POINTS	
R35381	Number of expansion DO points EXP_DO_POINTS	
R35382	CPU Ethernet Port IP Address OCT1 (Leading) ETH_IP_OCT1	-
R35383	CPU Ethernet Port IP Address OCT2 ETH_IP_OCT2	
R35384	CPU Ethernet Port IP Address OCT3 ETH_IP_OCT3	
R35385	CPU Ethernet Port IP Address OCT4 ETH_IP_OCT4	
R35386	CPU Ethernet Port Mask OCT1 (Leading) ETH_SUBMASK_OCT1	
R35387	CPU Ethernet Port Mask OCT2 ETH_SUBMASK_OCT2	
R35388	CPU Ethernet Port Mask OCT3 ETH_SUBMASK_OCT3	
R35389	CPU Ethernet Port Mask OCT4 ETH_SUBMASK_OCT4	
R35390	CPU Ethernet Port Router OCT1 (Leading) ETH_GATEWAY_OCT1	
R35391	CPU Ethernet Port Router OCT2 ETH_GATEWAY_OCT2	
R35392	CPU Ethernet Port Router OCT3 ETH_GATEWAY_OCT3	

R35393	CPU Ethernet Port Router OCT4 ETH_GATEWAY_OCT4	
R35394	CPU Ethernet Primary DNS OCT1 (Leading) ETH_PRIM_DNS_OCT1	
R35395	CPU Ethernet Primary DNS OCT2 ETH_PRIM_DNS_OCT2	
R35396	CPU Ethernet Primary DNS OCT3 ETH_PRIM_DNS_OCT3	
R35397	CPU Ethernet Primary DNS OCT4 ETH_PRIM_DNS_OCT4	
R35398	CPU Ethernet Secondary DNS OCT1(Leading) ETH_SEC_DNS_OCT1	
R35399	CPU Ethernet Secondary DNS OCT2 ETH_SEC_DNS_OCT2	
R35400	CPU Ethernet Secondary DNS OCT3 ETH_SEC_DNS_OCT3	
R35401	CPU Ethernet Secondary DNS OCT4 ETH_SEC_DNS_OCT4	
R35402	Modbus: Y Starting Address MODBUS_COIL_ADDR_Y	
R35403	Modbus: Coil Starting Address MODBUS_COIL_Y	
R35404	Modbus: Corresponding Length MODBUS_ COIL_TOTALS_Y	
R35405	Modbus: X Starting Address MODBUS_ COIL_ADDR_X	
R35406	Modbus: Coil Starting Address MODBUS_COIL_X	
R35407	Modbus: Corresponding Length MODBUS_ COIL_TOTALS_X	
R35408	Modbus: M Starting Address MODBUS_ COIL_ADDR_M	
R35409	Modbus: Coil Starting Address MODBUS_COIL_M	
R35410	Modbus: Corresponding Length MODBUS_ COIL_TOTALS_M	

R35411	Modbus: S Starting Address MODBUS_ COIL_ADDR_S	
R35412	Modbus: Coil Starting Address MODBUS_COIL_S	
R35413	Modbus: Corresponding Length MODBUS_ COIL_TOTALS_S	
R35414	Modbus: T starting address MODBUS_ COIL_ADDR_T	
R35415	Modbus: Coil Starting Address MODBUS_COIL_T	
R35416	Modbus: Corresponding Length MODBUS_ COIL_TOTALS_T	
R35417	Modbus: C Starting Address MODBUS_ COIL_ADDR_C	
R35418	Modbus: Coil Starting Address MODBUS_COIL_C	
R35419	Modbus: Corresponding Length MODBUS_ COIL_TOTALS_C	
R35420	Modbus: R Starting Address MODBUS _HOLDING_ADDR_R	
R35421	Modbus: Holding Starting Address MODBUS_HOLDING_R	
R35422	Modbus: Corresponding Length MODBUS_HOLDING _TOTALS_R	
R35423	Modbus: D Starting Address MODBUS _HOLDING_ADDR_D	
R35424	Modbus: Holding Starting Address MODBUS_HOLDING_D	
R35425	Modbus: Corresponding Length MODBUS _HOLDING_TOTALS_D	
R35426	Modbus: RT Starting Address MODBUS _HOLDING_ADDR_RT	
R35427	Modbus: Holding Starting Address MODBUS_HOLDING_RT	
R35428	Modbus: Corresponding Length MODBUS _HOLDING_TOTALS_RT	
R35429	Modbus: RC Starting Address MODBUS _HOLDING_ADDR_RC	

R35430	Modbus: Holding Starting Address MODBUS_HOLDING_RC	
R35431	Modbus: Corresponding Length MODBUS _HOLDING_TOTALS_RC	
R35432	Modbus: LC Starting Address MODBUS _HOLDING_ADDR_DRC	
R35433	Modbus: Holding Starting Address MODBUS_HOLDING_DRC	
R35434	Modbus: Corresponding Length MODBUS _HOLDING_TOTALS_DRC	
R35435	1ms Timer STM 0 Cycle Setting STM0_PV	
R35436	1ms Timer STM 0 Current Time STM0_CV	
R35437	1ms Timer STM 1 Cycle Setting STM1_PV	
R35438	1ms Timer STM 1 Current Time STM1_CV	
R35439	1ms Timer STM 2 Cycle Setting STM2_PV	
R35440	1ms Timer STM 2 Current Time STM2_CV	
R35441	1ms Timer STM 3 Cycle Setting STM3_PV	
R35442	1ms Timer STM 3 Current Time STM3_CV	
R35443	10 ms Timer STM 0 Cycle Setting LTM0_PV	
R35444	10 msTimer STM 0 Current Time LTM0_CV	
R35445	10 ms Timer STM 1 Cycle Setting LTM1_PV	
R35446	10 msTimer STM 1 Current Time LTM1_CV	
R35447	10 ms Timer STM 2 Cycle Setting LTM2_PV	
R35448	10 msTimer STM 2 Current Time LTM2_CV	

R35449	10 ms Timer STM 3 Cycle Setting LTM3_PV	
R35450	10 msTimer STM 3 Current Time LTM3_CV	
R35451	0.1ms Timer HST 0 Cycle Setting LOW WORD HST0_PV	
R35452	0.1ms Timer HST 0 Cycle Setting HIGH WORD HST0_PV	
R35453	0.1ms Timer HST 0 Current Time LOW WORD HST0_CV	
R35454	0.1ms Timer HST 0 Current Time HIGH WORD HST0_CV	
R35455	0.1ms Timer HST 1 Cycle Setting LOW WORD HST1_PV	
R35456	0.1ms Timer HST 1 Cycle Setting HIGH WORD HST1_PV	
R35457	0.1ms Timer HST 1 Current Time LOW WORD HST1_CV	
R35458	0.1ms Timer HST 1 Current Time HIGH WORD HST1_CV	
R35459	0.1ms Timer HST 2 Cycle Setting LOW WORD HST2_PV	
R35460	0.1ms Timer HST 2 Cycle Setting HIGH WORD HST2_PV	
R35461	0.1ms Timer HST 2 Current Time LOW WORD HST2_CV	

R35462	0.1ms Timer HST 2 Current Time HIGH WORD HST2_CV	
R35463	0.1ms Timer HST 3 Cycle Setting LOW WORD HST3_PV	
R35464	0.1ms Timer HST 3 Cycle Setting HIGH WORD HST3_PV	
R35465	0.1ms Timer HST 3 Current Time LOW WORD HST3_CV	
R35466	0.1ms Timer HST 3 Current Time HIGH WORD HST3_CV	
R35467	0.1ms HSTA HSTA Current Count LOW WORD HSTA_CV	
R35468	0.1ms HSTA HSTA Current Count HIGH WORD HSTA_CV	

R35469- R35478	It is used for designating the Data Register that should be replicated in the SD Card for reading, and the user needs to create such field before replicating the SD Card. After turning on the PC, it will execute the required action according to SR18~SR27 that have been replicated in the SD Card. SD_GROUP_FLAG SD_GROUP_COUNT SD_GROUP_LEN1 SD_GROUP_ADDR1 SD_GROUP_LEN2 SD_GROUP_ADDR2 SD_GROUP_LEN3 SD_GROUP_ADDR3 SD_GROUP_LEN4 SD_GROUP_ADDR4	When using ROM Pack to save the Ladder program and the data register, this table should be used to determine the registers that should be replicated. When turning on the PC, it will be read by ROM Pack for executing the required initialization procedure.
R35479	Control the register to be read by SD Card. Determine if the data register in the PACK should be read when turning on the PC. SD_GROUP_LOAD_FLAG	=5530H: When turning on the PC, it will not read the data register that has been replicated to ROM Pack. = Other value: When turning on the PC, the content of the data register being replicated to ROM Pack will be initialized as the value when the register is replicated.
R35480	Test-run modification mode or replicate the SD Card related command and the state SD_STATE	
R35481	User-defined TCP port of Fatek binary server ETH_FATEK_CUSTOM_PORT	
R35482	User-defined TCP port of Modbus TCP server ETH_MODBUS_CUSTOM_PORT	

R35483	iMonitor Connection Status IMONITOR_STATUS	0: Offline 1: Online 2: Connecting Others: Error code
R35484- R35786	Host MAC address SYS_MAC1 SYS_MAC2 SYS_MAC3	
R35487- R35643	Reserve	
R35644	SD Operation Information Word Group High byte: State Code Low byte: Operation Code SD_OPERATION_STATUS	
R35645	Build-in Analog Input Channel 0 Read Value (M2 Type) PLC_AI0	
R35646	Build-in Analog Input Channel 1 Read Value (M2 Type) PLC_AI1	
R35647	Error code of PSO 4 PSO4_ERR_CODE	
R35648	Error code of PSO 5 PSO5_ERR_CODE	
R35649	Error code of PSO 6 PSO6_ERR_CODE	
R35650	Error code of PSO 7 PSO7_ERR_CODE	
R35651	Completed step number of positioning program for PSO4 PSO4_DN_STEP_NUM	

R35652	Completed step number of positioning program for PSO5 PSO5_DN_STEP_NUM	
R35653	Completed step number of positioning program for PSO6 PSO6_DN_STEP_NUM	
R35654	Completed step number of positioning program for PSO7 PSO7_DN_STEP_NUM	
R35655	Output frequency for Low Word of PSO4 PSO4_CUR_FREQ	
R35656	Output frequency for High Word of PSO4 PSO4_CUR_FREQ	
R35657	Output frequency for Low Word of PSO5 PSO5_CUR_FREQ	
R35658	Output frequency for High Word of PSO5 PSO5_CUR_FREQ	
R35659	Output frequency for Low Word of PSO6 PSO6_CUR_FREQ	
R35660	Output frequency for High Word of PSO6 PSO6_CUR_FREQ	
R35661	Output frequency for Low Word of PSO7 PSO7_CUR_FREQ	
R35662	Output frequency for High Word of PSO7 PSO7_CUR_FREQ	
R35663	Current pulse position for Low Word of PSO4 PSO4_CUR_POS	
R35664	Current pulse position for High Word of PSO4 PSO4_CUR_POS	
R35665	Current pulse position for Low Word of PSO5 PSO5_CUR_POS	
R35666	Current pulse position for High Word of PSO5 PSO5_CUR_POS	
R35667	Current pulse position for Low Word of PSO6 PSO6_CUR_POS	

R35668	Current pulse position for High Word of PSO6 PSO6_CUR_POS	
R35669	Current pulse position for Low Word of PSO7 PSO7_CUR_POS	
R35670	Current pulse position for High Word of PSO7 PSO7_CUR_POS	
R35671	Pulse count remaining for output for Low Word of PSO4 PSO4_REMAINING_COUNT	
R35672	Pulse count remaining for output for High Word of PSO4 PSO4_REMAINING_COUNT	
R35673	Pulse count remaining for output for Low Word of PSO5 PSO5_REMAINING_COUNT	
R35674	Pulse count remaining for output for High Word of PSO5 PSO5_REMAINING_COUNT	
R35675	Pulse count remaining for output for Low Word of PSO6 PSO6_REMAINING_COUNT	
R35676	Pulse count remaining for output for High Word of PSO6 PSO6_REMAINING_COUNT	
R35677	Pulse count remaining for output for Low Word of PSO7 PSO7_REMAINING_COUNT	
R35678	Pulse count remaining for output for High Word of PSO7 PSO7_REMAINING_COUNT	

R35679	MQTT Connection Status MQTT_STATUS	MQTT_CONNECT_ACCEPTED = 0, MQTT_CONNECT_REFUSED_PROTOCOL_VERSION = 1, MQTT_CONNECT_REFUSED_IDENTIFIER = 2, MQTT_CONNECT_REFUSED_SERVER = 3, MQTT_CONNECT_REFUSED_USERNAME_PASS = 4, MQTT_CONNECT_REFUSED_NOT_AUTHORIZED_ = 5, MQTT_CONNECT_DISCONNECTED = 256, MQTT_CONNECT_TIMEOUT = 257
R35680~ R35760	Reserved	
R35761	Able to dynamically change the high-speed pulse output frequency HSPO_FREQ_CTRL	
R35762	COM2 high-speed network enable/mode	Low Bit=55H, High-speed slave station enable =Others, Disabled High Bit=55H, High-speed master station communication with one-table cycle =Others, Continuous

R35763	COM2 Receive/Transmit Timeout Setting (High-Speed) COM_RX_TX_TIME_P2H	When the value of the high byte is not 56H, the system will generate appropriate settings based on the communication parameters set in R4161, and the user does not need to configure them. When the value of the high byte is 56H, the low byte is reserved for manual settings when the system configuration does not meet the requirements of use.R4160[5-0] : Reserve R4160[7-6]：Set TX TMO, =0, 500us ; =1, 700us ; =2, 900us ; =3, 1100us FBs high-speed communication Baud Rate and the automatic setting for Rx/Tx time out values is as follows: For reference: Baud Rate = 38400, Rx = 8 bit time , Tx = 700us Baud Rate = 153600, Rx = 16 bit time , Tx = 700us Baud Rate = 614400, Rx = 20 bit time , Tx = 700us
R35764	Type of FUN30 system analog value bit PID_AI_RESOLUTION	=0, 12bit =1, 14bit =Other, Nbit
R35765	Gain constant of FUN30 PID_GAIN	
R35766~ R35821	Reserved	
R35822~ R35871	Expansion module calibration reserved registers	
R35872~ R36871	Starting register of expansion module status	

R36872~ R36879	TEST RUN Reserve Register (Read-Only)	
R36880~ R36979	For Motion related special Registers	
R36980-	Motion axis data start (read-only)	
R43194~ R43213	P0 (R43194), Reserved(R43195), P1(R43196), Reserved(R43197), P2(R43198), ... P9 (R43212), Reserved(R43213) INDEX_P0 INDEX_P1 INDEX_P2 INDEX_P3 INDEX_P4 INDEX_P5 INDEX_P6 INDEX_P7 INDEX_P8 INDEX_P9	
R43214	V INDEX_V	
R43216	Z INDEX_Z	

2-5 Motion Special Relay Details

※ For more detailed information, please refer to MPLC Motion User Manual Chapter 2 〝Motion Parameters and Status (Special Register and Relay)〞

※ The table lists the register numbers for axis 1, and for each subsequent axis, the register number is obtained by adding 40*(n-1), 'n' represents the axis number.

Relay No.	Function	System Tag Symbol
M10520	All axes: Servo ON	ALL_SERVO_ON
M10521	All axes: Servo Reset	ALL_SERVO_FAULT_RST
M10522	Write all mapping parameters during initialization	MAP_PARM_DURING_INIT
M10523	Restart motion control card	RESTART_MOTION_CARD
M10524	Custom PDO packet with special registers	CUSTOM_PDO_PACKET
M10600 + 40*(n-1)	Axis control command: Servo ON	AX1_SERVO_ON
M10601+ 40*(n-1)	Axis control command: Fault Reset	AX1_FAULT_RST
M10602+ 40*(n-1)	Axis control command: Deceleration Stop	AX1_DEC_STOP
M10603+ 40*(n-1)	Axis control command: Emergency stop	AX1_EMG_STOP
M10604+ 40*(n-1)	Axis Synchronous main clutch ON	AX1_SYNC_ON
M10605+ 40*(n-1)	Axis Origin On	AX1_ORG_SIG
M10606+ 40*(n-1)	Axis limit(+) on	AX1_POST_SIG
M10607+ 40*(n-1)	Axis limit(-) on	AX1_NEG_SIG
M10608+ 40*(n-1)	Z Count Signal	AX1_Z_SIG
M10609+ 40*(n-1)	Axis Synchronous main clutch ON Disable	AX1_SYNC_ON_DIS
M10610+ 40*(n-1)	Axis Synchronous main clutch OFF Disable	AX1_SYNC_OFF_DIS

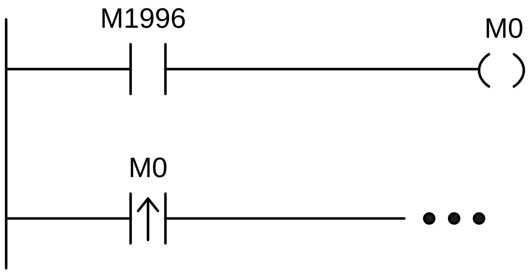
Relay No.	Function	System Tag Symbol
M10611+ 40*(n-1)	Axis Synchronous auxiliary clutch ON	AX1_SYNC_AUX_CLU_ON
M10612+ 40*(n-1)	Axis Synchronous auxiliary clutch ON Disable	AX1_SYNC_AUX_CLU_ON_BAN
M10613+ 40*(n-1)	Axis Synchronous auxiliary clutch OFF Disable	AX1_SYNC_AUX_CLU_OFF_BAN
M10614+ 40*(n-1)	Reserved	
M10615+ 40*(n-1)	Reserved	
M10616+ 40*(n-1)	Reserved	
M10617+ 40*(n-1)	Axis Probe 1 Function ON	AX1_PROBE1_ON
M10618+ 40*(n-1)	Axis Probe 1 Function Reset	RESET_AX1_PROBE1
M10619+ 40*(n-1)	Axis Probe 2 Function ON	AX1_PROBE2_ON
M10620+ 40*(n-1)	Axis Probe 2 Function Reset	RESET_AX1_PROBE2
M10621+ 40*(n-1)	Axis Synchronization Parameter Immediate Effect Request	AX1_SYNC_PARM_APPLY_IMMED
M10622+ 40*(n-1)	Axis Synchronization Parameter Next Period Effect Request	AX1_SYNC_PARM_APPLY_NXT_PER
M10623+ 40*(n-1)	Axis Syncronizationc Clutch Edge Trigger Buffer On	AX1_SYNC_CLU_EDGE_TRIG_CACHE_ON
M10624+ 40*(n-1)	Initialization of the Cam phase when the axis synchronous clutch is OFF	OUTPUT_PHASE_INIT_WHEN_AX1_SYNC_CLU_OFF
M10625+ 40*(n-1)	Axis Rotation Angle Choose Near	AX1_ROTA_ANG_CHOOSE_NEAR
M10626+ 40*(n-1)	Axis Rotation Angle Target Direction	AX1_ROTA_ANG_TGT_DIR
M10627+ 40*(n-1)	Axis Syncronizationc Mode ON	AX1_SYNC_MODE_ON

Relay No.	Function	System Tag Symbol
M10628+ 40*(n-1)	Pause Current Action	AX1_PAUSE_CURR_ACTN
M10629+ 40*(n-1)	Enable high-speed mode for axis origin search	AX1_HIGH_SPD_MODE
M10630+ 40*(n-1)	Set the current coordinates of the axis.	SET_AX1_COOR
M10631+ 40*(n-1)	Axis operation mode on	AX1_OPERATION_MODE
M10632+ 40*(n-1)	Axis operation mode unit	AX1_OPERATION_MODE_UNIT
M10633+ 40*(n-1)	The axis operation mode uses absolute positioning.	AX1_OPERATION_MODE_ABS_COOR
M11240+ 40*(n-1)	Servo On	AX1_SERVO_IS_ON
M11241+ 40*(n-1)	Operation Ready	AX1_OP_READY
M11242+ 40*(n-1)	Axis error in progress	AX1_IN_ERR
M11243+ 40*(n-1)	Axis warning in progress	AX1_IN_WARN
M11244+ 40*(n-1)	Control in progress	AX1_IN_CTRL
M11245+ 40*(n-1)	Homing in progress	AX1_IN_HOM
M11246+ 40*(n-1)	Homing done	AX1_HOM_DN
M11247+ 40*(n-1)	Positioning in progress	AX1_IN_POSI
M11248+ 40*(n-1)	Positioning done	AX1_POSI_DN
M11249+ 40*(n-1)	JOG in progress	AX1_IN_JOG
M11250+ 40*(n-1)	JOG done	AX1_JOG_DN
M11251+ 40*(n-1)	Synchronizing in progress	AX1_IN_SYNC
M11252+ 40*(n-1)	Synchronizing done	AX1_SYNC_ON

Relay No.	Function	System Tag Symbol
M11253+ 40*(n-1)	Speed mode in progress	AX1_SPEED_MODE
M11254+ 40*(n-1)	Speed mode done	AX1_SPEED_MODE_IS_DONE
M11255+ 40*(n-1)	Torque mode in progress	AX1_TORQ_MODE
M11256+ 40*(n-1)	Torque mode done	AX1_TORQ_MODE_IS_DONE
M11257+ 40*(n-1)	Axis soft limit(+) status	AX1_SOFT_LIM_POS_STATUS
M11258+ 40*(n-1)	Axis soft limit(-) status	AX1_SOFT_LIM_NEG_STATUS
M11259+ 40*(n-1)	Axis origin limit status	AX1_ORIG_LIM_STATUS
M11260+ 40*(n-1)	Axis limit(+) status	AX1_LIM_POS_STATUS
M11261+ 40*(n-1)	Axis limit(-) status	AX1_LIM_NEG_STATUS
M11262+ 40*(n-1)	Axis Probe 1 triggered state	TRIG_STATUS_OF_AX1_PROBE1
M11263+ 40*(n-1)	Axis Probe 2 triggered state	TRIG_STATUS_OF_AX1_PROBE2
M11264+ 40*(n-1)	Axis synchronization parameter effective state	VALID_STATUS_OF_AX1_SYNC_PARM
M11265+ 40*(n-1)	Axis tracking error state	AX1_FLO_ERR_STATUS
M11266+ 40*(n-1)	Axis Pause Status	AX1_PAUSE_STATUS

Motion special relay list

※All special relays do not provide TU and TD differential contact commands (TU、TD)，If it is necessary to perform differential action on the special relay, it can be replaced by an indirect method. (Refer to the picture below)



special relays use TD/TD by an indirect method

Note: Those marked with "◤" in special relays and temporary registers are forbidden to be written. Meanwhile, this kind of relays are still prohibited/disabling control and forced setting, and TU and TD contacts are not provided.

2-6 Motion Special Register Details

Relay No.	Function	System Tag Symbol
R36880	Motion controller state	UNIT_STATE
R36884 - 36903	Current Step	CURRENT_STEP_1 - CURRENT_STEP_20
R36904 - 36923	Current Block State	CURRENT_BLOCK_STATE_1 - CURRENT_BLOCK_STATE_20
R36924 - 36943	Flow State ID	FLOW_STATE_ID_1 - FLOW_STATE_ID_20
DR36964 - 36970	Encoder value	ENCODER_VALUE_2 - ENCODER_VALUE_4
DR36972	Gray code encoder value	GRAY_CODE_ENCODER_VALUE
DR36974	Gray code encoder turns	GRAY_CODE_ENCODER_TURNS

※ The table lists the register numbers for axis 1, and for each subsequent axis, the register number is obtained by adding 150*(n-1), 'n' represents the axis number.

Relay No.	Function	System Tag Symbol
R36980 + 150*(n-1)	Axis properties	-
R36984+ 150*(n-1)	Current Control Mode	AX1_CTRL_MODE
R37004+ 150*(n-1)	Error Detail Information 1	AX1_ERR_INFO_1
R37005+ 150*(n-1)	Error Detail Information 2	AX1_ERR_INFO_2
R37006+ 150*(n-1)	Warning Detail Information 1	AX1_WARN_INFO_1
R37007+ 150*(n-1)	Warning Detail Information 2	AX1_WARN_INFO_2
R37012+ 150*(n-1)	Axis Control	AX1_AX_CTRL
R37013+ 150*(n-1)	Axis Warning Code	AX1_WARN_CODE
DR37014+ 150*(n-1)	Command Coordinate	AX1_CMD_COORD
DR37016+ 150*(n-1)	Command Speed	AX1_CMD_SPD
DR37018+ 150*(n-1)	Command Position	AX1_CMD_POSI
R37020+ 150*(n-1)	Positioning Current Point No.	AX1_POSI_CUR_PT_NUM
DR37021+ 150*(n-1)	Current Coordinate	AX1_CUR_COORD
DR37023+ 150*(n-1)	Feedback Speed Monitor	AX1_SPD
DR37025+ 150*(n-1)	Position Deviation Monitor	AX1_POSI_DEV
DR37027+ 150*(n-1)	Digital Input from Driver	AX1_DRIVE_DI
R37029+ 150*(n-1)	Current Flow ID	CURRENT_AX_FLOW_NUM / AX2_FLOW_NUM
DR37030+ 150*(n-1)	Contact Output	AX1_CNTA_OUT
R37032+ 150*(n-1)	Current Torque	AX1_CUR_TORQ
DR37033+ 150*(n-1)	E-Cam Input Phase	AX1_ECAM_IN_PHASE

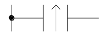
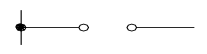
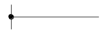
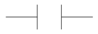
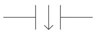
Relay No.	Function	System Tag Symbol
DR37035+ 150*(n-1)	Origin Position	AX1_ORG_POSI
R37037 - R37039+ 150*(n-1)	Axis Status Word 1-3	AX1_CONTROL_STATUS_WORD1 - AX1_CONTROL_STATUS_WORD3
DR37040+ 150*(n-1)	Main Clutch Output Phase	AX1_MAIN_CLUTCH_OUTPUT_PHASE
DR37042+ 150*(n-1)	Probe 1 Coordinate	AX1_DRIVER_PROBE1_COORDINATES
DR37044+ 150*(n-1)	Probe 2 Coordinate	AX1_DRIVER_PROBE2_COORDINATES

Table 1 Motion special register list

3

M SERIES PLC Instruction Lists

3-1 Sequential Instructions

Operand	Symbol	Function	Instruction Type
X,Y,M, S,T,C		Starting a network with a normally open (A) contact	Network starting instructions
		Starting a network with a normally closed (B) contact	
		Starting a network with a differential up (TU) contact	
		Starting a network with a differential down (TD) contact	
		Starting a network with a open circuit contact	
		Starting a network with a short circuit contact	
X,Y,M, S,T,C		Starting a relay circuit from origin or branch line with a normally open contact	Origin or branch line starting instructions
		Starting a relay circuit from origin or branch line with a normally closed contact	
		Starting a relay circuit from origin or branch line with a differential up contact	
		Starting a relay circuit from origin or branch line with a differential down contact	
		Starting a relay circuit from origin or branch line with a open circuit contact	
		Starting a relay circuit from origin or branch line with a short circuit contact	
X,Y,M, S,T,C		Serial connection of normally open contact	Serial connection instructions
		Serial connection of normally close contact	

Operand	Symbol	Function	Instruction Type
		Serial connection of differential up contact	
		Serial connection of differential down contact	
		Serial connection of open circuit contact	
		Serial connection of short circuit contact	
X,Y,M, S,T,C		Parallel connection of normally open contact	Parallel connection instructions
		Parallel connection of normally closed contact	
		Parallel connection of differential up contact	
		Parallel connection of differential down contact	
		Parallel connection of open circuit contact	
		Parallel connection of short circuit contact	
		Serial connection of two circuit blocks	Blocks merge instructions
		Parallel connection of two circuit blocks	
Y,M,S		Send result to coil	Coil output instruction
		Send inverted result to coil	
Y		Send result to an external output coil and appoint it as of retentive type	
TR		Save the node status to a temporary relay	Node operation instruction
		Load the temporary relay	
		Take the transition up of the node status	
		Take the transition down of the node status	
		Invert the node status	

Operand	Symbol	Function	Instruction Type
	—•—(S)	Set a coil	
	—•—(R)	Reset a coil	

sequential instructions list

※The 36 sequential instructions listed above are all applicable to every models of M - SERIES PLC.

3-2 Function Instruction

There are more than 100 different M SERIES PLC function instructions. If put the D and P derivative instructions into account, the total number of instructions is over 200. On top of these, many function instructions have multiple input controls (up to 4 inputs) which can have up to 8 different types of operation mode combinations. Hence, the size of M SERIES PLC instruction sets is in fact not smaller than that of a large PLC. Having powerful instruction functions, though may help for establishing the complicated control applications, but also may impose a heavy burden on those users of small type PLC's. For ease of use, M-Series PLC function instructions are divided into two groups, the Basic function group (The instructions attached with "★" symbol are basic functions which amounts to 26 function instructions and 4 SFC instructions) and the advanced function group.

- General Timer / Counter Function Instructions

FUN No	Name	Operand	Derivative Instruction	Function descriptions
★	T nnnn	PV		General timer instructions ("nnnn" range 0 ~ 1023, total 1024)
★	C nnnn	PV		General counter instructions ("nnnn" range 0 ~ 1279, total 1280)
★ 7	UDCTR	CV, PV	DP	16-Bit or 32-Bit up/down counter

General Timer / Counter Function Instructions list

- Single Operand Function Instructions

★ 4	DIFU	D	P	To get the up differentiation of a D relay and store the result to D
★ 5	DIFD	D	P	To get the down differentiation of a D relay and store the result to D
★10	TOGG	D	P	Toggle the ON/OFF status of the D relay

Single Operand Function Instructions List

● Setting / Resetting Instructions

★	SET	D	DP	Set all bits of register or a discrete point to 1
★	RST	D	DP	Clear all bits of register or a discrete point to 0
114	Z-WR	N	P	Zone set or clear

Setting / Resetting Instructions List

● SFC Instructions

★	STP	Snnnn		STEP declaration
★	STPEND			End of the STEP program
★	TO	Snnnn		STEP divergent instruction
★	FROM	Snnnn		STEP convergent instruction

SFC Instructions List

● Mathematical Operation Instructions

★11	(+)	Sa,Sb,D	D P	Perform addition of Sa and Sb and then store the result to D
★12	(-)	Sa,Sb,D	D P	Perform subtraction of Sa and Sb and then store the result to D
★13	(*)	Sa,Sb,D	D P	Perform multiplication of Sa and Sb and then store the result to D
★14	(/)	Sa,Sb,D	D P	Perform division of Sa and Sb and then store the result to D
★15	(+1)	D	D P	Adds 1 to the D value
★16	(–1)	D	D P	Subtracts 1 from the D value
24	SUM	S,N,D	D P	Take the sum of the successive N values beginning from S and store it in D
25	MEAN	S,N,D	D P	Take the mean average of the successive N values beginning from S and store it in D

27	NEG	D	D P	Take the 2's complement (negative number) of the D value and store it back in D
28	ABS	D	D P	Take the absolute value of D and store it back in D
38	PID2	ID,CH, SR,OR, PR,WR		PID operation
33	LCNV	Md,S,Ts,D,L	P	Linear Conversion
34	MLC	Rs,SI,Tx,Ty,TI, D	P	Multiple Linear Conversion
200	I→F	S,D	D P	Integer to floating point number conversion
201	F→I	S,D	D P	Floating point number to integer conversion
202	FADD	Sa,Sb,D	P	Addition of floating point number
203	FSUB	Sa,Sb,D	P	Subtraction of floating point number
204	FMUL	Sa,Sb,D	P	Multiplication of floating point number
205	FDIV	Sa,Sb,D	P	Division of floating point number
206	FCMP	Sa,Sb	P	Comparison of floating point number and then store the result to FO0 ~ FO2
207	FZCP	S,SU,SL	P	Comparison of floating point number S to the zones formed by the upper limit SU and the lower limit SL and then store the result to FO0 ~ FO2
209	FSIN	S,D	P	SIN trigonometric function
210	FCOS	S,D	P	COS trigonometric function
211	FTAN	S,D	P	TAN trigonometric function
212	FNEG	D	P	Change sign of floating point number
213	FABS	D	P	Take absolute value of floating point number

Mathematical Operation Instructions List

- Logical Operation Instructions

★18	AND	Sa,Sb,D	D P	Perform logical AND for Sa and Sb and store the result to D
★19	OR	Sa,Sb,D	D P	Perform logical OR for Sa and Sb and store the result to D
35	XOR	Sa,Sb,D	D P	Take the result of the Exclusive OR logical operation made between Sa and Sb, and store it in D
36	XNR	Sa,Sb,D	D P	Take the result of the Exclusive NOR logical operation made between Sa and Sb, and store it in D

Logical Operation Instructions List

- Comparison Instructions

★17	CMP	Sa,Sb	D P	Compare the data at Sa and data at Sb and store the result to FO0~FO2
37	ZNCMP	S,SU,SL	D P	Compare S with the zones formed by the upper limit SU and lower limit SL, and store the result to FO0~FO2

Comparison Instructions List

- In Line Comparison Instructions

170	=	Sa,Sb	D	Equal to compare
171	>	Sa,Sb	D	Greater than compare
172	<	Sa,Sb	D	Less than compare
173	< >	Sa,Sb	D	Not equal to compare
174	> =	Sa,Sb	D	Greater than or equal to compare
175	= <	Sa,Sb	D	Less than or equal to compare

In Line Comparison Instructions List

- Data Movement Instructions

★ 8	MOV	S,D	D P	Transfer data from S to D
★ 9	MOV/	S,D	D P	Invert data S, and then transfers the result to D
40	BITRD	S,N	D P	Read the status of the bits specified by N within S, and send it to FO0

41	BITWR	D,N	D P	Write the INB input status into the bits specified by N within D
42	BITMV	S,Ns,D,Nd	D P	Write the status of bit specified by Ns within S into the bit specified by Nd within D
43	NBMV	S,Ns,D,Nd	D P	Write the Ns nibble within S to the Nd nibble within D
44	BYMV	S,Ns,D,Nd	D P	Write the byte specified by Ns within S to the byte specified by Nd within D
45	XCHG	Da,Db	D P	Exchange the values of Da and Db
46	SWAP	D	P	Swap the high-byte and low-byte of D
115	DBUF	ID,CH,D		Store the data form Expansion Module to PLC register
160	RWFR	Sa,Sb,Pr,L,Sa		Read/write file register commands
161	WR-MP	S,Bk,Os,Pr,L,WR		Write memory pack
162	RD-MP	Bk,Os,Pr,L,D		Read memory pack

Data Movement Instructions List

- Shifting/Rotating Instructions

★ 6	BSHF	D	D P	Shift left or right 1 bit of D register
51	SHFL	D,N	D P	Shift left the D register N bits and move the last shifted out bits to FO0. The empty bits will be replaced by INB input bit
52	SHFR	D,N	D P	Shift right the D register N bits and move the last shifted out bits to FO0. The empty bits will be replaced by INB input bit
53	ROTL	D,N	D P	Rotate left the D operand N bits and move the last rotated out bits to FO0
54	ROTR	D,N	D P	Rotate right the D operand N bits and move the last rotated out bits to FO0

Shifting/Rotating Instructions List

- Code Conversion Instruction

61	→SEC	S,D	P	Convert the time data (hours, minutes, seconds) of the three successive registers starting from S into seconds data then store to D
62	→HMS	S,D	P	Convert the seconds data of S into time data (hours, minutes, seconds) and store the data in the three successive registers starting from D

Code Conversion Instruction List

- Flow Control Instructions

★ 0	MC	N		The start of master control loop
★ 1	MCE	N		The end of master control loop
★ 2	SKP	N		The start of skip loop
★ 3	SKPE	N		The end of skip loop
	END			End of Program
22	BREAK		P	Exit from FOR-NEXT loop
65	LBL	1 ~ 6 alphanumeri		Define the label with 1~6 alphanumeric characters
66	JMP	LBL	P	Jump to LBL label and continues the program execution
67	CALL	LBL	P	Call the sub-program begin with LBL label
68	RTS			Return to the calling main program from sub-program
69	RTI			Return to interrupted main program from sub-program
70	FOR	N		Define the starting point of the FOR Loop and the loop count N
71	NEXT			Define the end of FOR loop
199	TXTDF	LN		Ladder Program blocking function

Flow Control Instructions List

- I/O Function Instructions

74	IMDIO	D,N	P	Update the I/O signal on the main unit immediately
99	TPCTL 2	ID,CH,SR,PR, OR,WR		PID control Instructions

I/O Instructions List

- Cumulative Timer Function Instructions

87	T1mS	CV,PV		Cumulative timer using 1mS as the time base
88	T10mS	CV,PV		Cumulative timer using 10mS as the time base
89	T100mS	CV,PV		Cumulative timer using 100mS as the time base

Cumulative Timer Function Instructions List

- Watch Dog Timer Control Function Instructions

90	WDT	N	P	Set the WDT timer time out time to N mS
91	RSWDT		P	Reset the WDT timer to 0

Watch Dog Timer Control Function Instructions List

- High Speed Counter Control Function Instructions

92	HSCTR	CN	DP	Read the current CV value of the hardware HSCs, HSC0 ~ HSC3, or HST on SOC to the corresponding CV register in the PLC respectively
93	HSCTW	S,CN,D	DP	Write the CV or PV register of HSC0 ~ HSC3 or HST in the PLC to CV or PV register of the hardware HSC or HST on SOC respectively

High Speed Counter Control Function Instructions List

- Ramp Up/Down Function Instructions

98	RAMP2	Om,Ta Td,Rt Rc,WR		Tracking type ramp function for analog output

Ramp Function Instructions List

- Communication Function Instructions

150	M-Bus	Pt,SR,WR	P	Modbus protocol communication instruction
151	CLINK	Pt,MD,SR,WR	P	FATEK/Generic protocol communication instruction
152	NCR			Active network communication
156	CMCTL	ID,Pt,Ts,MD,WR		Communication module instruction

Communication Function Instructions List

- Table Function Instructions

103	BT_M	Ts,Td,L	D P	Copy the entire contents of Ts to Td
107	T_FIL	Rs,Td,L	D P	Fill the table Td with Rs
113	SORT	S,D,L	D P	Sorting the registers starting from S length L and store the sorted result to D

Table Function Instructions List

- Matrix Instructions

130	MBCNT	Ms,L,D	P	Calculate the total number of bits that are 0 or 1 in Ms, then store the results into D

Matrix Instructions List

- NC Positioning Instruction

140	HSPSO	Ps,SR,WR		HSPSO instruction of NC positioning control
141	MPARA	Ps,SR		Parameter setting instruction of NC positioning control

NC Positioning Instruction List

- Interrupt Control Instruction

145	EN	LBL	P	Enable HSC, HST, external INT or peripheral operation
146	DIS	LBL	P	Disable HSC, HST, external INT or peripheral operation

Interrupt Control Instruction List

- Motion Control Instruction

176	MFFlowStart	ID		Start the motion flow
177	MFSysStop			Control motion system stop
178	MFHome	AX		Control the axis homing
179	MFPointMov	PT		Start point position control
180	MFJog	AX,D,MD		Control the axis homing
182	MFFlowPause			Pause the motion flow
183	MFFlowResume			Resume the motion flow
184	MFFlowHalt			Suspend the motion flow
185	MFSysRstAlm			Reset motion alarm status
186	MFFlowStop	ID		Terminate the motion flow
187	MFSysInit			Servo initialization

Motion Control Instruction list

4

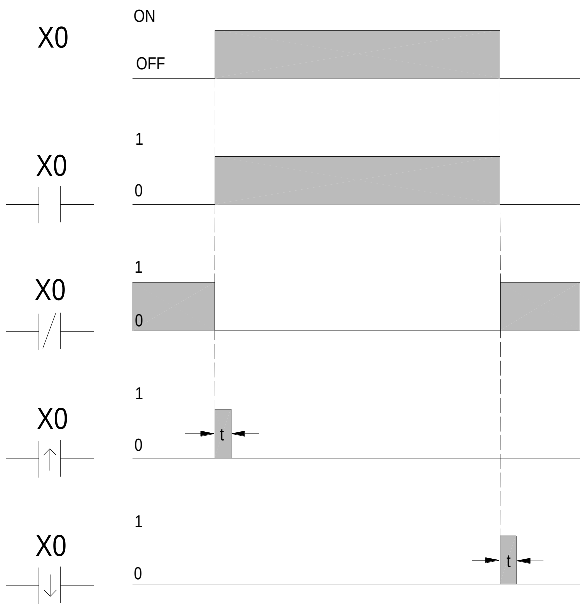
Sequential Instructions

This chapter only describes the Element features and functions of sequence commands.

4-1 Element Description

4-1.1 Characteristics of A,B,TU and TD Contacts

- Input X0 from the input terminal block
- A contact Element status
- B contact Element status
- TU contact Element status
- TD contact Element status

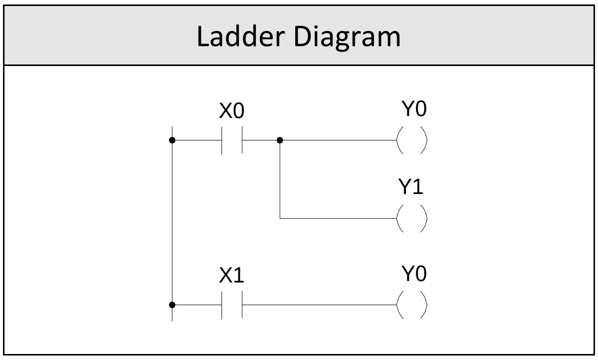


Characteristics of A,B,TU and TD Contacts

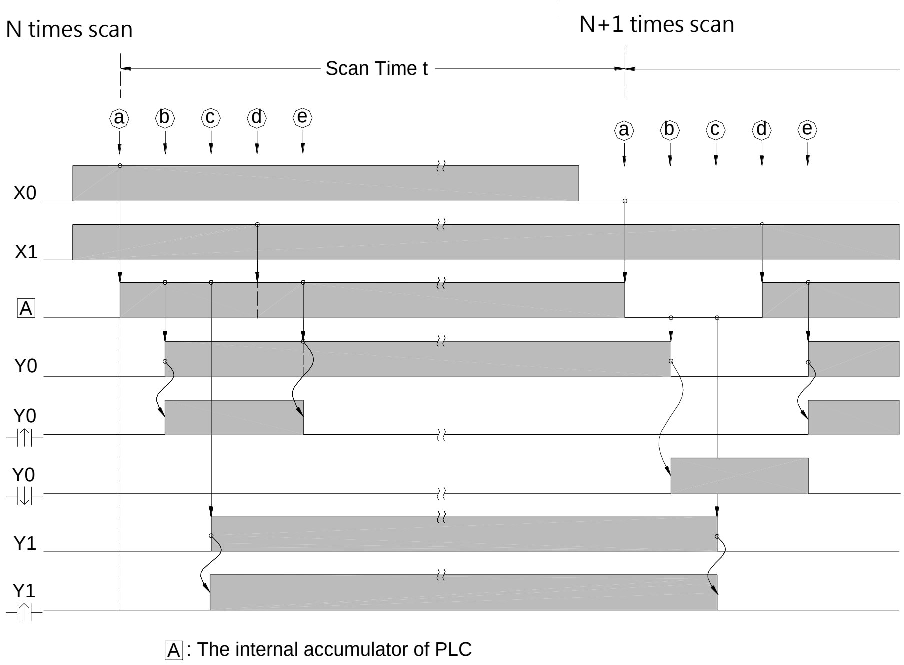
The waveform shown above reveals the function of A, B, TU and TD elements by exercising the external input X0 form OFF to ON then OFF.

- TU (Transition Up): This is the “Transition Up Contact”. Only a rising edge (0→1) of the referenced signal will turn on this element for one scan time.
- TD (Transition Down): This is the “Transition Down Contact”. Only a falling edge (1→0) of the referenced signal will turn on this element for one scan time.
- TU and TD contact will automatically generate the TU or TD pulse corresponding to the contacts or coils for all X, Y, M, S, T, C contact or coil state changes. However, if the state change of the coil is operated by the "application Instruction" in units of 16 or 32 bits（WY△△△△，WM△△△△△，WS△△△△）, TU or TD pulses will not be generated.

Note: The "ON" maintenance time of the TU and TD elements of the M SERIES PLC relay is the first scan after the "ON" condition of the element is established (for example, the TU element changes from 0 to 1, and the TD element changes from 1 to 0). Set it to "ON" for coil elements. Once it is set to "ON", it will be cleared to "OFF" immediately when it is scanned again. In most applications, each element will only be scanned once during the CPU problem-solving scan cycle, so the "ON" time of TU and TD elements must be equal to the scan time of the CPU. However, if it is scanned more than once in a CPU scan cycle (such as using "immediate input" or "multiple coil output" in the program), the TU and TD states of its elements will be the first time the "ON" condition is met. Set to "ON" when the scan arrives, and clear to "OFF" immediately when the second scan arrives, and the "ON" time will be less than one CPU scan time. The TU of Y0 in the following illustration is that. Therefore, if the customer needs to capture the TU of Y0 for trigger operation, one must insert the application program in the range of Y0 TU "ON" to "OFF" (in this example, between b and e), otherwise he will not be able to capture any Y0 or TU trigger signal.



Example diagram of the contact and scan time relationship

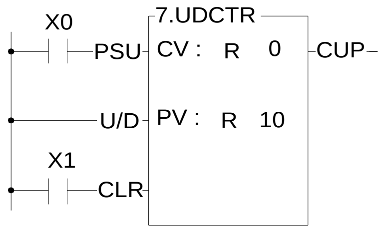


PLC contact trigger and scan time relationship

- Besides the TU/TD instructions which can detect the status change of reference operand, M SERIES PLC also provides the instructions to detect the change of node status (power flow). For details please refer the descriptions of FUN4 (DIFU) and FUN5 (DIFD) instructions.

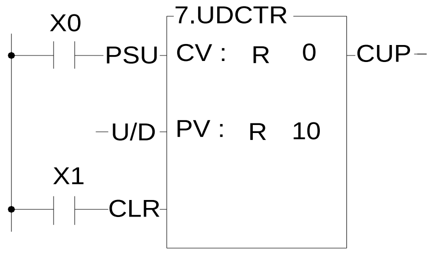
4-1-2 OPEN and SHORT Contact

The status of OPEN and SHORT contacts are fixed and can't be changed by any ladder instructions. Those two contacts are mainly used in the places of the Ladder Diagram where fixed contact statuses are required, such as the place where the input of an application instruction is used to select the mode. The sample program shown below gives an example of configuring an Up/Down counter (UDCTR) to an Up counter by using the SHORT contact.



Up counter using the SHORT contact

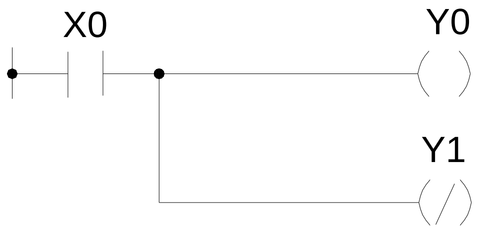
- FUN7 is the UDCTR function. While rising edge of CK input occur, FUN7 will count up if the U/D status is 1 or count down if the U/D status is 0. The example shown above, U/D status is fixed at 1 since U/D is directly connected from the origin-line to a SHORT contact, therefore FUN7 becomes an Up counter. On the contrary, if the U/D input of FUN7 is connected with an OPEN contact from the origin-line, the FUN7 becomes a DOWN counter.



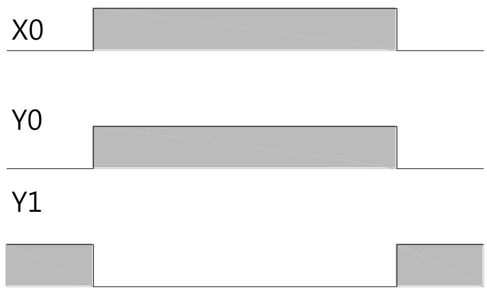
Down counter using the OPEN contact

4-1-3 Output Coil and Inverse Output Coil

Output Coil writes the node status into an operand specified by the coil instruction. Invert Output Coil writes the complement status of node status into an operand specified by the coil instruction. The characteristics depicts at below.



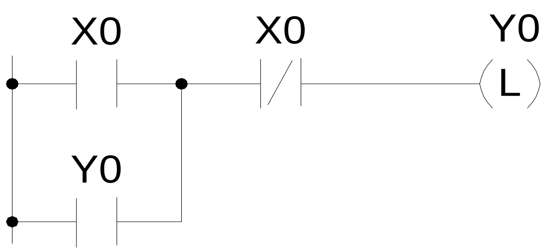
Ladder Diagram of Output Coil and Invert Output Coil



Output Coil and Invert Output Coil

4-1-4 Retentive Output Coil

For the internal coil, it can be set as holding or non-holding (it is a dichotomy, such as M0-M8519 of the internal coil M0-M9119 is non-holding, then M8520-M9119 is holding), but for the output point, due to practical It is not suitable to use the dichotomy method to set hold or non-hold, so if most PLCs need to hold the output point, they must first send the result to the internal hold coil, and then send the internal hold coil to the indirect method of the output point, M SERIES PLC Then provide you with the method of selecting the output point to be maintained under the page of I/O Configuration -> Output Power Failure Hold, the following self-protection circuit:

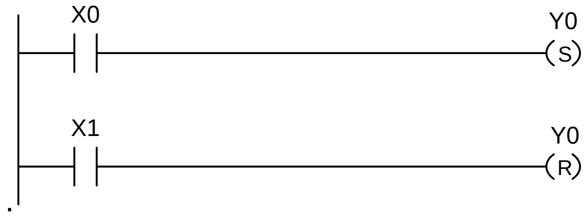


Ladder Diagram of Retentive Output Coil

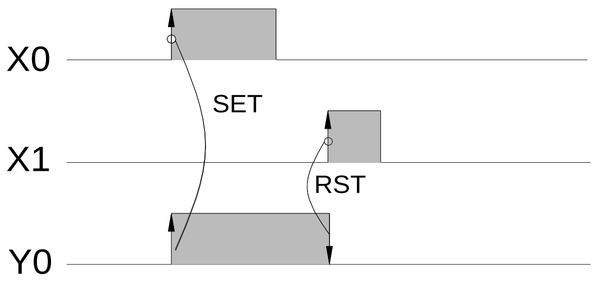
From the above example, if turn the X0 "ON" then "OFF", Y0 will keep at "ON". When change the PLC state from RUN to STOP then RUN or turn the power off then on, the Y0 still keep at ON state. But if use the OUT Y0 instruction instead of the OUT L Y0, need to turn the X0 "ON" again after change the PLC state from RUN to STOP then RUN or turn the power off then on, Y0 status will be ON.

4-1-4 Set Coil and Reset Coil

Set Coil writes 1 into an operand specified. Reset Coil writes 0 into an operand specified. The characteristics depicts at below.



Ladder Diagram of Set Coil and Reset Coil



Set Coil and Reset Coil

5

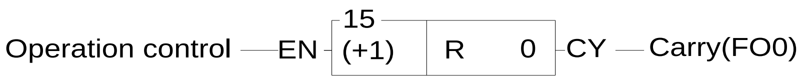
Description of Function Instructions

5-1 The Format of Function Instructions

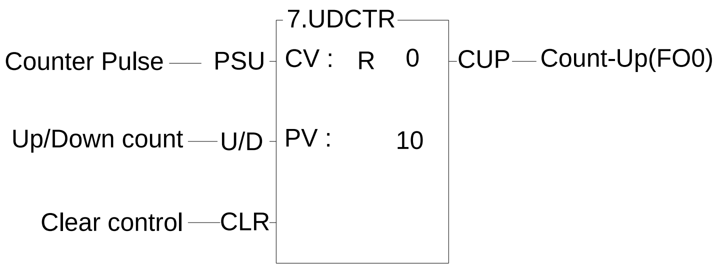
Function Instructions of M Series PLC will be divided into four parts including input control, instruction number/name, operand and function output. The number of input controls, operands, and function outputs of each instruction is different (please refer to the description of each instruction).

Ladder Diagram

Example 1: Single Input Instruction



Example 2: Multiple Input Instruction



The Format of Function Instructions

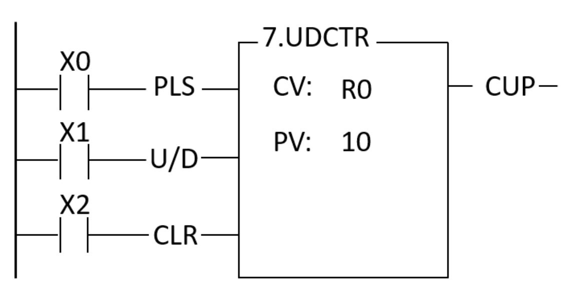
5-1-1 Input Control

M SERIES PLC has at least one input control for other application commands except for 7 application commands without input control, up to four. Application instructions are based on the combination of input control signals to determine whether to execute the instruction and what kind of operation to perform. On the software package of UperLogic and when the ladder diagram program is printed out, all the input control and function output terminals of the application instruction symbols are marked with English comment abbreviations to indicate what kind of function control or output the terminal is, so as to facilitate memory and Read, as shown in the above figure 2, the first input is marked "PLS", which means that the counter only counts once when the counting pulse pulse changes from 0→1 (rising edge), and the second input is marked "U/D" on the U meter above the slash Count Up, D at the bottom means count Down, if this input is 1, when the counting pulse PLS comes, the counter value will increase by 1, otherwise, if it is 0, it will decrease by 1, and the third input is marked "CLR", which means clear Clear, that is, when this input is 1, the count value of the counter will be cleared to 0. For the input control comments of other application commands, please refer to the description of each command.

Note: No input control command means that the command needs to be directly connected to the bus, and cannot be connected in series with input control components, and has no functional output. The command itself forms a network. There are 6 non-input control commands such as MCE, SKPE, LBL, RTS, FOR, NEXT, etc., please refer to the description of each command in Chapter 6 and 7.

All input controls of the function instructions should be connected by the corresponding elements, otherwise a syntax error will occur. As shown in example 3 below, the function instruction FUN7 has three inputs and three elements before FUN7. X0, X1, and X2 corresponds to the first input PLS, second input U/D and third input CLR.

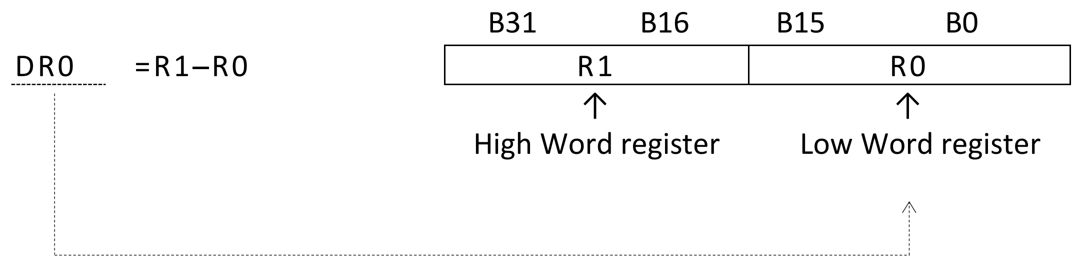
Ladder Diagram

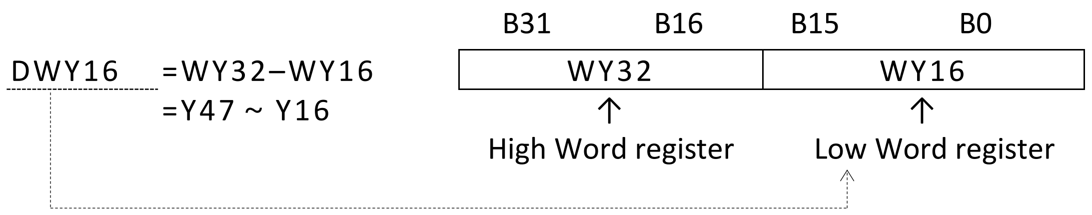


Function instruction of Input Control

5-1-2 Instruction Number and Derivative Instructions

D : Indicates a Double Word (32-bit). The 16-bit word is the basic unit of the registers in M-Series PLC. The data length of R, T and C (except C1024~C1063) registers are 16-bit. If a register with 32-bit data length is required, then it is necessary to combine two consecutive 16-bit registers together such as R1-R0, R3-R2 etc. and those registers are represented by prefix a D letter before register name such as DR0 represents R1-R0 and DR2 represents R3-R2. If you enter DR0 or DWY16 in the monitor mode of FP-08, then a 32-bit long value (R1-R0 or WY32-WY16) will be displayed.



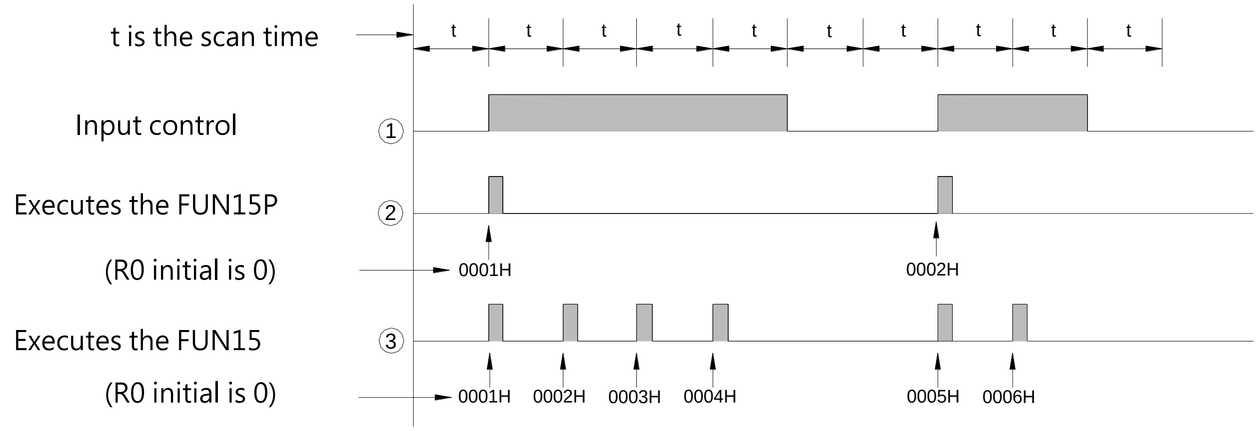


Note 1: In order to differentiate between 16-bit and 32-bit instructions while using the ladder diagram and mnemonic code, we add the postfix letter D after the "Instruction number" to represent 32-bit instructions and the size of their operand are 32-bit. The instruction FUN 11D has a postfix letter D, therefore the source and destination operands need to prefix a letter D as well, such as the augend Sa:R0 is actually Sa=DR0=R1-R0 and Sb=DR2=R3-R2. Please also pay special attention to the length of the other operands except source and destination are only one word whether 16-bit or 32-bit instructions are used. Note 2: Reading register status at labber Diagram, we can add the prefix letter W before the "Instruction number" to access the register status of consecutive 16 bits, for example, WX0 represent X0~X15, WM32 represent M32~M47, the accessed bit address must be a multiple of 16 bits. For example, WM16 is a legal address, but WM8 is an illegal address.

P : indicates the pulse mode instruction. The instruction will be executed when the status of input control changes from 0 to1 (rising edge). If a postfix letter P is added to the instruction (FUN 15P), the instruction FUN 15P will only be executed when the status of input control signal changes from 0 to 1. The execution of the instruction is in level mode if it does not have a P postfix, this means the instruction will be executed for every scan until the status of input control changes from 1 to 0. In this operation manual, an example of the operation statement of a function instruction is shown below.

● When the operation control “EN”=1 or (P instruction) from 0→1, · · · · · · · · · ·

The first one indicates the execution requirement for non-P instruction (level mode) and the second one indicates the execution requirement for P instruction (pulse mode). The following waveform shows the result (R0) of FUN15 and FUN15P under the same input condition.



FUN15 R0 of FUN15 and FUN15P under the same input condition

D P : Indicates the instruction is a 32-bit instruction operating with pulse mode.

Note: P instruction is much more time saving than level instruction in program scanning, so user should use P instruction as much as possible.

5-1-3 Operand

The operand is used for data reference and storage. The data of source (S) operand are only for reference and will not be changed by the execution of the instruction. The destination (D) operand is used to store the result of the operation and its data may be changed after the execution of the instruction. The following table illustrates the names and functions of M-Series PLC function instruction's operands and types of contacts, coils, or registers that can be used as an operand.

- The names and functions of the major operands：

Abbreviation	Name	Description
S	Source	The data of source (S) operand are only for reading and reference and will not be changed with the execution of the instruction. If there are more than one source operands, each operand will be identified by the footnote such as Sa and Sb.
D	Destination	The destination (D) operand is used to store the result of operation. The original data will be changed after operation. Only the coils and registers which are not write prohibited can be the destination operand.
L	Length	Indicates the data size or the length of the table, usually are constants.
N	Number	A constant most often used as numbers and times. If there are more than one constant, each constant will be identified by the footnotes such as Na, Nb, Ns, Nd, etc..
Pr	Point	Used to point to a specific a block of data or a specific data or register in a table. Generally, the Pr value can be varied, therefore cannot be constant or input register.
CV	Current value	Used in T and C instruction to store the current value of T or C
PV	Set value	Used in T and C instructions for reference and comparison
T	Table	A combination of a set of consecutive registers forms a table. The basic operation units are word and double word. If there is more than one table, each table will be identified by footnotes such as Ta, Tb, Ts and Td etc..

Abbreviation	Name	Description
M	Matrix	A combination of a set of consecutive registers forms a matrix. The basic operation unit is bit. If there is more than one matrix, each matrix will be identified by footnotes such as Ma, Mb, Ms and Md etc..

Major operands list

Besides the major operands mentioned above, there are other operands which are used for certain special purposes such as the operand Fr for frequency, ST for stack, QU for Queue etc., please refer to the instruction descriptions for more details.

- The types of the operand and their range: The types of operands for the function instructions are Discrete, Register and Constant.

a. Discrete (Digital) Operand:

There is total five function instructions that reference the discrete operand, namely SET, RST, DIFU, DIFD and TOGG. Those five instructions can only be used for operations of Y△△△△ (external output), M△△△△△ (internal and special) and S△△△△ (step) relays. The table shown below indicates the operands and ranges of the five function instructions.

Range / Operand	Y	M	SM	S
	Y0 ~ Y255	M0 ~ M1911	M1912 ~ M2001	S0 ~ S999
D	○	○	○	○

Discrete operand ranges list

Symbol "O" indicates the D (Destination operand) can use this type of coils as operands. The "*" sign above the "O" shown in SM column indicates that should exclude the write prohibited relays as operands. Please refer to Chapter 2-3 for introduction of the special relays

b. Register Operand:

The major operand for function instructions is register operand. There are two types of register operands: the native registers which already is of Words or Double Words data such as R, T, C. The other is derivative registers (WX, WY, WM, WS) which are formed by discrete bits. The types of registers that can be used as instruction operands and their ranges are all listed in the following table:

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ~ WX1008	WY0 ~ WY1008	WM0 ~ WM29584	WS0 ~ WS3088	T0 ~ T1023	C0 ~ C1279	R0 ~ R34767	R34768 ~ R34895	R35024 ~ R35151	R35280 ~ R43223	R43224 ~ R47319	D0 ~ D11999	16/32-bit +/- number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○*	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○
⋮														

Register operand ranges list

The "○" symbol in the table indicates can apply this kind of data as operand. The "○*" symbol indicates can apply this kind of data except the write prohibited registers as operand. To learn more about write prohibited registers please refer to page 2.4 for introduction of the special register. When R43224 ~ R47319 are not set to be read only registers, can used as normal registers (read, and write)

Note 1: The registers with a prefix W, such as WX, WY, WM and WS are formed by 16 bits. For example, WX0 means the register is formed by X0(bit 0) ~ X15(bit 15). WY144 means the register is formed by Y144(bit 0) ~ Y159(bit 15). Please note that the discrete number must be the multiple of 16 such as 0, 16, 32, 48....
Note 2: The last register (Word) in a table can not be represented as a 32-bit operand in the function because 2 Words are required for a 32-bit operand. The use of WM, WX, WY must be a multiple of 16, for example: WM16, WM32 are supported; WM8 is not supported.
Note 3: TMR (T0~T1023) and CTR (C0~C1279) are special temporary registers for timers and counters. Although they can also be used as general temporary registers, they will make the system complex and difficult to debug. Therefore, except for T or C commands, other instructions should avoid writing to TMR or CTR.
Note 4: T0 ~ T1023 and C0 ~ C1023 are 16-bit register. But C1024~C1279 are 32-bit register, therefore can’t be used as 16-bit operands.
Note 5: Apart from being directly appointed by register’s number (address) as the foregoing discussions, RXXXXX and DXXXXX register can be combined with pointer register V 、Z or P0~P9 to make indirect addressing. Please refer to the example in the next section (Section 5.3) for the description of using pointer register (XR) to make indirect addressing.

c. Constant Operand:

The range of 16-bit constant is between -32768~32767. The range of 32-bit constant is between -2147483648~2147483647. The constant for several function instructions can only be a positive constant. The range of 16-bit and 32-bit constants are listed in the table shown below.

Classification	Range
16-bit signed number	-32768 ~ 32767
16-bit un-signed number	0 ~ 32767
32-bit signed number	-2147483648 ~ 2147483647

32-bit un-signed number	0 ~ 2147483647
16/32-bit signed number	-32768 ~ 32767 or -2147483648 ~ 2147483647
16/32-bit un-signed number	0 ~ 32767 or 0 ~ 2147483647

Constant category and its range table

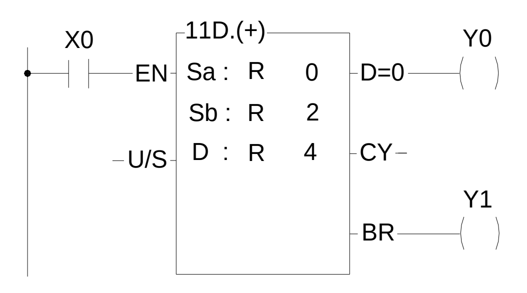
In addition, some specific operands have different lengths (such as length L, number of bits...N, etc.) and the range will be directly marked on the field of each operand. Please refer to the description of individual instructions.

5-1-4 Functions Output (FO)

The “Function Output” (FO) is used to indicate the operation result of the function instruction. Like control input, each function outputs shown in the screen of programming software are all attached with a word which comes from the abbreviation of the output functionality. Such as CY derived from CarrY. The maximum number of function outputs is 4 and those are denoted as FO0, FO1, FO2, FO3 respectively. The order is from top to bottom, first FO is FO0, second FO is FO1, last FO is FO3. The FO status must be taken out by FO instruction. The unused FO may be left without connecting to any elements, such as FO1 (CY) shown in Example 4 below.

Example 4:

Ladder Diagram



Function output diagram using FUN11

5-2 Use W Prefix for Word and Bit Access Transformation

The single-point (BIT state) memory of M-Series PLC can use W prefix word for word access, that is to access 16 single points at a time, for example, WX0 means one access to X0~X15 On the contrary, you can also use this technique to access any single-point state of the character group data, for example, you can place the character group data in WM0, if you want to read the 6th bit of the character group state, just read M6 directly.

5-3 Numbering System

5-3-1 Binary values and the terms

Binary is the basic number system of digital computers. PLC is composed of digital computers, and naturally uses binary. In order to express and grasp binary values, you first need to understand the following terms:

- Bit: (Bit is abbreviated as B, such as B0, B1...) Bit is the most basic unit of binary value, and its state is either 1 or 0.
- Nibble: (Nibble is abbreviated as NB, such as NB0, NB1...) It is composed of 4 consecutive bits (such as B3~B0) and can be used to represent a decimal number 0~9 or 0~F in hexadecimal.
- Byte: (Byte abbreviated as BY, such as BY0, BY1...) is composed of two consecutive digits (that is, 8 bits, such as B7~B0). It can represent the two-digit value of hexadecimal 00~FF.
- Word group: (Word abbreviation W, such as W0, W1...) is composed of two consecutive bits (that is, 16 bits such as B15~B0) can represent 16 The 4-digit value in base system is 0000~FFFF.
- Double word group: (Double word abbreviation DW, such as DW0, DW1...) is composed of two consecutive word bytes (that is, 32 bits, such as B31~B0) can represent the 8-digit value of hexadecimal 00000000~FFFFFFFF.

DW ←Double word group

W1 W0 ←Word group

BY3 BY2 BY1 BY0 ←Byte

NB7 NB6 NB5 NB4 NB3 NB2 NB1 NB0 ←Nibble

B31	B30	B29	B28	B27	B26	B25	B24	B23	B22	B21	B20	B19	B18	B17	B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0

←Bit

Schematic diagram of binary values and the terminology

Floating Point Number: It is composed of two consecutive word bytes. The maximum range that can be represented by floating point numbers is ±(1.8*10-38 ~ 3.4*1038), please refer to Section 5.3.6 for the detailed format description.

5-3-2 M SERIES PLC Digit

The numerical calculation or storage inside the M SERIES PLC all uses binary values (Binary), so the values input from the outside to the PLC must be converted into binary codes before the PLC can process them. Similarly, the numerical results retrieved from the PLC are also binary values so all the numbers of UpperLogic must be converted into binary before they can be input to PLC. However, because binary values are extremely difficult to input and read, UpperLogic provides users with the familiar decimal or hexadecimal to input or display in the man-machine interface (numerical input or display), But in fact, all numerical processing is carried out in binary code. 。

Note: If your numerical input or display is not through UperLogic (for example, use a dip switch or a 7-segment display to input to or get from the PLC through the I/O point), then you have to use the ladder diagram program instructions to process the binary and the conversion between decimals allows you to input in decimals and get output in decimals without using UperLogic. Please refer to the description of FUN20(BIN→BCD) and FUN21(BCD→BIN).

5-3-3 Value Range

As mentioned above, all M SERIES PLCs use binary internally (the BCD value is only for people's habit, and the binary value is converted into a digital display suitable for people to read). There are three types of values in PLC: 16-bit, 32-bit and floating-point numbers, which can represent the following ranges respectively.

16-bit	-32768 ~ 32767
32-bit	-2147483648 ~ 2147483647
Floating Point Number	$\pm(1.8*10^{-38} \sim 3.4*10^{38})$

5-3-4 Display of Values (Please skip this section for beginners)

The following sections describe the representation and format of 16-bit and 32-bit values. For users to have an in-depth understanding of the calculation process and results of numerical values and to meet various complex application requirements.

Whether it is a 16-bit or 32-bit value, its highest bit MSB (16-bit B15, 32-bit B31) indicates the sign of the value (0: positive number, 1: negative number), and the rest Bits (B14~B0 or B30~B0) are really used to represent the numerical value, hereby take 16 bits as an example to explain as follows: (32 bits are also done in the same way, only the length is doubled).

0 : Positive Number

	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	
12345 (Decimal value) ⇨	0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	1 ⇨ 3039H (Hex value)

B15 B0

8192+4096+32+16+8+1=12345

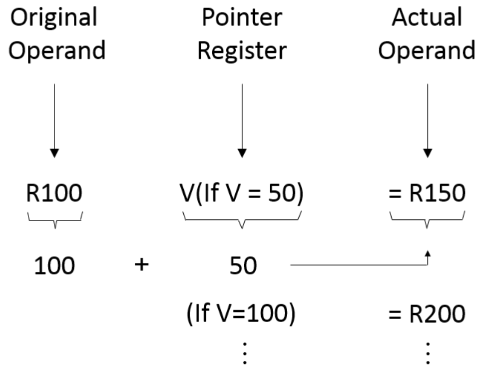
Schematic diagram of displayed values

As in the above example, regardless of 16-bit or 32-bit, the binary bits start from the lowest bit LSB (B0), B0 represents 1, B1 represents 2, B2 represents 4, B3 represents 8,…the rest can be deduced by analogy, and its value is the sum of the values represented by all the bits that are 1.

5-4 Use Index Register (XR) for Indirect Addressing

In the M-Series PLC function instructions, there are some operands that can be combined with pointer register (V, Z, P0~P9) to make indirect addressing (will be shown in the operand table if it applicable). Registers in the range RXXXXX can be combined with a pointer register to perform indirect addressing useing operand (V, Z), range RXXXXX can be combined with an pointer register to perform indirect addressing useing operand(P0~P9). Other operands such as discrete and constant cannot be used for indirect addressing).

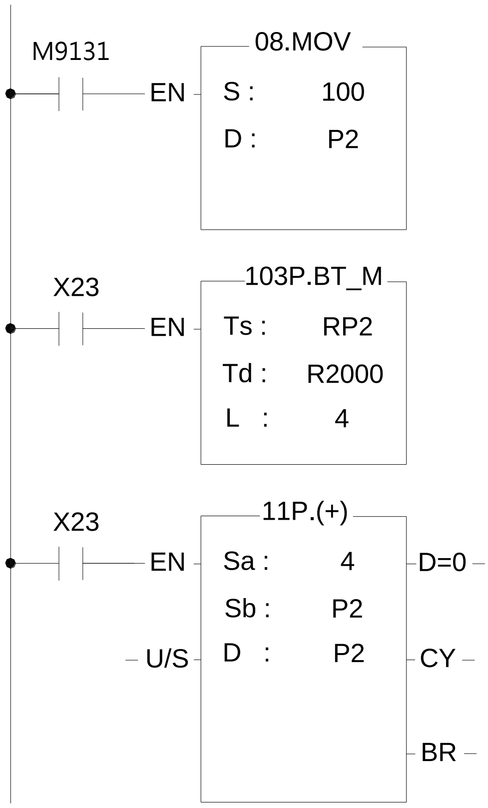
There are twelve pointer registers XR (V, Z, P0~P9). The V register in the M SERIES PLC is R43214, and the Z register is R43216. The actual addressed register by index addressing is just offset the original operand with the content of the index register.



As shown in the above diagram, you only need to change the V value to change the operand address. After combining the index addressing with the M-Series PLC function instructions, a powerful and highly efficient control application can be achieved by using very simple instructions. Using the program shown in the diagram below as an example, you only need to use a block move instruction (BT_M) to achieve a dynamic block data display, such as a parking management system.

5-4-1 Index Register (P0~P9) Introduction

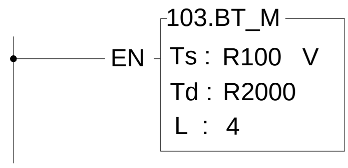
- In indirect addressing application, Rxxxxx register can combine V, Z , P0~P9 for index addressing; Dxxxxx register can't combine V, Z for index addressing, but P0~P9 are allowed.
- When V, Z index register being combined with the Rxxxxx register, for example, R0 with V, Z, the instruction format is R0V (where V=100, it means R100) or R0Z(where Z=500, it means R500); when P0~P9 index register being combined with the Rxxxxx register, the instruction format is RPn (n=0~9) or RPmPn (m,n=0~9), for example RP5 (where P5=100, it means R100) or RP0P1(where P0= 100, P1=50, it means150).
- When P0~P9 index register being combined with the Dxxxxx register, the instruction format is DPn (n=0~9) or DPmPn (m, n=0~9), for example DP3 (where P3=10, it means D10) or DP4P5 (where P4=100, P5=1, it means D101).
- It can combine both P0~P9 index register, for example P2=20, P3=30, when Rxxxxx or Dxxxxx register combines both index register, RP2P3 will point to R50, DP2P3 will point to D50, it means the summation of both indexes register for indirect addressing.



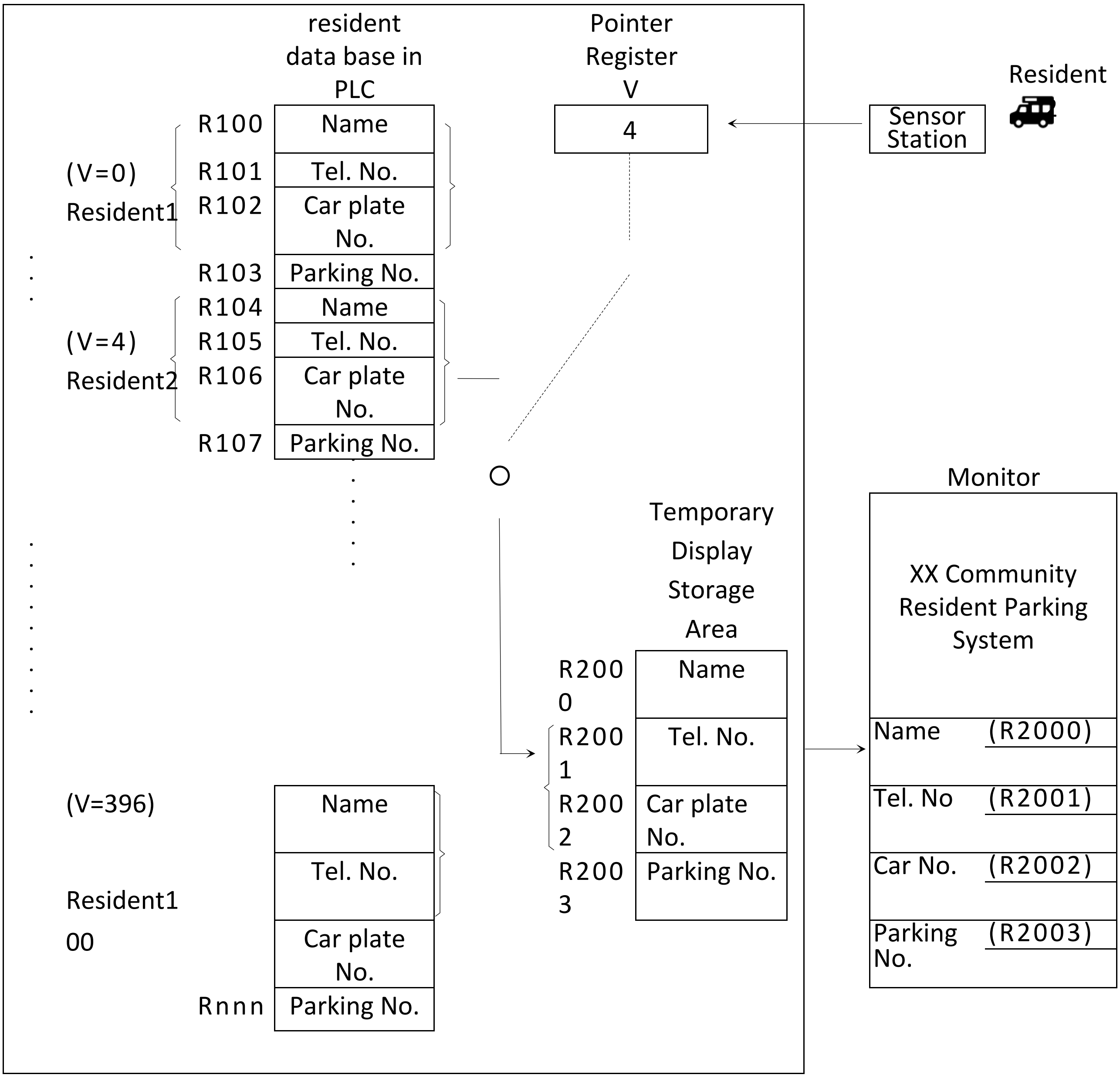
1. Power up and the initial pulse M9131 will move 100 into the index register P2.
2. When X23 changes from 0 1, FUN103 will perform the table movement, the source starts from R100 (P2=100), the destination starts from R2000, the amount is 4.
   Coping the content of R100~R103 for R2000~R2003 at first execution, coping the content of R104~R107 for R2000~R2003 at second execution…
3. Fun11 is used to increase the index by 4 words each time, every time X23 is "ON", P2 index register will be increased by 4.

5-4-2 Indirect Addressing Program Example

Ladder Diagram



Ladder Diagram of FUN103 BT_M



Automated Parkinglot Management System

Program Description

The above example assumes that the automated parking lot management system for residents in a community has a total of 100 resident parking spaces, and each resident has 1 set of basic information, which are resident name, phone number, car plate number, parking number, etc. As shown in the figure above, it occupies 4 consecutive PLC internal temporary registers, occupying a total of 400 temporary registers such as R100~R499. Each household has a card with a different card number, which is used for entrance and exit control and parking lot. 0, 4, ……, 396, etc. 100 types, after the PLC senses the card number, it will be stored in the index temporary register "V", and displayed on the terminal (LCD or CRT) at the administrator's office The data is captured and displayed by R2001~R2003 inside the PLC. For example, in this example, the card of resident 2 is sensed, and its value=4, so the V register=4, and the PLC immediately moves the data of R104~R107 to the display Temporary storage area (R2000~R2003), so the terminal at the administrator can immediately display the information on the terminal when it senses the card of resident 2.

⚠ Warning
1. Although using pointer register for indirect addressing application is powerful and flexible, but changing the V and Z values freely and carelessly may cause great damages with erroneous writing to the normal data areas. The user should take special caution during operation. 2. In the data register range that can be used for indirect addressing application (RXXXXX,DXXXXX), the 12552 registers R34768~R47319 (i.e. IR, OR and SR) are important registers reserved for system or I/O usage. Writing at-will to these registers may cause system or I/O errors and may result in a major disaster. Due to the fact that users may not easily detect or control the flexible register address changes made by the V and Z values, M-Series PLC will automatically check if the destination address is in the R34768~R47319 range. In case it is necessary to write to the registers R34768~R47319, please use the direct addressing.

5-4-3 Representation of Negative Number (Beginners should skip this section)

As prior discussion, when the MSB is 1, the number will be a negative number. The M-Series PLC negative numbers are represented by 2'S Complement, i.e to invert all the bits (B15 ~ B0 or B31 ~ B0) of its equivalent positive number (The so-called 1'S Complement is to change the bits equal 1 to 0 and the bits equal 0 to 1) then add 1. In the above example, the positive number is 12345. The calculation of its 2'S Complement (i.e. –12345) is described below:

12345	⇨	0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	1	⇨	3039H
1'S Complement of 12345	⇨	1	1	0	0	1	1	1	1	1	1	0	0	0	1	1	0	⇨	CFC6H
+																	1		
2'S Complement of 12345 (-12345)	⇨	1	1	0	0	1	1	1	1	1	1	0	0	0	1	1	1	⇨	CFC7H

Example of Negative Number

5-4-4 Representation of Floating Point Number (Beginners should skip this section)

The format of floating point number of FATEK-PLC follows the IEEE-754 standard, which use a double word for storage and can be expressed as follow :

floating point number = sign + Exponent + Mantissa

Sign	Exponent	Mantissa
b_{31}	b_{30} ～ b_{23}	b_{22} ～ b_0
1 bit	8 bits	23 bits

32 bits

Representation of Floating Point Number

If the sign bit is 0 the number is positive, if the sign bit is 1 the number is negative.
The exponent is denoted as 8-bit excess 127. For example, if the value of exponent is 128, it represents the power of 1, if the value of exponent is 129, it represents the power of 2 . . . So on and so forth.If you want to express the negative value of the exponent, then 126 is the power of -1, and 125 is the power of -2. . . So on and so forth.

The mantissa is 23-bit with radix 2. A normalized mantissa always starts with a bit 1, followed by the radix point, followed by the rest of the mantissa. The leading bit 1, which is always present in a normalized mantissa, is implicit and is not represented.

$$N = (-1)^{S} * 2^{(E-127)} * (1.M) \qquad 0 < E < 255$$

Example 1

$1 = (-1)^{0} * 2^{(01111111)} * (1.000 \cdot\cdot\cdot\cdot\cdot\cdot\cdot\ 0)$

The sign is represented by 0, the exponent's code in excess 127 is 127 = 01111111, and the significant bit is 1, resulting in the mantissa being all O's. The simple precision IEEE 754 representation of 1, is thus:

Code(1) =

0	0	1	1	1	1	1	1	1	0	0	0	0	0···············0	0	0
s	e	e	e	e	e	e	e	e	m	m	m	m	m m······ m	m	m

= 3F800000H

Example 2

$0.5 = (-1)^{0} * 2^{(01111110)} * (1.000 \cdot\cdot\cdot\cdot\cdot\cdot\cdot\ 0)$

The sign is represented by 0, the exponent's code in excess 127 is 126 - 127 = 01111110, and the significant bit is 1, resulting in the mantissa being all O's. The simple precision IEEE 754 representation of 0.5, is thus:

Code(0.5) =

0	0	1	1	1	1	1	1	0	0	0	0	0	0···············0	0	0
s	e	e	e	e	e	e	e	e	m	m	m	m	m m······ m	m	m

= 3F000000H

Example 3

$-500.125 = (-1)^{1} * 2^{(10000111)} * (1.11110100001000000000000)$

The sign is represented by 1, the exponent's code in excess 127 is 135 - 127 = 10000111, and the significant bit is 1, resulting in the mantissa is 11110100001000000000000. The simple precision IEEE 754 representation of -500.125, is thus:

1	1	0	0	0	0	1	1	1	1	1	1	1	0	1	0	0	0	0	1	0	0	0	00······0	0
s	e	e	e	e	e	e	e	e	m	m	m	m	m	m	m	m	m	m	m	m	m	m	mm······m	m

= C3FA1000H

5-5 Overflow and Underflow of Increment (+1) or Decreme nt (-1) Instruction (Beginners please skip this section

The maximum positive value that can be represented by 16-bit and 32-bit operands are 32767 and 2147483647, and the maximum negative value are -32768 and -2147483648, respectively. When increase or decrease an operand (e.g., when Up/Down Count of a counter or the register value is +1 or -1), and the result exceeds the value of the positive limit of the operand, then “Overflow” (OVF) occurs. This will cause the value to cycle to its negative limit (e.g., add 1 to the 16-bit positive limit 32767 will change it to –32768). If the result is smaller than the negative limit of the operand, then “Underflow” (UDF) occurs. This will cause the value to cycle to its positive limit (e.g., deducting 1 from the negative limit –32768 will change it to 32767) as shown in the table below. The flag output of overflow or underflow exists in the FO of M-Series PLC and can be used in cascaded instructions to obtain over 16-bit or 32-bit operation results.

Increase (Decrease) Result / Overflow/ Underflow	16-bit Operand	32-bit Operand
Increase	OVF = 1 -32767 -32768 32767 32766 32765	OVF = 1 -2147483646 -2147483647 -2147483648 2147483647 2147483646
Decrease	UDF = 1 -32767 -32768 32767 32766 32765	UDF = 1 -2147483647 -2147483648 2147483647 2147483646 2147483645

Increment or Decrement in 16-bit and 32-bit Operand

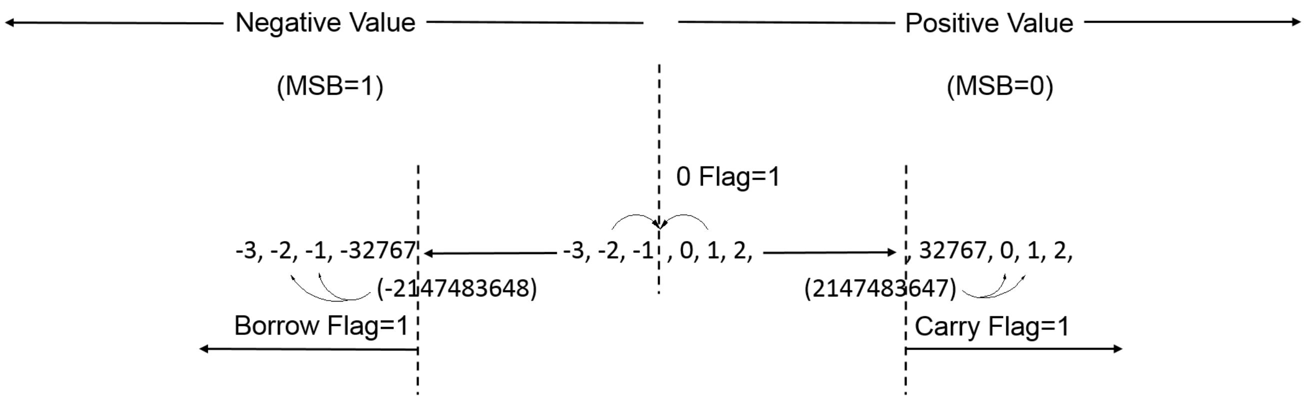
5-6 Carry and Borrow in Addition/Subtraction

Overflow/Underflow takes place when the operation of increment/decrement causes the value of the operand to exceed the positive/negative limit that can be represented in the PLC, consequently a flag of overflow/underflow is introduced. Carry/Borrow flag is different from overflow/underflow. At first, there must be two operands making addition (subtraction) where a sum (difference) and a flag of carry/borrow will be obtained. Since the number of bits of the numbers to be added (subtracted), to add (subtract) and of sum (difference) are the same (either 16-bit or 32-bit), the result of addition (subtraction) may cause the value of sum (difference) to exceed 16-bit or 32-bit. Therefore, it is necessary to use carry/borrow flag to be in coordination with the sum (difference) operand to represent the actual value. The carry flag is set when the addition (subtraction) result exceeds the positive limit (32767 or 2147483647) of the sum (difference) operand. The borrow flag is set when addition (subtraction) result exceeds the negative limit (-32768 or -2147483648) of the sum (difference) operand. Hence, the actual result after addition (subtraction) is equal to the carry/borrow plus the value of the sum (difference) operand. The FO of M-Series PLC addition/subtraction instruction has both carry and borrow flag outputs for obtaining the actual result.

		16-bit / 32-bit	To Be Added (Subtracted) Operand
+(-)		16-bit / 32-bit	Addition (Subtraction) Operand
1-bit	Carry/Borrow	16-bit or 32-bit	Sum (Difference) Operand

16-bit and 32-bit Addition/Subtraction

While all M-Series PLC numerical operations use 2'S Complement, the representation of the negative value of the sum (difference) obtained from addition (subtraction) is different from the usual negative number representation. When the operation result is a negative value, 0 can never appear in the MSB of the sum (difference) operand. The carry flag represents the positive value 32768 (2147483648) and the borrow flag represents the negative value -32768 (-2147483648).



			MSB ... LSB	
C=1	B=0	Z=0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	32769
C=1	B=0	Z=0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	32768
C=0	B=0	Z=0	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	32767
C=0	B=0	Z=0	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0	32766
C=0	B=0	Z=0	0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1	32765
C=0	B=0	Z=0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0	2
C=0	B=0	Z=0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	1
C=0	B=0	Z=1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
C=0	B=0	Z=0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	-1
C=0	B=0	Z=0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0	-2
C=0	B=0	Z=0	1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0	-32766
C=0	B=0	Z=0	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	-32767
C=0	B=0	Z=0	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	-32768
C=0	B=1	Z=0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	32769
C=0	B=1	Z=0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0	-32770

Positive Value

C = Carry B = Borrow Z = Zero

Carry and Borrow in Addition/Subtraction

※If carry and borrow processing is not required, it is recommended to use Fun224 fast addition and Fun225 fast subtraction, because compared with Fun11 addition and Fun12 subtraction, no carry/borrow is required

6

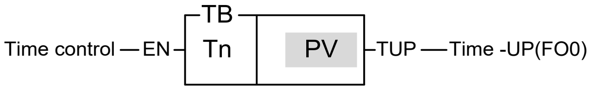
Basic Function Instructions

6-1 TIMER(T)

T	TIMER	T
Command Description		

Operand

Ladder symbol



TB: Time Base (0.01S, 0.1S, 1S)

Tn: Timer Number

PV: Preset value of the timer.

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	WX0 ~ WX1008	WY0 ~ WY1008	WM0 ~ WM29584	WS0 ~ WS3088	T0 ~ T1023	C0 ~ C1279	R0 ~ R34767	R34768 ~ R34895	R35024 ~ R35151	R35280 ~ R43223	R43224 ~ R48471	D0 ~ D11999	0 ~ 32767
Tn					○								
PV	○	○	○	○	○	○	○	○	○	○	○	○	○

- The total number of timers is 1024 (T0 ~ T1023) with three different time bases, 0.01S, 0.1S and 1S.The default number and allocation of timers is shown as below (Can be adjusted according to user’s actual requirements by the “Configuration” function):

  T0 ~ T255 ：0.001S timer（default as 0.00 ~ 32.767S）

  T256 ~ T511 ：0.01S timer（default as 0.0 ~ 327.67S）

  T512 ~ T767 ：0.1S timer（default as 0 ~ 3276.7S）

  T768 ~ T1023 ：1S timer（default as 0 ~ 32767S）
- The Timer of M-Series PLC will start to run when subprogram triggers. To stop the Timer operation, you must Disable the Timer, not calling the subprogram, which is not equal to Disable Timer. If there is no Disable Timer, it will continue to count.

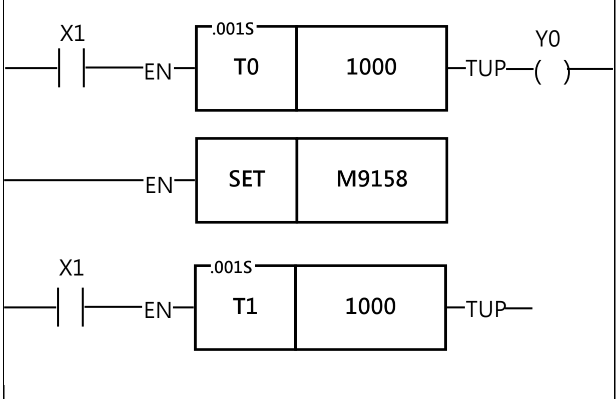
T	TIMER	T

- If PV is a register, then Timer's time = Time base x register content. Therefore, you only need to change the register content to change the timer's time. Please refer to Example 2.
- The maximum error of a timer is a time base plus a scan time. In order to reduce the timing error in the application, please use the timer with a smaller time base.

Function Description	

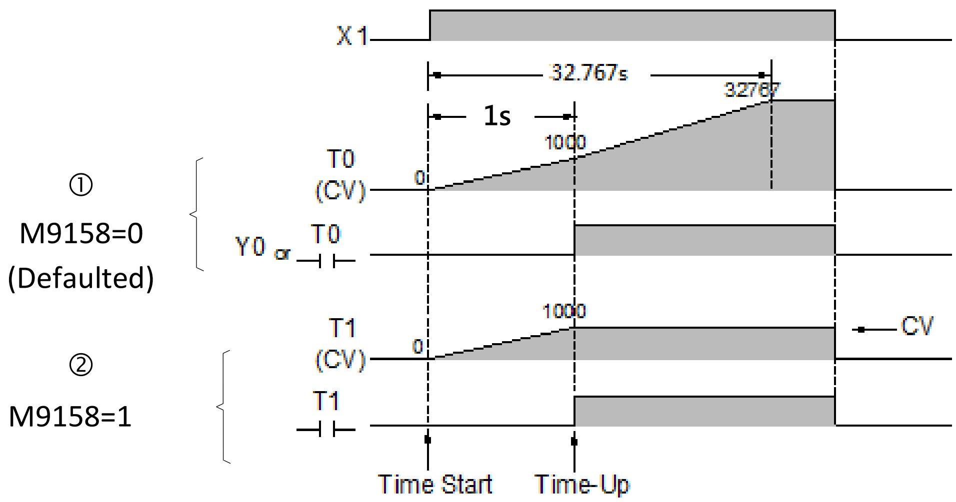
- When the time control "EN" is 1, the timer will start timing (the current value will accumulate from 0) until "Time Up" (i.e., CV≧PV), then the Tn contact and TUP (FO0) will change to 1. As long as the timer control "EN" input is kept as 1, even the CV of Tn has reached or exceeded the PV, the CV of the timer will continue accumulating (with M9158 = 0) until it reaches the maximum limit (32767). The Tn contact status and flag will remain as 1 when CV≧PV, unless the "EN" input is 0. When "EN" input is 0, the CV of Tn will be reset to 0 immediately and the Tn contact and "Time Up" flag TUP will also change to 0 (please refer to the diagram ① below).
- M-Series PLC can set the M9158 to 1 so the CV will not accumulate further after "Time Up"and stops at the PV value. The default value of the M9158 is 0, therefore the status of M9158 can be set before executing any timer instruction inthe program to individually set the timer CV to continue accumulating or stop at the PV after "Time Up" (please refer to the diagram ② below).

T	TIMER	T
Example 1	Fixed Time Timer	

<table>
<tr><th>Ladder diagram</th><th>ST</th></tr>
<tr><td>
</td><td rowspan="2">Timer(EN:= X1,
      T:= 0,
      PV:= 1000,
      IsTimeout=> Y0);

M9158 := TRUE;

Timer(EN:= X1,
      T:= 1,
      PV:= 1000,
      IsTimeout=> T1);</td></tr>
<tr><td>An example of taking “Time-Up” signal directly from FO0.</td></tr>
</table>

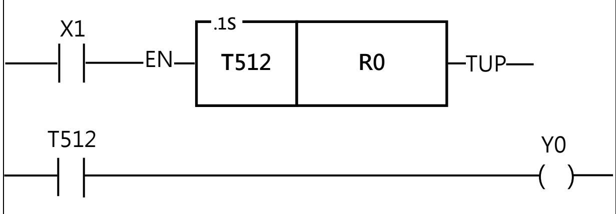


T	TIMER	T

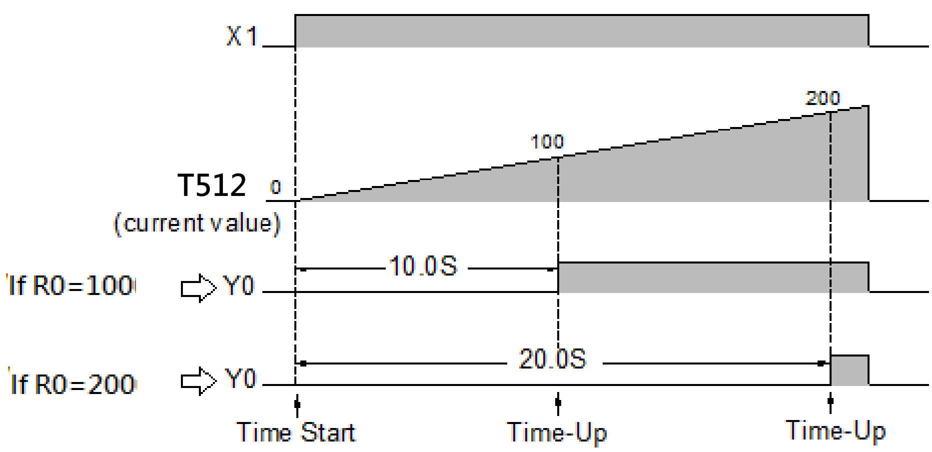
Example 2 Variable Time Timer

The preset value (PV) shown in example 1 is a constant which is equal to 1000. This value is fixed and can not be changed once programmed. In many circumstances, the preset time of the timers needs to be varied while PLC running. In order to change the preset time of a timer, can first use a register as the PV operand (R or D...) and then the preset time can be varied by changing the register content. As shown in this example, if set R0 to 100, then T becomes a 10S Timer, and hence if set R0 to 200, then T becomes a 20S Timer. So that we can easily change the timer dynamically while the PLC is running.

Ladder diagram	ST

	`Timer( EN:= X1,`
	`T:= 512,`
	`PV:= R0,`
	`IsTimeout=> T512);`
	
	`IF T512 THEN`
	`Y0 := TRUE;`
	`ELSE`
	`Y0 := FALSE;`
	`END_IF`

An example of applying the “time-up” status by using the T512 contact.	

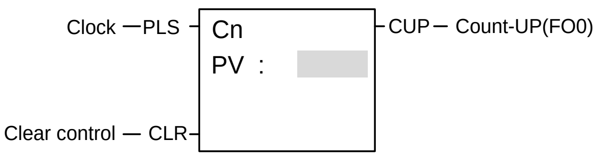


T	TIMER	T
Note: If the preset value of the timer is equal to 0, then the timer's contact status and become 1 ("EN" input must be at 1) immediately, after the PLC finishes its first scan because "Time-Up" has occurred. (TUP) stays at 1 until "EN" input changes to 0.		

6-2 COUNTER(C)

C	COUNTER (16-Bit: C0 ~ C1023，32-Bit: C1024 ~ C1279)	C
Command Description		

Ladder symbol



Operand
Cn: The Counter number
PV: Preset value

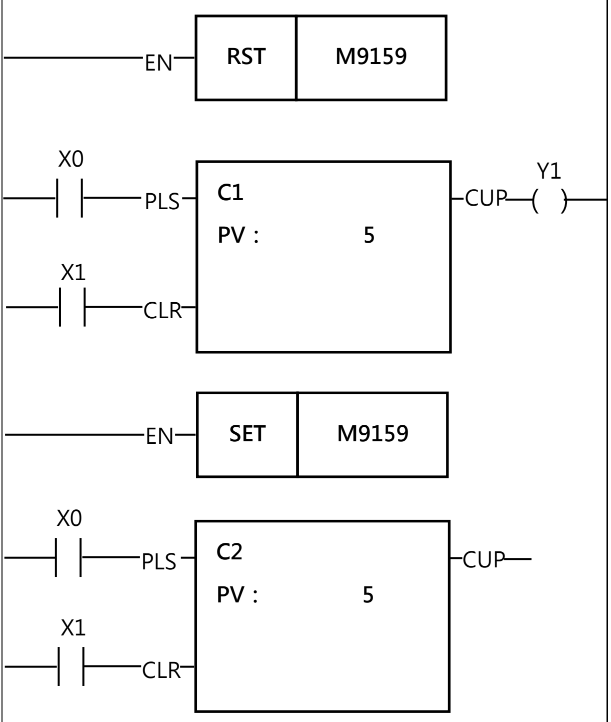
Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	WX0 ~ WX1008	WY0 ~ WY1008	WM0 ~ WM29584	WS0 ~ WS3088	T0 ~ T1023	C0 ~ C1279	R0 ~ R34767	R34768 ~ R34895	R35024 ~ R35151	R35280 ~ R43223	R43224 ~ R47319	D43224 ~ D11999	0 ~ 32767
Cn						○							
PV	○	○	○	○	○	○	○	○	○	○	○	○	○

- There is total 1024 16-Bit counters (C0~C1023). The range of preset value is between 0~32767. C0~C139 are Retentive Counters and the CV value will be retained when the PLC turns on or RUN again after a power failure or a PLC STOP. C140~C1023 are Non-Retentive Counters, if a power failure or PLC STOP occurs, the CV value will be reset to 0 when the PLC turns on or RUN again.
- There is total 56 32-Bit counters (C1024~C1079). The range of the preset value is between 0~2147483647. C1024~C1063 are Retentive Counters and C1063~C1279 are Non-Retentive Counters.
- The above 16-bit and 32-bit counters' retentive/non-retentive number distribution is the original factory setting. If this does not meet your needs, you can use the "Frame Configuration" function to adjust.
- To ensure the proper counting from C0~C1024, the sustain time of input status of CLK should greater than 1 scan time.
- The max counting frequency with this instruction can only up to 20Hz, for higher frequency please use the high-speed soft/hardware counter.

C	COUNTER (16-Bit: C0 ~ C1023，32-Bit: C1024 ~ C1279)	C
Function Description		

- When "CLR" is at 1, all of the contact Cn, FO0 (CUP), and CV value of the counter CV are cleared to 0 and the counter stops counting.
- When "CLR" = 0, the counter is allowed to count, because the counter command is essentially a "P command", so only when the counting pulse "PLS" changes from 0 to 1, the current value CV of the counter Cn will increase by 1. Until "Count up" (Count up, that is, CV value ≧ set value), the count up contact Cn and the count up flag CUP (FO0) of the counter will both become 1. If there is still counting pulse input at this time, the current value CV of Cn will exceed the set value and continue to accumulate (when M9159=0), until it reaches the upper limit (32767 or 2147483647), and the Cn contact and the counting flag CUP will As long as CV≧PV, it will always be 1, unless the clear control CLR input becomes 1. (Please refer the diagram ① below)
- M-Series PLC can set the M9159 to 1 so the CV will not accumulate further after “Count Up”and stops at the PV. M9159 default value is 0, therefore the status of M9159 can be set before executing any counter instruction in the program to individually set the counter CV to continue accumulating or stops at the PV after “Count Up” (please refer to the diagram ② below).

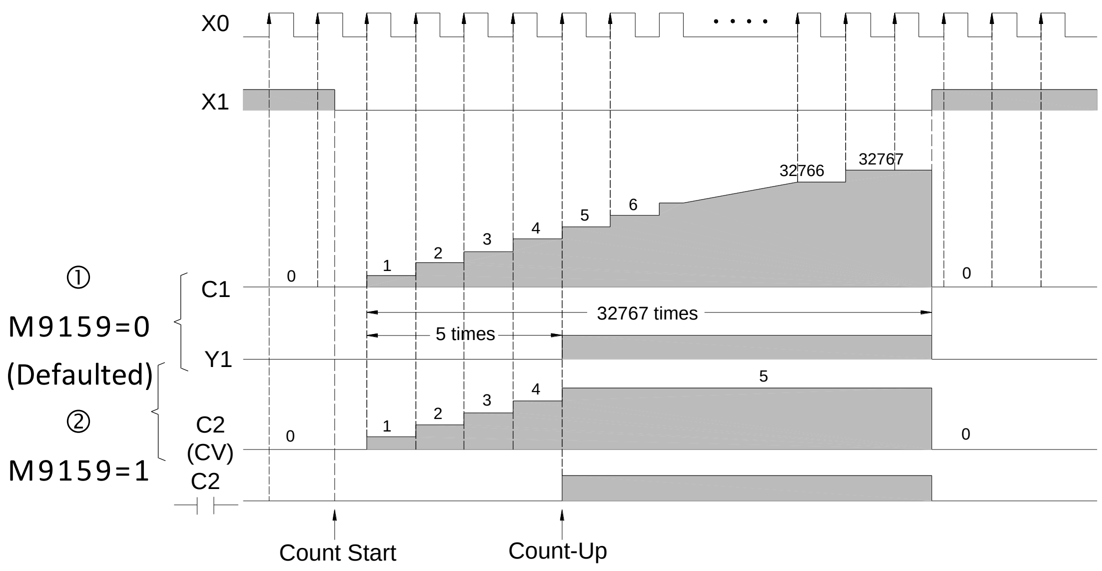
C	COUNTER (16-Bit: C0 ~ C1023 ， 32-Bit: C1024 ~ C1279)	C
Example 1	Fixed Time Counter	

<table>
<tr><th>Ladder diagram</th><th>ST</th></tr>
<tr><td>
</td><td>

```
M9159 := FALSE;

Counter( Pulse:= X0,
         Clr:= X1,
         C:= 1,
         PV:= 5,
         IsUp=> Y1);

M9159 := TRUE;

Counter( Pulse:= X0,
         Clr:= X1,
         C:= 2,
         PV:= 5,
         IsUp=> C2);
```

</td></tr>
<tr><td>An example of applying the “Count-Up” status by using FO0 directly.</td><td></td></tr>
</table>

C	COUNTER (16-Bit: C0 ~ C1023，32-Bit: C1024 ~ C1279)	C

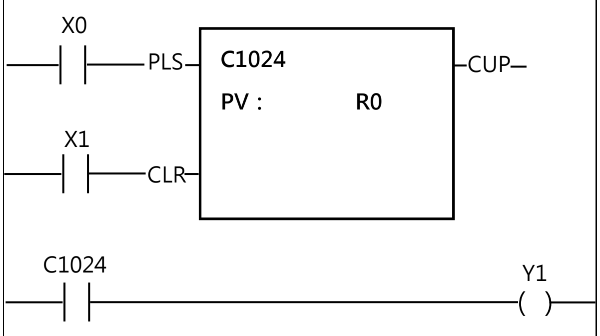


Example 2	32-Bit counter with variable value

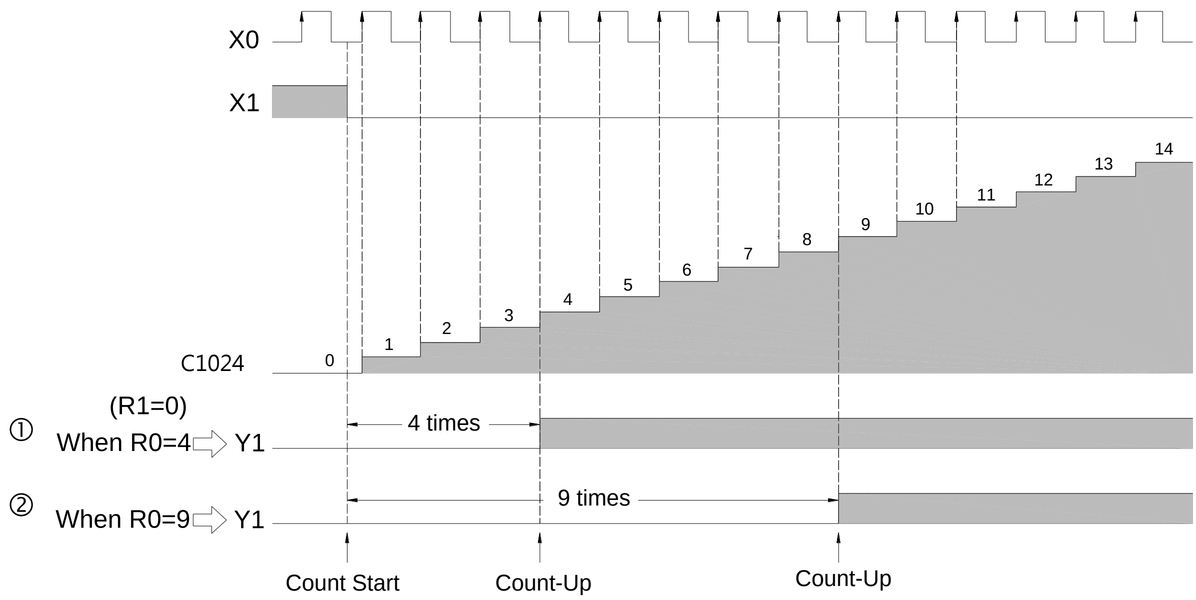
Like a timer, if the PV of a counter is changed to a register (such as R, D, and so on), the counter will use the register contents as the counting PV. Therefore, only need to change the register contents to change the PV of the counter while PLC is running. Below is an example of a 32-bit counter that uses the data register R0 as the PV (in fact it is the 32-bit PV formed by R1 and R0).

C	COUNTER (16-Bit: C0 ~ C1023，32-Bit: C1024 ~ C1279)	C

Ladder diagram	ST


 	
	<pre>Counter(Pulse:= X0,
	        Clr:= X1,
	        C:= 1024,
	        PV:= R0,
	        IsUp=> C1024);
	
	IF C1024 THEN
	Y1 := TRUE;
	ELSE
	Y1 := FALSE;
	END_IF</pre>

An example of applying the “Time-up” status by using the C1024 contact.	

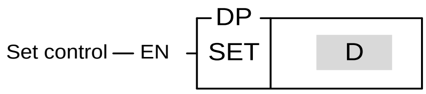


Note: If the preset value of the counter is 0 and "CLR" input also at 0, then the Cn contact status and FO0 (CUP) becomes 1 immediately after the PLC finishes its first scan because the "Count-Up" has occurred. It will stay at 1 regardless how the CV value varies until "CLR" input changes to 1.

6-3 SET(S)

SET D P	SET (Set coil or all the bits of register to 1)	SET D P
Command Description		

Ladder symbol



Operand
D: destination to be set
(The number of a coil or a register)

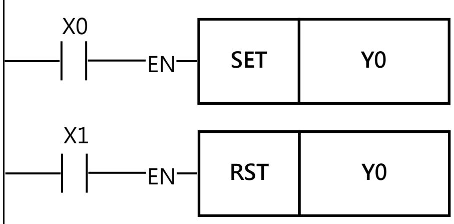
Operand \ Range	Y	M	SM	S	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR
	Y0 Y1023	M0 M9119	M9120 M29599	S0 S3103	WY0 WY1008	WM0 WM29584	WS0 WS3088	T0 T1024	C0 C1280	R0 R34767	R35024 R35151	R35280 R40279	R40280 R48471	D0 D11999
D	○	○	○*	○	○	○	○	○	○	○	○	○*	○*	○

Function Description

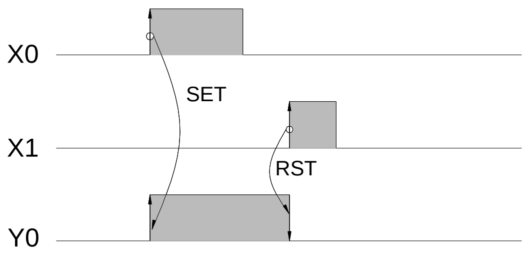
- When the set control "EN" =1 or from 0 → 1 (P instruction), sets the bit of a coil or all bits of a register to 1.

Example 1 Single Coil Set

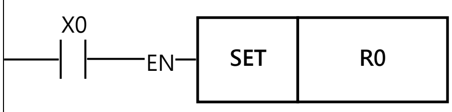
Ladder diagram	ST

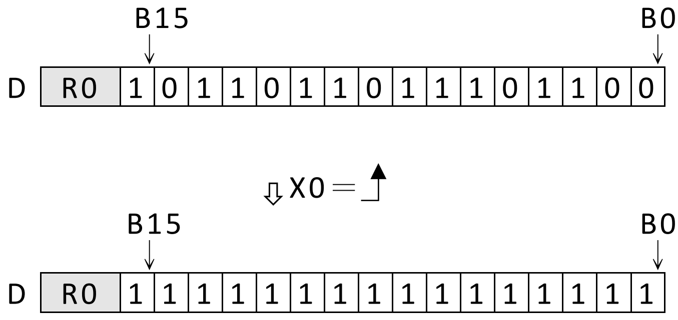
	IF X0 THEN
	Y0 := TRUE;
	END_IF
	IF X1 THEN
	Y0 := FALSE;
	END_IF

SET D P	SET (Set coil or all the bits of register to 1)	SET D P

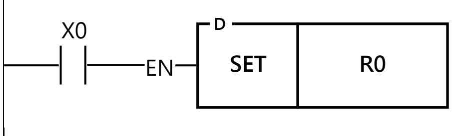


Example 2	Set 16-Bit Register

Ladder diagram	ST
	IF X0 THEN R0 := -1; END_IF



Example 3	32-Bit Register Set

Ladder diagram	ST
	IF X0 THEN DR0 := -1; END_IF

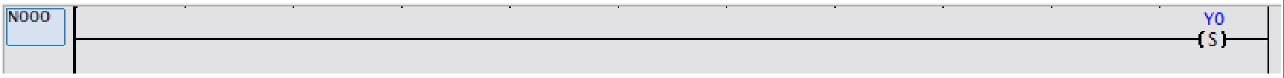
B31 R1 R0 B0

D	R0	1	0	1	1	0	1	1	0	1	0	1	1	1	0	1	1	1	1	1	1	0	1	0	1	1	0	1	1	0	0	0	1

⇩X0 = 1

D	R0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

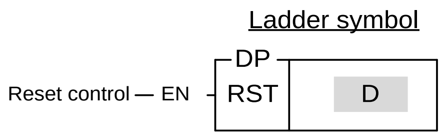
Note:
If you use a single contact set (Y, M, S), it is recommended to use "coil set", the efficiency of using the PLC will be better than set instruction
The example is as follows:



6-4 RST(R)

RST D P	RESET (Reset the coil or the register to 0)	RST D P

Command Description



Operand
D: Destination to be reset
(The number of a coil or a register)

Range / Operand	Y	M	SM	S	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR
	Y0 | Y1023	M0 | M9119	M9120 | M29599	S0 | S3103	WY0 | WY1023	WM0 | WM29584	WS0 | WS3104	T0 | T1024	C0 | C1280	R0 | R34767	R35024 | R35151	R35280 | R40279	R40280 | R48471	D0 | D11999
D	○	○	○*	○	○	○	○	○	○	○	○	○*	○*	○

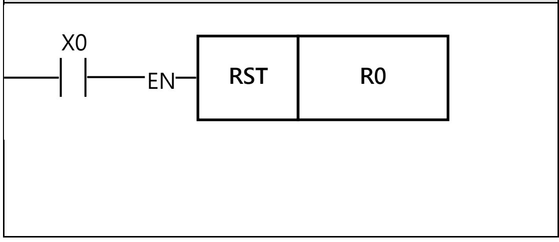
Description

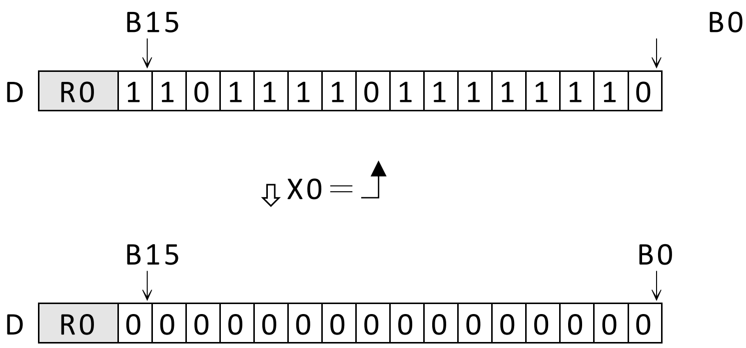
- When the reset control "EN" =1 or from 0 → 1 (P instruction), resets the coil or register to 0.

Example 1 Single Coil Reset

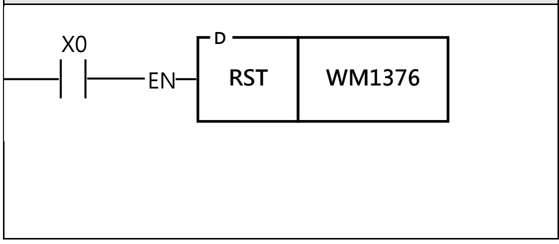
Please refer to example 1 for the SET instruction

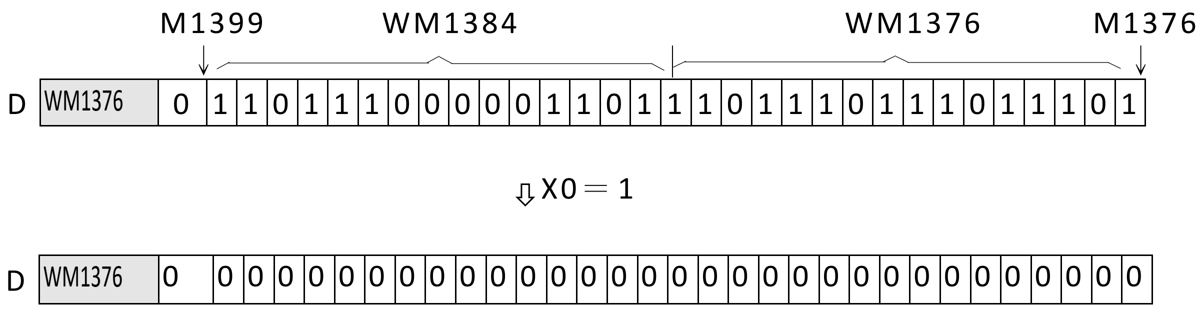
Example 2 16-Bit Register Reset

Ladder diagram	ST
	IF X0 THEN R0 := 0; END_IF



Example 3	32-Bit Register Reset

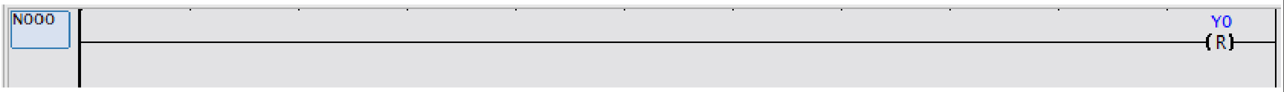
Ladder diagram	ST
	IF X0 THEN WM1376 := 0; END_IF



Note:

If you use a single contact reset (Y, M, S), it is recommended to use "coil reset", the efficiency of using the PLC will be better than set instruction

The example is as follows:

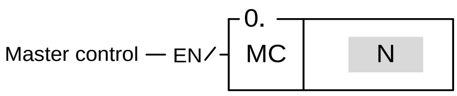


6-5 MASTER CONTROL(MC)

FUN 0 MC	MASTER CONTROL LOOP START	FUN 0 MC

Command Description

Ladder symbol



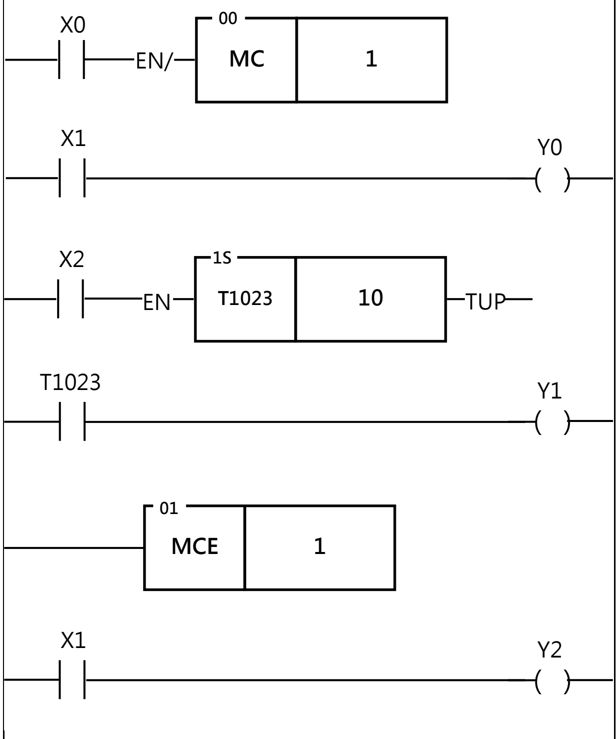
Operand

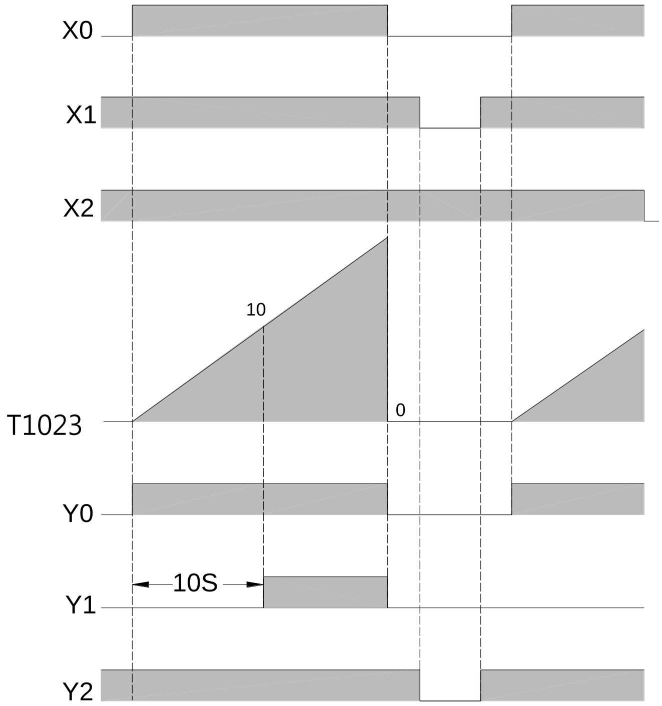
N: Master Control Loop number (N=0~127) the number N cannot be used repeatedly.

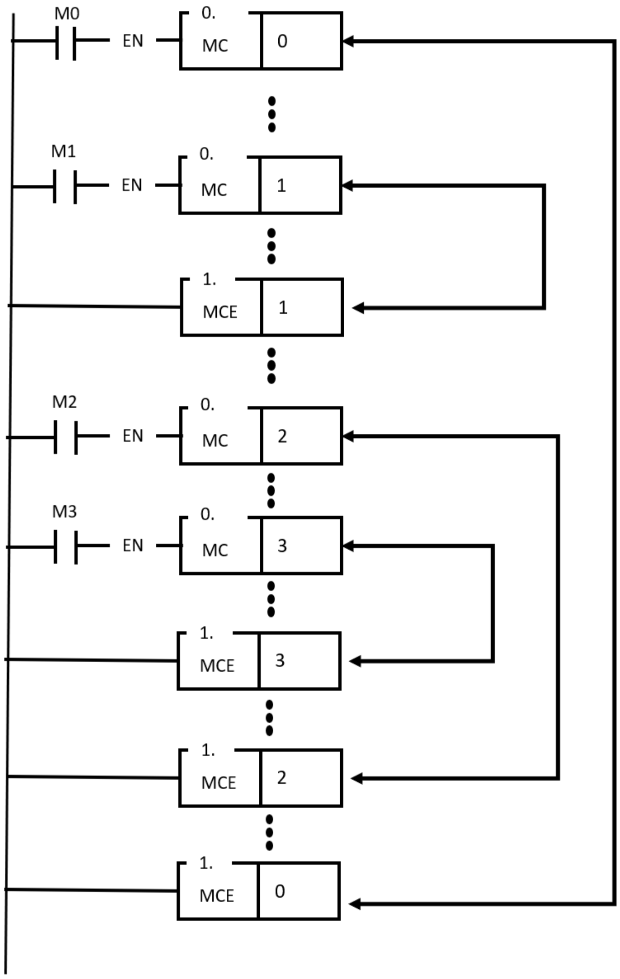
Description

- M There are a total of 128 MC loops (N=0~127). Every Master Control Start instruction, MC N, must correspond to a Master Control End instruction, MCE N, which has the same loop number as MC N. They must always be used in pairs and you should also make sure that the MCE N instruction is after the MC N instruction.
- When the Master Control input "EN/" is 1, then this MC N instruction will not be executed, as it does not exist.
- When the Master Control input "EN/" is 0, the master control loop is active, the area between the MC N and MCE N is called the Master Control active loop area. All the status of OUT coils or Timers within Master Control active loop area will be cleared to 0. Other instructions will not be executed.

FUN 0 MC	MASTER CONTROL LOOP START	FUN 0 MC

Ladder diagram	ST
	



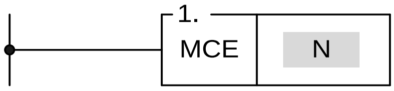
<table>
<tr><td>FUN 0
MC</td><td colspan="2">MASTER CONTROL LOOP START</td><td>FUN 0
MC</td></tr>
<tr><td colspan="2">Note 1: MC/MCE instructions can be used in nesting or interleaving as shown to the right:</td><td colspan="2">
</td></tr>
</table>

6-6 MASTER CONTROL END(MCE)

FUN 1 MCE	MASTER CONTROL LOOP END	FUN 1 MCE

Command Description

Ladder symbol



Operand

N: Master Control End number (N=0~127) N can not be used repeatedly.

Description

- Every MCE N must correspond to a Master Control Start instruction. They must always be used as a pair and you should also make sure that the MCE N instruction is after the MC N instruction. After the MC N instruction has been executed, all output coil status and timers will be cleared to 0 and no other instructions will be executed. The program execution will resume until a MCE instruction which has the same N number as MC N instruction appears.

- MCE instruction does not require an input control because the instruction itself forms a network which other instructions can not connect to it. If the MC instruction has been executed then the master control operation will be completed when the execution of the program reaches the MCE instruction. If MC N instruction has never been executed then the MCE instruction will do nothing.

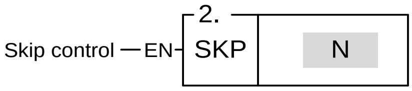
Example 1

- Please refer to the example and explanations for MC instruction.

6-7 SKIP(SKP)

FUN 2 SKP	SKIP START	FUN 2 SKP

Command Description

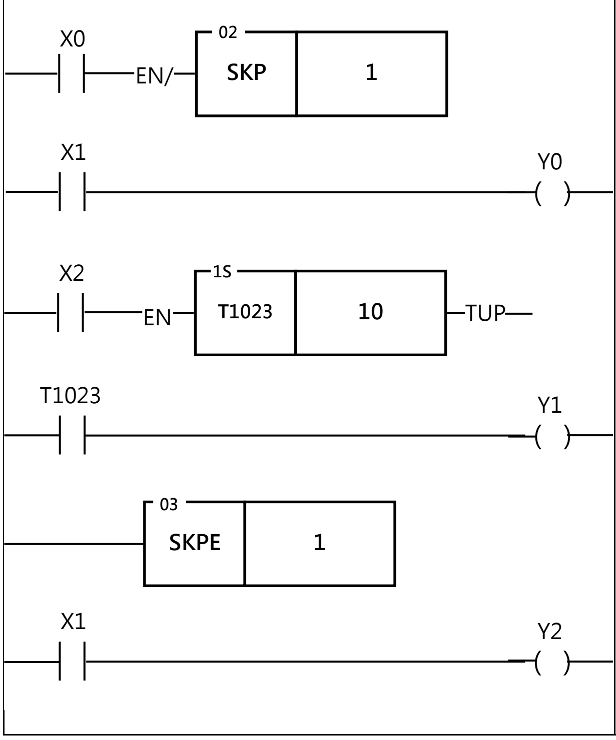
Ladder symbol	Operand
	N: Skip loop number (N=0~127), N can not be used repeatedly.

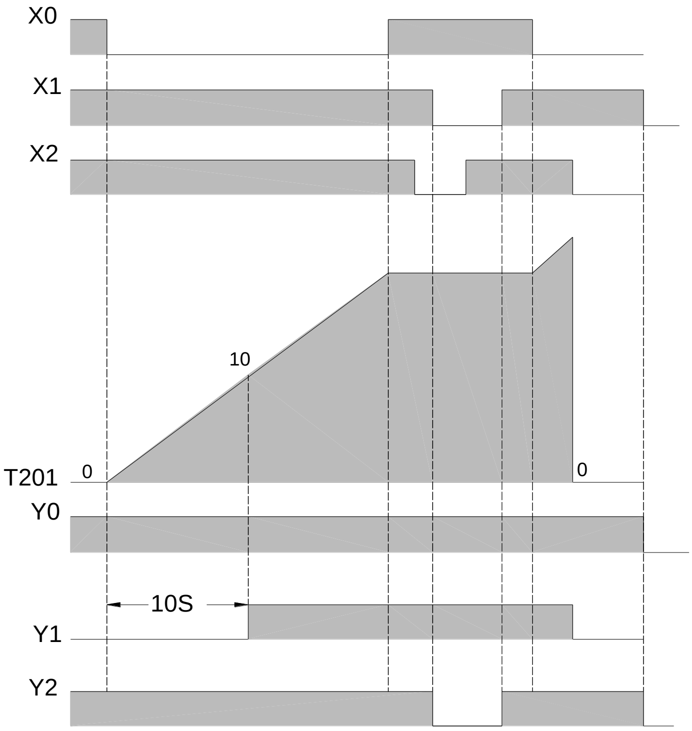
Description

- There is total 128 SKP loops (N=0~127). Every skip start instruction, SKP N, must correspond to a skip end instruction, SKPE N, which has the same loop number as SKP N. They must always be used as a pair and you should also make sure that the SKPE N instruction is after the SKP N instruction.
- When the skip control "EN" is 0, then the Skip Start instruction will not be executed (An equivalent SKP N command does not exist).
- When the skip control "EN" is 1, the range between the SKP N and SKPE N which is so called the Skip active loop area will be skipped, that is all the instructions in this area will not be executed. Therefore, the statuses of the discrete or registers in this Skip active loop area will be retained.

FUN 2 SKP	SKIP START	FUN 2 SKP
Example		

Ladder diagram	ST



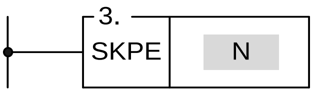


6-8 SKIP END(SKPE)

FUN 3 SKPE	SKIP END	FUN 3 SKPE

Command Description

Ladder symbol



Operand

N ： SKIP END Loop number (N=0~127) N can not be used repeatedly.

Description

- Every SKPE N must correspond to a SKP N instruction. They must always be used as a pair and you should also make sure that the SKPE N instruction is behind the SKP N instruction.
- SKPE instruction does not require an input control because the instruction itself forms a network which other instructions can not connect to it. If the SKP N instruction has been executed then the skip operation will be completed when the execution of the program reaches the SKPE N instruction. If SKP N instruction has never been executed then the SKPE instruction will do nothing.

Example 1

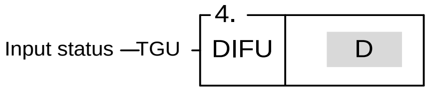
Please refer to the example and explanations for SKP N instruction.
Note: SKP/SKPE instructions can be used by nesting or interleaving. The coding rules are the same as for the MC/MCE instructions. Please refer to the section of MC/MCE instructions.

6-9 DIFFERENTIAL UP (DIFU)

FUN 4 P DIFU	DIFFERENTIAL UP	FUN 4 P DIFU

Command Description

Ladder symbol



Operand

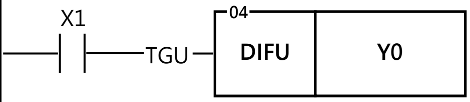
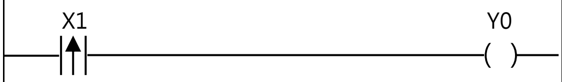
D: The specific coil number where the result of the Differential Up operation is stored.

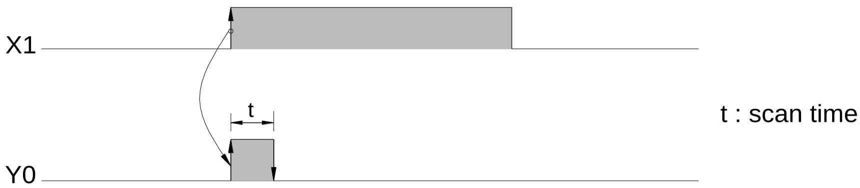
Range / Operand	Y	M	SM	S
	Y0 – Y1023	M0 – M19583	M9120 – M29599	S0 – S3104
D	○	○	○*	○

Description

- The DIFU instruction is used to output the up differentiation of a node status (status input to "TGU") and the pulse signal resulting from the status change at the rising edge of the "TGU" for one scan time is stored to a coil specified by D.
- The functionality of this instruction can also be achieved by using a TU contact. 。

Example	The results of the following two samples are exactly the same

Ladder diagram	ST
Example 1 	`R_TRIG( S:= X1, D=>Y0 );`
Example 2 	`R_TRIG( S:= X1, D=>Y0 );`

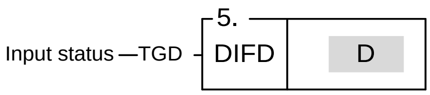


6-10 DIFFERENTIAL DOWN(DIFD)

FUN 5 P DIFD	DIFFERENTIAL DOWN	FUN 5 P DIFD

Command Description

Ladder symbol



Operand

D: The specific coil number where the result of the Differential Up operation is stored.

Range / Operand	Y	M	SM	S
	Y0 – Y1023	M0 – M19583	M9120 – M29599	S0 – S3103
D	○	○	○*	○

Description

- The DIFD instruction is used to output the down differentiation of a node status (status input to "TGD") and the pulse signal resulting from the status change at the falling edge of the "TGD" for one scan time is stored to a coil specified by D.
- The functionality of this instruction can also be achieved by using a TD contact.

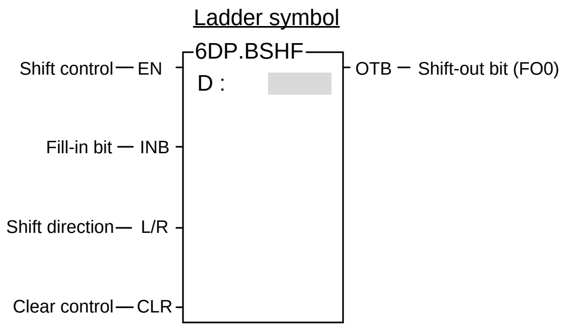
Example	The results of the following two samples are exactly the same

Ladder Diagram	ST
Example 1  	`F_TRIG( S:= X1, D=>Y0 );`
Example 2  	`R_TRIG( S:= X1, D=>Y0 );`

6-11 BIT SHIFT(BSHF)

FUN 6 D P BSHF	BIT SHIFT (Shifts the data of the 16-bit or 32-bit register to left or to right by one bit)	FUN 6 D P BSHF

Command Description



Operand

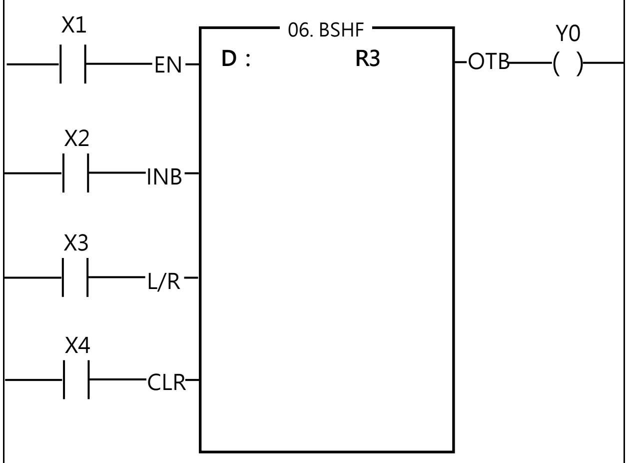
D: The register number for shifting

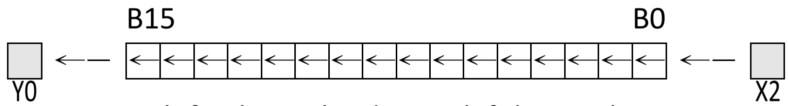
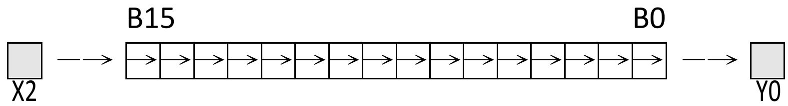
Range / Operand	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR
	WY0–WY1008	WM0–WM29584	WS0–WS3088	T0–T1023	C0–C1280	R0–R34767	R35024–R35151	R35280–R43223	R43224–R47319	D0–D11999
D	○	○	○	○	○	○	○	○*	○*	○

Description

- When the status of clear control "CLR" is at 1, then the data of register D and FO0 will all be cleared to 0. All other input signals are invalid.
- When the status of clear control is "CLR" at 0, then the shift operation is permissible. When the shift control "EN" = 1 or from 0 →1 (P Instruction), the data of the register will be shifted to right (L/R=0) or to left (L/R=1) by one bit. The shifted-out bit (MSB when shift to left and LSB when shift to right) for both cases will be sent to FO0. The vacated bit space (LSB when shift to left and MSB when shift to right) due to shift operation will be filled in by the input status of fill-in bit "INB".

FUN 6 D P BSHF	BIT SHIFT (Shifts the data of the 16-bit or 32-bit register to left or to right by one bit)	FUN 6 D P BSHF
Example	Shifts the 16-bit register data	

Ladder Diagram	ST
	

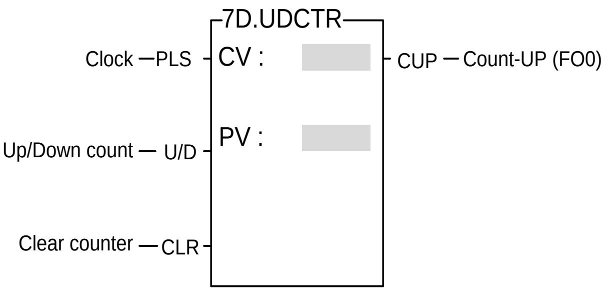
X3=1 (Left shift)	 Shifts the 16-bit data to left by one bit
X3=0 (Right shift)	 Shifts the 16-bit data to right by one bit

6-12 UP/DOWN COUNTER(UDCTR)

FUN 7 D P UDCTR	UP/DOWN COUNTER (16-bit or 32-bit up and down 2-phase Counter)	FUN 7D P UDCTR

Command Description

Ladder symbol



Operand

CV: The number of the Up/Down Counter

PV: Preset value of the counter or it's register number

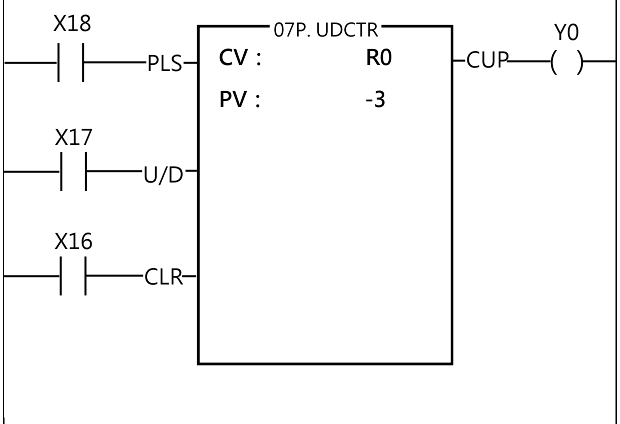
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	WX0 – WX1008	WY0 – WY1008	WM0 – WM29584	WS0 – WS3088	T0 – T1023	C0 – C1279	R0 – R34767	R34768 – R34895	R35024 – R35151	R35280 – R43223	R43224 – R47319	D0 – D11999	16/32-bit +/– number
CV		○	○	○	○	○	○		○	○*	○*	○	
PV	○	○	○	○	○	○	○	○	○	○	○	○	○

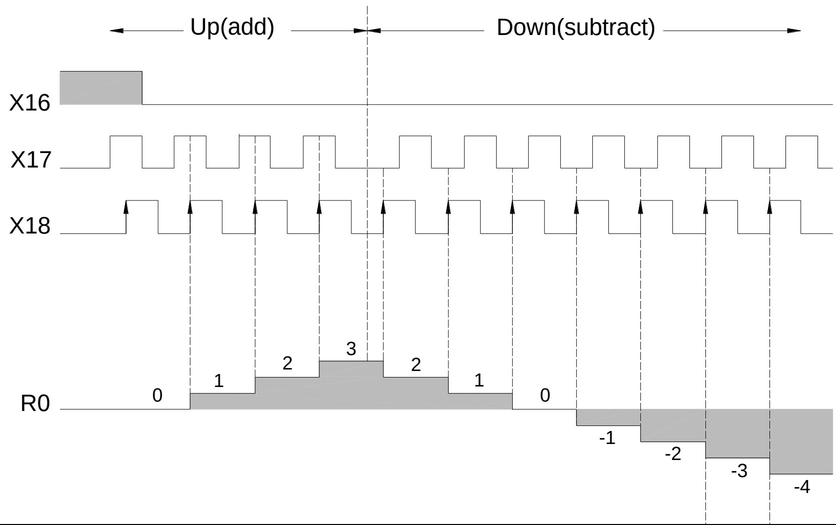
Description

- When the clear control “CLR” is 1, the counter’s CV will be reset to 0 and the counter will not be able to count.
- When the clear control “CLR” is 0, counting will then be allowed. The nature of the instruction is a P instruction. Therefore, when the count-pulse “PLS” is from 0→1 (rising edge), the CV will be increased by 1 (if U/D=1) or decreased by 1 (if U/D=0).

- When CV=PV, FO0(“Count-Up) will change to 1”. If there are more clocks input, the counter will continue counting which cause CV≠PV. Then, FO0 will immediately change to 0. This means the “Count-Up” signal will only be equal to 1 if CV=PV, or else it will be equal to 0 (Care should be taken to this difference from the “Count-Up” signal of the general counter).
- The upper limit of up count value is 32767 (16-bit) or 2147483647 (32-bit). After the upper limit is reached, if another up-count clock is received, the counting value will become –32768 or -2147483648 (the lower limit of down count).
- The lower limit of down count value is -32767 (16-bit) or -2147483647 (32-bit). After the lower limit is reached, if another down count clock is received, the counting value will become 32768 or 2147483648 (the upper limit of up count).
- If U/D is fixed as 1, the instruction will become a single-phase up count counter. If U/D is fixed as 0, the instruction will become a single-phase down count counter.

Example	

Ladder Diagram	ST
	



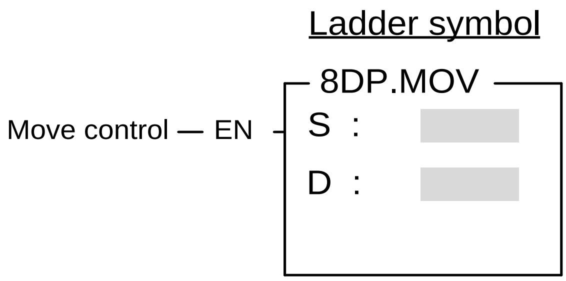
- Note 1: Since the counting operation of UDCTR is implemented by software scanning, therefore if the clock speed is faster than the scan speed, lose count may then happen (generally the clock should not exceed 20Hz depending on the size of the program). Please use the software or hardware high-speed counter in the PLC. Refer to the “High Speed Counter Application” in the Advanced Manual.
- Note 2: In order to ensure that this command can count correctly, the width of the counting pulse must be greater than one scan time no matter it is 1 or 0.

6-13 MOVE(MOV)

FUN 8 DP MOV	MOVE (Moves data from S to D)	FUN 8 DP MOV

Command Description

Ladder symbol



Operand

S: Source register number

D: Destination register number

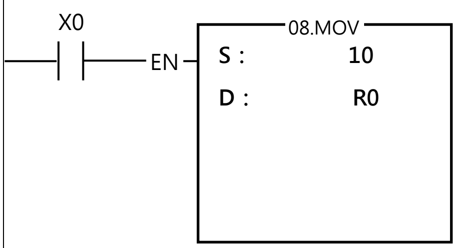
The S, D may combine with V, Z, P0~P9 to serve indirect addressing

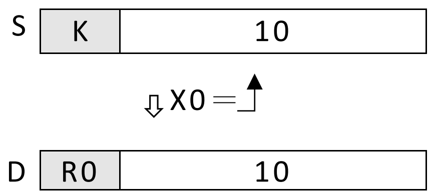
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WM29584	WS0 ∣ WS3008	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34895	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	16/32-bit +/− number	V、Z P0-P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

Description

- Move (write) the data of S to a specified register D when the move control input "EN" =1 or from 0 to 1 (P Instruction).

FUN 8 D P MOV	MOVE (Moves data from S to D)	FUN 8 D P MOV
Example	Writes a constant data into a 16-bit register.	

Ladder Diagram	ST
	IF X0 THEN R0 := 10; END_IF



6-14 MOVE INVERSE(MOV/)

FUN 9 D P MOV/	MOVE INVERSE (Inverts the data of S and moves the result to a specified device D)	FUN 9 D P MOV/

Command Description

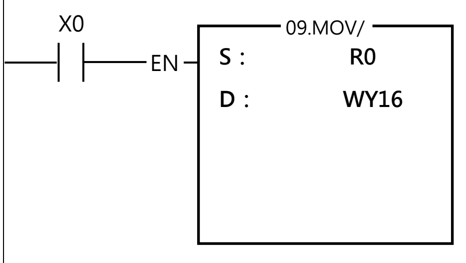
Ladder symbol	Operand
Move control — EN — 9DP.MOV/ S : D :	S: Source register number D: Destination register number S, D may combine with V, Z, P0~P9 to serve indirect addressing

Range / Operand	WX	WY	WM	WS	T	C	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +/-number	V,Z P0-P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

Description

- Inverts the data of S (changes the status from 0 to 1 and from 1 to 0) and moves the results to a specified register D when the move control input "EN" =1 or from 0 to1 (P Instruction).

FUN 9 D P MOV/	MOVE INVERSE (Inverts the data of S and moves the result to a specified device D)	FUN 9 D P MOV/
Example	Moves the inverted data of a 16-bit register to another 16-bit register.	

Ladder Diagram	ST
	IF X0 THEN WY16 := NOT R0; END_IF

B15 B15 B0

S	R0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	5555H

⇩X0＝1

Y31 Y16

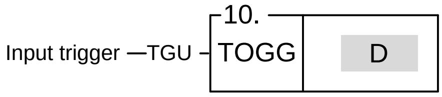
D	WY16	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	AAAAH

6-15 TOGGLE SWITCH(TOGG)

FUN 10 TOGG	TOGGLE SWITCH (Changes the output status when the rising edge of control input occur)	FUN 10 TOGG

Command Description

<u>Ladder symbol</u>



Operand

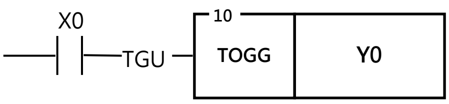
D: the coil number of the toggle switch

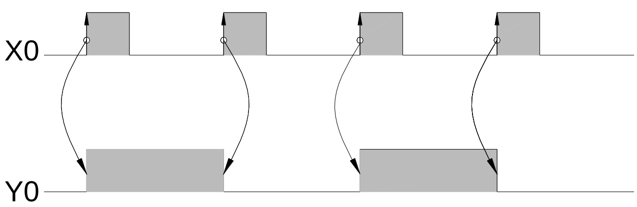
Range / Operand	Y	M	SM	S
	Y0 │ Y1023	M0 │ M19583	M9120 │ M29599	S0 │ S3104
D	○	○	○*	○

Description

- The coil D changes its status (from 1 to 0 and from 0 to 1) each time the input "TGU" is triggered from 0 to 1 (rising edge).

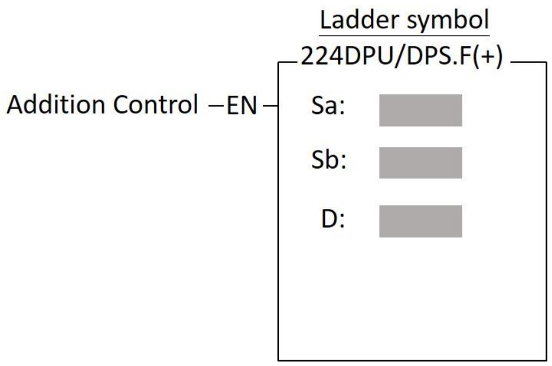
Example

Ladder Diagram	ST
	IF X0 THEN Y0 := NOT Y0; END_IF



6-16 FAST ADDITION F（ + ）

FUN224 D P F（ + ）	Fast ADDITION (Performs addition of the data specified at Sa and Sb and stores the result in D)	FUN224 D P F（ + ）
Command Description	Support after UperLogic: v_0.8.517 visions	



Operand

Sa: Augend

Sb: Addend

D: Destination register to store the results of the addition

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

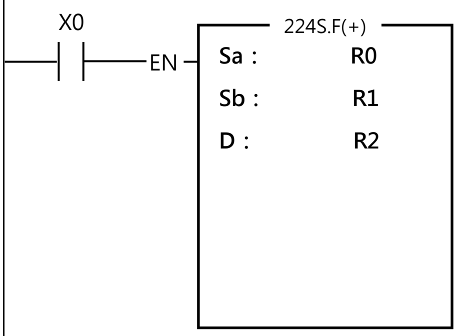
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +/-number	V,Z P0-P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

FUN224 D P F (+)	Fast ADDITION (Performs addition of the data specified at Sa and Sb and stores the result in D)	FUN224 D P F (+)
Description		

- Performs the fast addition control "EN"=1 or from 0→1 (P command) and the command is set to signed (S command), add Sa and Sb with the positive and negative number (Sign) algorithm and write the result into D.
- Performs the fast addition control "EN"=1 or from 0→1 (P command) and the command is set to unsigned (U command), use the positive integer (Unsigned) algorithm to add Sa and Sb and write the result in D .
- Compared with the addition operation of FUN11, the fast addition operation eliminates the overflow and underflow operations and flags, so the program execution time will be faster than the addition operation of FUN11, and the operation result will be the same as the general operation. The result after the computer calculation is the same as the result on the left side of the figure below,
  In addition, the calculation result of the addition operation of FUN11 will be different at the numerical boundary.

R10	HEX	7FFFH	Augend				
R11	HEX	0001H	Addend				
R12	HEX	8000H	fast addition operation resualt	R15	HEX	0000H	addition operation resualt
R10	DEC	32767	Augend				
R11	DEC	1	Addend				
R12	DEC	-32768	fast addition operation resualt	R15	DEC	0	addition operation resualt

FUN224 D P F (+)	Fast ADDITION (Performs addition of the data specified at Sa and Sb and stores the result in D)	FUN224 D P F (+)
Example		

Ladder Diagram	ST
	see code below


```
IF X0 THEN
R2 := R0 + R1;
END_IF
```

Sa	R0	12345
Sb	R1	20425

R0 + R1 = 32770(32767+3)

⇩ X0 = ↑

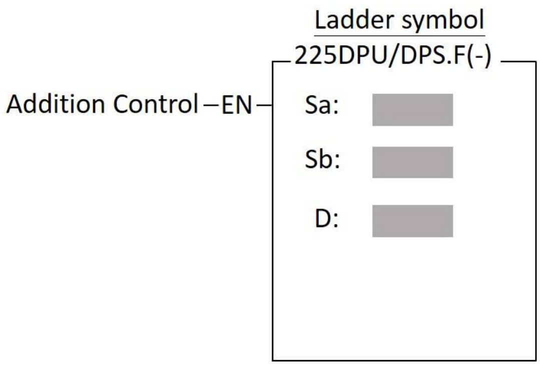
D	R2	2

32767+3>>-32768+2>>-32766

※When adding more than 32767 (0x7FFF), it will become -32768(0x8000)、-32767(0x8001)、-32766(0x8002)、… -1(0xFFFF)、0(0x0000)

6-17 FAST SUBTRACTION F（ - ）

FUN225 D P F（ - ）	Fast SUBTRACTION (Performs subtraction of the data specified at Sa and Sb and stores the result in D)	FUN225 D P F（ - ）
Command Description	Support after UperLogic: v_0.8.517 visions	



Operand

Sa: minuend

Sb: Subtrahend

D: Destination register to store the results of the subtraction

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

Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +/-number	V,Z P0-P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

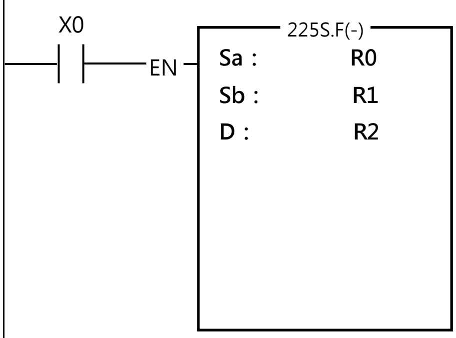
Description	

- When the subtraction control “EN”=1 or from 0→1 (P command) and the command is set to signed (S command), subtract Sa and Sb with the positive and negative number (Sign) algorithm and write the result into D.
- When the subtraction control “EN”=1 or from 0→1 (P command) and the command is set to unsigned (U command), subtract Sa and Sb with a positive integer (Unsigned) algorithm and write the result in D.
- Compared with the subtraction operation of FUN12, the fast subtraction operation eliminates overflow and underflow operations and flags, so the program execution time is faster than the subtraction operation of FUN12, and the operation result will be the same as the general operation. The result calculated by the computer is the same as the result on the left side of the figure on the next page, and it will also be different from the calculation result of the subtraction operation of FUN12 at the numerical boundary.

FUN225 D P F (-)	Fast SUBTRACTION (Performs subtraction of the data specified at Sa and Sb and stores the result in D)	FUN225 D P F (-)

HEX	8000H	minuend				
HEX	0001H	subtrahend				
HEX	7FFFH	fast subtraction operation result	R5	HEX	FFFFH	subtraction operation result
DEC	-32768	minuend				
DEC	1	subtrahend				
DEC	32767	fast subtraction operation result	R5	DEC	-1	subtraction operation result

Example	

Ladder Diagram	ST
	IF X0 THEN R2 := R0 - R1; END_IF

Sa	R0	－5	R0－R1＝－32772(-32768-4)
Sb	R1	32767	

⇩X0＝1

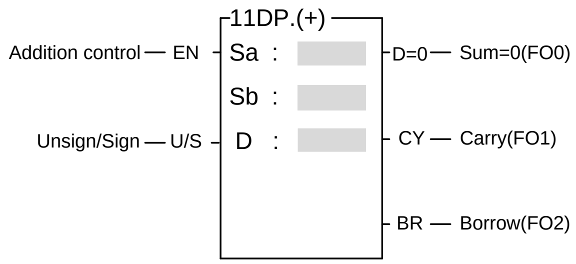
D	R2	32764	－32772>>-32768-4>>32767-3>>32764

※ when less than -32768(0x8000) will becomes to 32767(0x7FFF)、32766(0x7FFE)….0(0x000)

6-18 ADDITION（ + ）

FUN11 D P （ + ）	ADDITION (Performs addition of the data specified at Sa and Sb and stores the result in D)	FUN11 D P （ + ）
Command Description		

Ladder symbol



Operand

Sa: Augend

Sb: Addend

D: Destination register to store the results of the addition

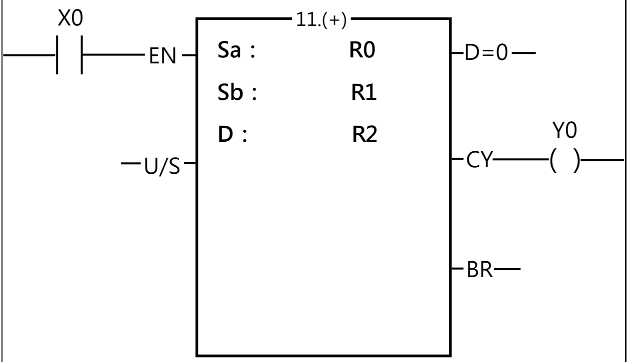
Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect addressing

Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +/-number	V,Z P0-P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

Description	

- Performs the addition of the data specified at Sa and Sb using signed number and writes the results to a specified register D when the add control input "EN" =1 or from 0 to 1 (P instruction) and "U/S" = 0. If the result of addition is equal to 0 then set FO0(D = 0) to 1. If carry occurs (the result exceeds 32767 or 2147483647) then set FO1(CY) to 1. If borrow occurs (adding negative numbers resulting in a sum less than -32768 or -2147483648), then set the FO2(BR) to 1. All the FO statuses are retained until this instruction is executed again and overwritten by a new result.
- Performs the addition of the data specified at Sa and Sb using unsigned number and writes the results to a specified register D when the add control input "EN" =1 or from 0 to 1 (P instruction) and "U/S" = 1. If the result of addition is equal to 0 then set FO0(D = 0) to 1. If carry occurs (the result exceeds 65535 or 4294967295) then set FO1(CY) to 1

FUN11 D P (+)	ADDITION (Performs addition of the data specified at Sa and Sb and stores the result in D)	FUN11 D P (+)
Example		

Ladder Diagram	ST
	

Sa	R0	12345	R0 ＋ R1 ＝ 32770
Sb	R1	20425	

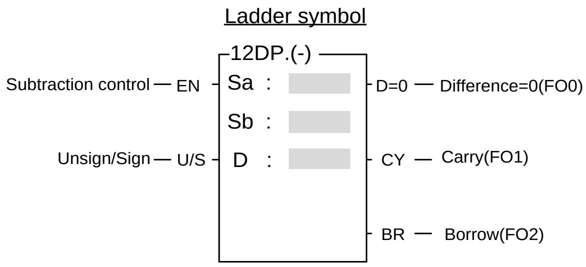
⇩ X0 ＝ ↑

D	R2	2	32768+2=32770

Y0 ＝ 1 (carry 1 represents + 32768)

6-19 SUBTRACTION (-)

FUN12 D P (-)	SUBTRACTION (Performs subtraction of the data specified at Sa and Sb and stores the result in D)	FUN12 D P (-)
Symbol		



Operand

Sa: Minuend

Sb: Subtrahend

D: Destination register to store the results of the subtraction

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

Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +/-number	V,Z P0-P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

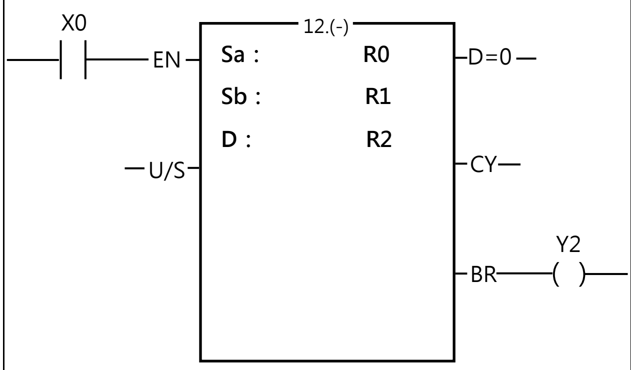
Description	

- Performs the subtraction of the data specified at Sa and Sb using signed number and writes the results to a specified register D when the subtract control input "EN" =1 or from 0 to 1 (P instruction) and "U/S" =0”. If the result of subtraction is equal to 0 then set FO0(D=0) to 1. If carry occurs (subtracting a negative number from a positive number and the result exceeds +32767 or +2147483647), then set FO1(CY) to 1. If borrow occurs (subtracting a positive number from a negative number and the resulted difference is less than -32768 or -2147483648), then set FO2(BR) to 1. All the FO statuses are retained until this instruction is executed again and overwritten by a new result.

FUN12 D P (-)	SUBTRACTION (Performs subtraction of the data specified at Sa and Sb and stores the result in D)	FUN12 D P (-)

- When the subtraction control "EN"=1 or from 0→1 (P command) and "U/S"=1, subtract Sa and Sb with the positive integer (Unsign) algorithm and write the result to D. At the same time, if the difference is 0, FO0 (D=0) is set to 1, and if a borrow occurs (Sa-Sb<0), FO2 (BR) is set to 1.

Example	

Ladder Diagram	ST
	

D	R2	－4	－32768－4＝－32772

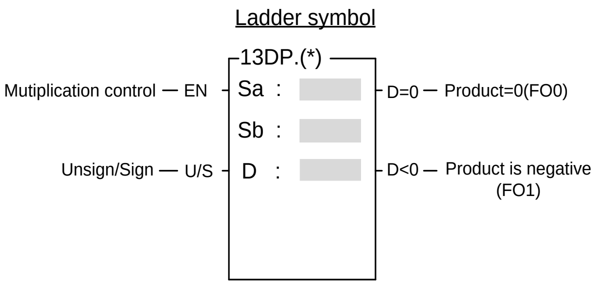
Y2＝1
（borrow 1 represents - 32768）Please refer to section 5.5

Sa	R0	－5	R0－R1＝－32772
Sb	R1	32767	

⇩X0＝1

6-20 MULTIPLICATION (*)

FUN13 D P (*)	MULTIPLICATION (Performs multiplication of the data specified at Sa and Sb and stores the result in D)	FUN13 D P (*)
Command Description		



Operand

Sa: Multiplicand

Sb: Multiplier

D: Destination register to store the results of the multiplication

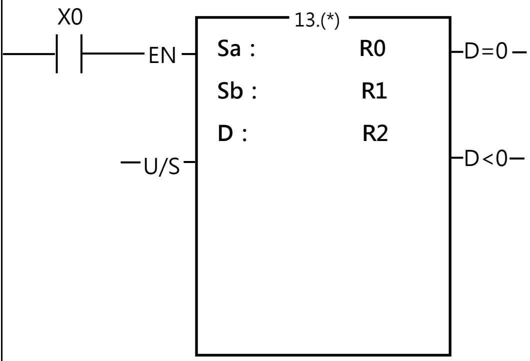
Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect addressing

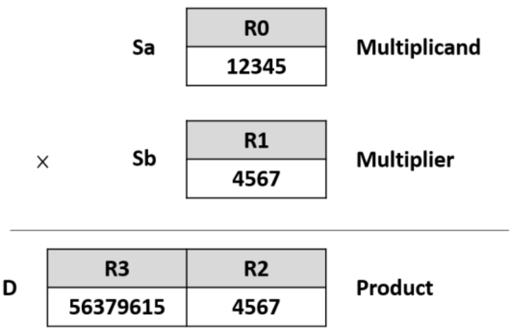
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +/-number	V,Z P0-P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

Description	

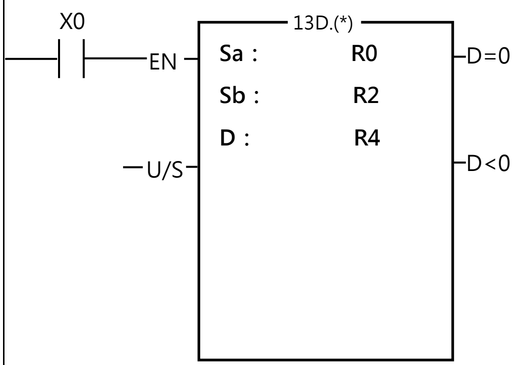
- Performs the multiplication of the data specified at Sa and Sb using the signed number and writes the results to a specified register D when the multiplication control input "EN" =1 or from 0 to 1 (P instruction) and "U/S" = 0. If the product of multiplication is equal to 0 then set FO0(D=0) to 1. If the product is a negative number, then set FO1(D<0) to 1.
- Performs the multiplication of the data specified at Sa and Sb using the unsigned number and writes the results to a specified register D when the multiplication control input "EN" =1 or from 0 to 1 (P instruction) and "U/S" =1 . If the product of multiplication is equal to 0 then set FO0(D=0) to 1.

FUN13 D P (*)	MULTIPLICATION (Performs multiplication of the data specified at Sa and Sb and stores the result in D)	FUN13 D P (*)
Example 1	16-bit multiplication	

Ladder Diagram	ST
	



FUN13 D P (*)	MULTIPLICATION (Performs multiplication of the data specified at Sa and Sb and stores the result in D)	FUN13 D P (*)

Ladder Diagram	ST
	

Sa

R1	**R0**
12345678	

Multiplicand

× **Sb**

R3	**R2**

Multiplier

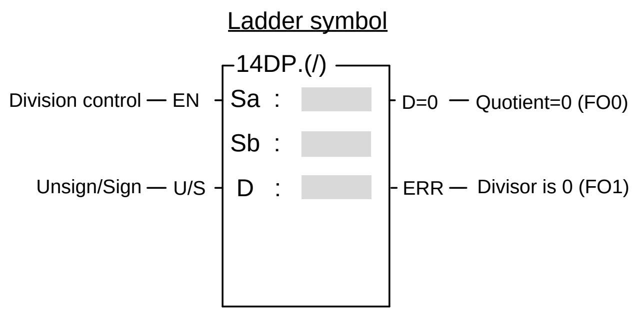
D

R7	**R6**	**R5**	**R4**
5629629168			

Product

6-21 DIVISION (/)

FUN14 D P (/)	DIVISION (Performs division of the data specified at Sa and Sb and stores the result in D)	FUN14 D P (/)
Command Description		



Operand

Sa: Dividend

Sb: Divisor

D: Destination register to store the results of the divison

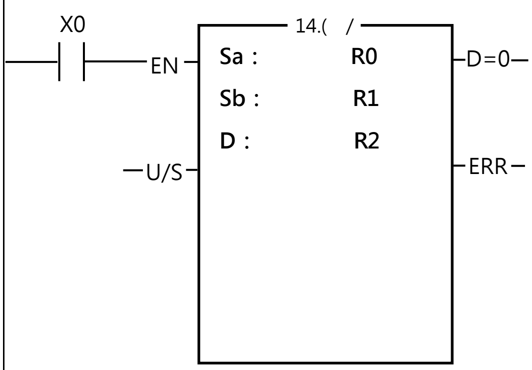
Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect addressing

Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +/-number	V,Z P0-P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

Description	

- Performs the division of the data specified at Sa and Sb using the signed number and writes the results to a specified register D when the multiplication control input "EN" =1 or from 0 to 1 (P instruction) and "U/S" =0. If the quotient of division is equal to 0 then set FO0 to 1. If the divisor Sb=0 then set the error flag FO1 to 1 without executing the instruction.
- Performs the division of the data specified at Sa and Sb using the unsigned number and writes the results to a specified register D when the multiplication control input "EN" =1 or from 0 to 1 (P instruction) and "U/S" =1. If the quotient of division is equal to 0 then set FO0 to 1. If the divisor Sb=0 then set the error flag FO1 to 1 without executing the instruction.

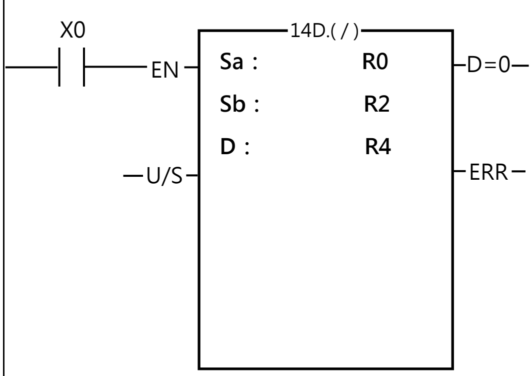
FUN14 D P (/)	DIVISION (Performs division of the data specified at Sa and Sb and stores the result in D)	FUN14 D P (/)
Example 1	16-bit division	

Ladder Diagram	ST
	

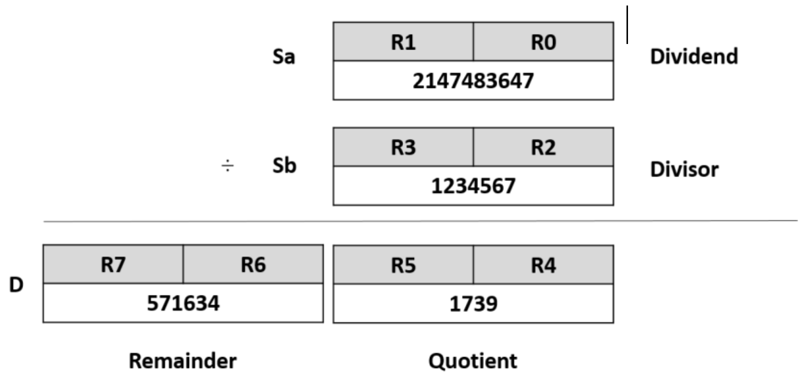
	Sa	R0	Dividend
		256	
÷	Sb	R1	Divisor
		12	

D	R3	R2
	4	21
	Remainder	Quotient

Example 2	32-bit division

Ladder Diagram	ST
	

FUN14 D P (/)	DIVISION (Performs division of the data specified at Sa and Sb and stores the result in D)	FUN14 D P (/)

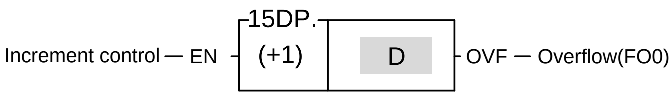


6-22 INCREMENT (+1)

FUN15 D P (+1)	INCREMENT (Adds 1 to the D value)	FUN15 D P (+1)

Command Description

Ladder symbol



Operand

D: The register to be increased

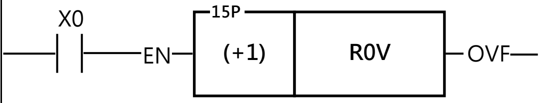
D may combine with V, Z, P0~P9 to serve indirect addressing

Range / Ope-rand	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	WY0 ∣ WY1008	WM0 ∣ WY29584	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	V,Z P0-P9
D	○	○	○	○	○	○	○	○*	○*	○	○

Description

- Adds 1 to the register D when the increment control input "EN" =1 or from 0 to 1 (P instruction). If the value of D is already at the upper limit of positive number 32767 or 2147483647, adding one to this value will change it to the lower limit of negative number -32768 or -2147483648. At the same time, the overflow flag FO0 (OVF) is set to 1.
- Please refer to Section 5.4 for details on overflow.

FUN15 D P (+1)	INCREMENT (Adds 1 to the D value)	FUN15 D P (+1)
Example	16-bit increment register	

Ladder Diagram	ST
 	IF R_TRIG(S:= X0) THEN R0V++ ; END_IF

WHEN V＝100，0+100＝100

D | R100 | 1 |

⇩X0＝↑

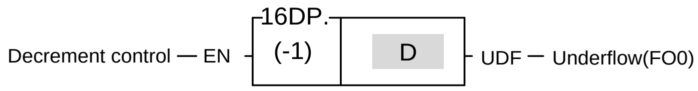
D | R100 | 2 |

6-23 DECREMEMT (-1)

FUN16 D P (-1)	DECREMENT (Subtracts 1 from the D value)	FUN16 D P (-1)

Command Description

Ladder symbol



Operand

D : The register to be decreased

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

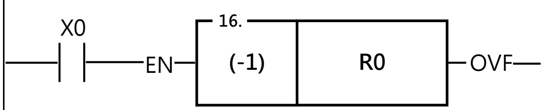
Range / Operand	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	V,Z | P0-P9
D	○	○	○	○	○	○	○	○*	○*	○	○

Description

- Subtracts 1 from the register D when the decrement control input "EN" =1 or from 0 to 1 (P instruction). If the value of D is already at the lower limit of negative number -32768 or -2147483648, subtracting one from this value will change it to the upper limit of positive number 32767 or 2147483647. At the same time, the underflow flag FO0 (UDF) is set to 1.
- Please refer to section 5.4 for detailed description of missing bits.

Example 16-bit decrement register

Ladder Diagram	ST

	`IF X0  THEN`
	`R0-- ;`
	`END_IF`

FUN16 D P (- 1)	DECREMENT (Subtracts 1 from the D value)	FUN16 D P (- 1)

D	R0	0

⇩ X0 = ↑

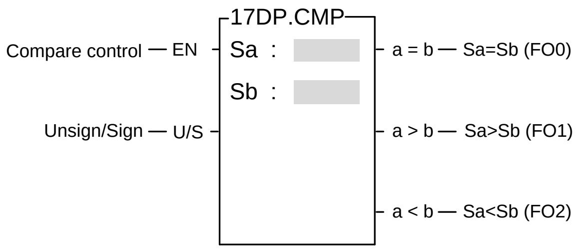
D	R0	− 1

6-24 COMPARE(CMP)

FUN17 D P CMP	COMPARE (Compares the data of Sa and Sb and outputs the results to function Outputs)	FUN17 D P CMP

Command Description

Ladder symbol



Operand

Sa: The register to be compared

Sb: The register to be compared

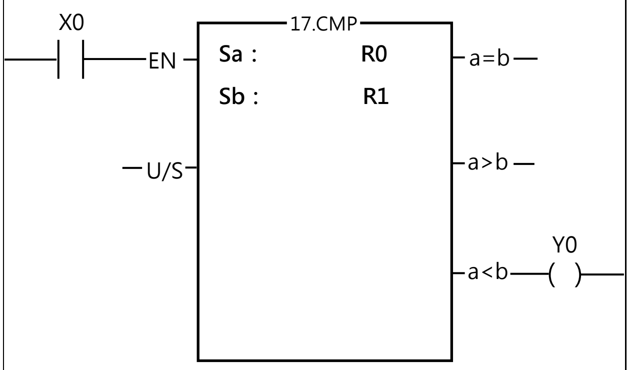
Sa, Sb may combine with V, Z, P0~P9 to serve indirect addressing

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WM29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32 bit +/- number	V、Z P0~P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Description

- Compares the data of Sa and Sb using signed number when the compare control input "EN" =1 or from 0 to 1 (P instruction) and "U/S" =0. If the data of Sa is equal to Sb, then set FO0 to 1. If the data of Sa>Sb, then set FO1 to 1. If the data of Sa<Sb, then set FO2 to 1. If the data of Sa < Sb, then set the FO2 to 1.
- Compares the data of Sa and Sb using unsigned number when the compare control input "EN" =1 or from 0 to 1 (P instruction) and "U/S" =1. If the data of Sa is equal to Sb, then set FO0 to 1. If the data of Sa>Sb, then set FO1 to 1. If the data of Sa<Sb, then set FO2 to 1. If the data of Sa < Sb, then set the FO2 to 1.

Example	Compares the data of 16-bit register

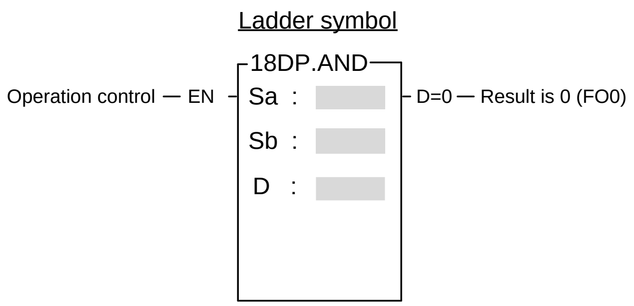
Ladder Diagram	ST
	

- From the above example, we first assume the data of R0 is 1 and R1 is 2, and then compare the data by executing the CMP instruction. The FO0 and FO1 are set to 0 and FO2 (a<b) is set to 1 since a<b.
- If you want to have the compound results, such as≧、≦、< > etc., please send =、< and > results to relay first and then combine the result from the relays.

6-25 LOGICAL AND(AND)

FUN18 D P AND	LOGICAL AND	FUN18 D P AND

Command Description



Operand

Sa: The register to be ANDed

Sb: The register to be ANDed

D: The register to store the result of AND

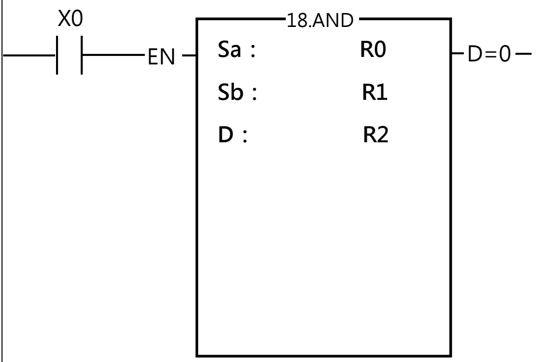
The Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect addressing application

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	WX0 ~ WX1008	WY0 ~ WY1008	WM0 ~ WM29584	WS0 ~ WS3088	T0 ~ T1023	C0 ~ C1279	R0 ~ R34767	R34768 ~ R34895	R35024 ~ R35151	R35280 ~ R43223	R43224 ~ R47319	D0 ~ D11999	16/32 bit +/- number
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○	

Description

- Performs logical AND operation for the data of Sa and Sb when the operation control input "EN" =1 or from 0 to 1 (P instruction). This operation compares the corresponding bits of Sa and Sb (B0~B15 or B0~B31). The bit in the D is set to 1 if both of the corresponding bit data of Sa and Sb is 1. The bit in the D is set to 0 if one of the corresponding bits is 0.

FUN18 D P AND	LOGICAL AND	FUN18 D P AND
Example	Operation of 16-bit logical AND	

Ladder Diagram	ST
	

B15 → B0

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

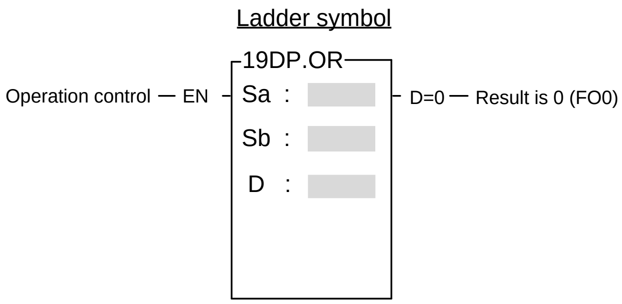
⇩ X0 = ↑

B15 → B0

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

6-26 LOGICAL(OR)

FUN19 D P OR	LOGICAL OR	FUN19 D P OR
Command Description		



Operand

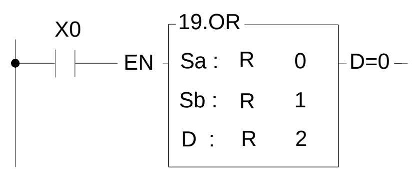
Sa: The register to be ORed
Sb: The register to be ORed
D: The register to store the result of OR
The Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect addressing

Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WM29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +/-number	V,Z P0-P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

Description	Operation of 16-bit logical OR

- Performs logical OR operation for the data of Sa and Sb when the operation control input "EN" =1 or from 0 to 1 (P instruction). This operation compares the corresponding bits of Sa and Sb (B0~B15 or B0~B31). The bit in the D is set to 1 if one of the corresponding of Sa or Sb is 1. The bit in the D is set to 0 if both of the corresponding bits of Sa and Sb is 0.

FUN19 D P OR	LOGICAL OR	FUN19 D P OR
Example	Operation of 16-bit logical OR	

Ladder diagram	ST
 	

B15 B0

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

⇩ X0 = 1

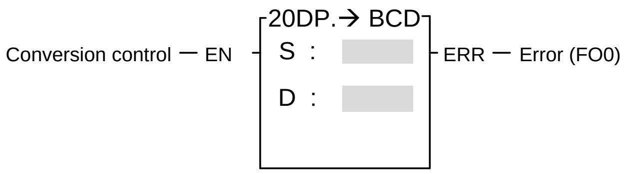
B15 B0

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

6-27 BIN→BCD CONVERSION

FUN 20 D P →BCD	BIN→BCD CONVERSION (Converts BIN data of the device specified at S into BCD and stores the result in D)	FUN 20 D P →BCD
Command Description		

Ladder symbol



Operand

S: The register to be converted

D: The register to store the converted data (BCD code)

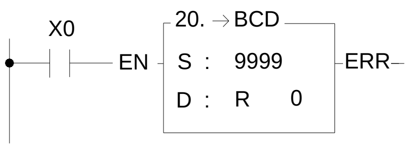
The S, D may combine with V, Z, P0~P9 to serve indirect addressing

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +/-number	V,Z P0-P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

Description	

- FB-PLC uses binary code to store and to execute calculations. If want to send the internal PLC data to the external displays such as seven-segment displays, it is more convenient for us to read the result on screen by converting the BIN data to BCD data. For example, it is more clear for us to read the reading "12" instead of the binary code "1100."
- Converts BIN data of the device specified at S into BCD and writes the result in D when the operation control input "EN" =1 or from 0 to 1 (P instruction). If the data in S is not a BCD value (0~9999 or 0~9999999), then the error flag FO0 is set to 1 and the old data of D are retained.

FUN 20 D P →BCD	BIN→BCD CONVERSION (Converts BIN data of the device specified at S into BCD and stores the result in D)	FUN 20 D P →BCD
Example	16-bit BIN → BCD conversion	

Ladder diagram	ST
	IF X0 THEN ToBCD(S:= 9999, D=> R0); END_IF


```
IF X0 THEN
ToBCD( S:= 9999, D=> R0);
END_IF
```

B15 ↓ … B0 ↓

		B15															B0
S	K	0	0	1	0	0	1	1	1	0	0	0	0	1	1	1	1

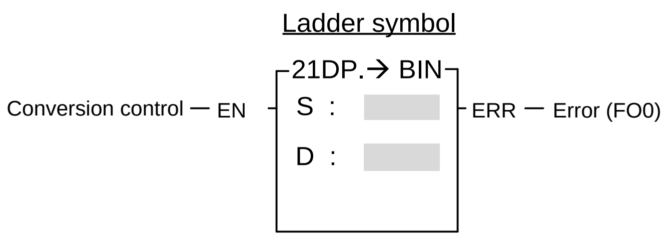
⇩X0 = 1

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

6-28 BCD→BIN CONVERSION

FUN 21 D P →BIN	BCD→BIN CONVERSION (Converts BCD data of the device specified at S into BIN and stores the result in D)	FUN 21 D P →BIN

Symbol



Operand

S: The register to be converted

D: The register to store the converted data (BIN code)

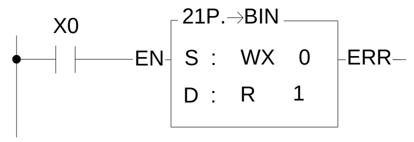
The S, D may combine with V, Z, P0~P9 to serve indirect addressing

Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WY29584	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34895	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	V,Z ∣ P0-P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○	○

Description

- The decimal (BCD) data must be converted to binary (BIN) data first in order for PLC to accept the data which is originally in decimal unit (BCD code) inputted from external device such as digital switch because the BCD data can not be accepted by PLC for its operations.
- Converts BCD data of the device specified at S into BIN and writes the result in D when the operation control input "EN" =1 or from 0 to 1 (P instruction). If the data in S is not in BCD, then the error flag FO0 is set to 1 and the old data of D are retained.
- Constant is converted to BIN automatically when store in program and can not be used as a source operand of this function.

FUN 21 D P →BIN	BCD TO BIN CONVERSION (Converts BCD data of the device specified at S into BIN and stores the result in D)	FUN 21 D P →BIN
Example	16-bit BCD to BIN conversion	

Ladder diagram	ST
	IF R_TRIG(S:= X0) THEN ToBIN(S:= WX0 , D=> R1); END_IF

X15 ↓ (1) … (2) … (3) … (4) ↓ X0

S	WX0	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0

⇩ X0 = 1

B15 ↓ … ↓ B0

D	R1	0	0	0	0	0	1	0	0	1	1	1	1	0	0	1	0

7

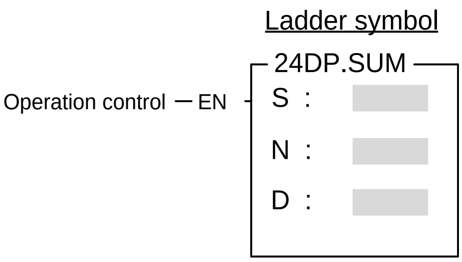
Advanced Function Instructions

7-1 Arithmetical Operation Instructions（FUN24 ~ 33）

7-1-1 Summation of Block Data（SUM）

FUN24 D P SUM	SUM (Summation of block data)	FUN24 D P SUM

Symbol



S: Starting number of source register

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

D: The register which stored the result (summation)

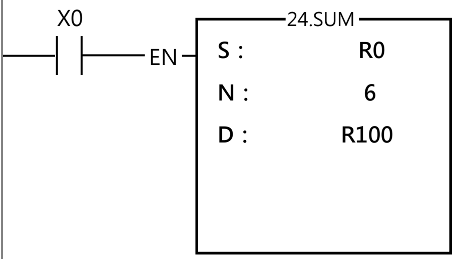
S, N, D, can associate with V, Z, P0~P9 index register to serve the indirect addressing application.

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	1 | 511	V,Z P0-P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

Description

- When operation control “EN”=1 or changes from 0→1 (P instruction), it puts the successive N units of 16bit or 32 bit (D instruction) registers for addition calculation to get the summation, and stores the result into the register which is designated by D.
- When the value of N is 0 or greater than 511, the operation will not be performed.
- Communication port1~2 can be used to serve as a general-purpose ASCII communication interface. If the data error detecting method is Checksum, this instruction can be used to generate the sum value for sending data or ot use this instruction to check if the received data is error or not.

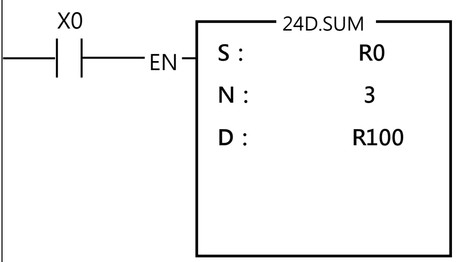
Example 1	When M1 changes from OFF→ON, following instruction will calculates the summation for 16-bit data

Ladder diagram	ST
	IF X0 THEN SUM(S:= R0, N:= 6, D:= R100); END_IF

R0=0030H
R1=0031H
R2=0032H
R3=0033H
R4=0034H
R5=0035H
➔R100=012FH

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

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

Ladder diagram	ST
	IF X0 THEN SUM_D(S:= R0, N:= 6, D:= R100); END_IF

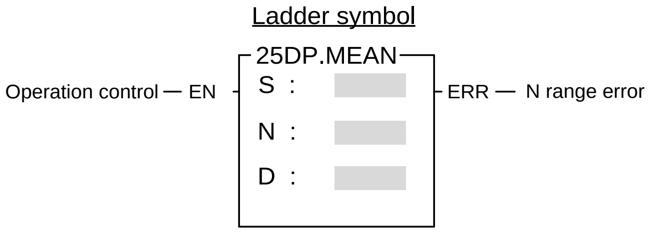
R1 ~ R0=00310030H
R3~R2=00330032H
R5 ~ R4=00410039H
➔ R101 ~ R100=00A5009BH

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

7-1-2 Average of Block Data (MEAN)

FUN25 D P MEAN	MEAN (Average of the block data)	FUN25 D P MEAN

Symbol



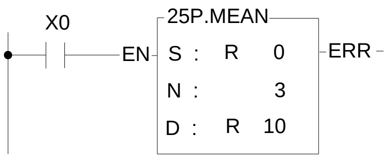
S: Source register number
N: Number of registers to be averaged
(N units of successive registers starting from S)
D: Register number for storing result (mean value)
The S, N, D may combine with V, Z, P0~P9
to serve indirect address application

Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	1 | 511	V,Z P0-P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

Description

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

FUN25 D P MEAN	MEAN (Average of the block data)	FUN25 D P MEAN
Example		

Ladder diagram	ST
	see below


```
IF R_TRIG( S:= X0 ) THEN
MEAN( S:= R0, N:= 3, D:= R10);
END_IF
```

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

S (N = 3)

R0	123
R1	9
R2	788

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

= 306(Rouding off the remainder)

⇩ X0 =

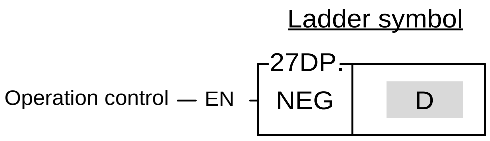
D

R10	306

7-1-3 Take the Negative Value (NEGATION)

FUN27 D P NEG	NEGATION (Take the negative value)	FUN27 D P NEG

Symbol



D : Register to be negated
D may combine with V, Z, P0~P9 to serve indirect address application

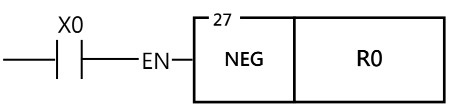
Range / Operand	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	V,Z P0-P9
D	○	○	○	○	○	○	○	○*	○*	○	○

Description

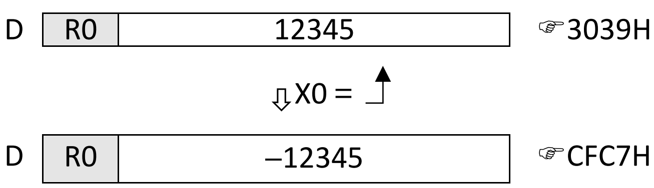
- When operation control "EN" = 1 or from 0 to 1 (P instruction), negate (ie. calculate 2's complement) the value of the content of the register specified by D, and store it back in the original D register.
- If the value of the content of D is negative, then the negation operation will make it positive.

Example

Ladder diagram	ST

	`IF R_TRIG( S:= X0 ) THEN`
	`R0 := - R0;`
	`END_IF`

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



7-1-4 Take the Absolute Value (ABSOLUTE)

FUN28 D P ABS	ABSOLUTE (Take the absolute value)	FUN28 D P ABS

Symbol

Ladder symbol

Operation control — EN — [28DP. ABS | D]

D : Register to be taken absolute value.
D may combine with V, Z, P0~P9 to serve indirect address application.

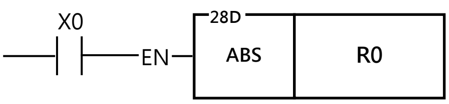
Range / Operand	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	V,Z P0-P9
D	○	○	○	○	○	○	○	○*	○*	○	○

Description

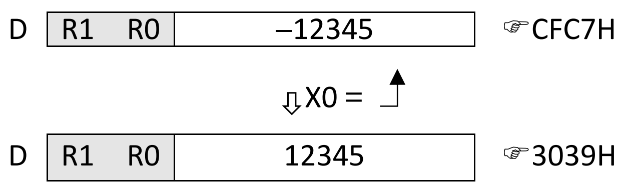
- When operation control "EN" = 1 or from 0 to 1 (P instruction), calculate the absolute value of the content of the register specified by D, and write it back into the original D register.

FUN28 D P ABS	ABSOLUTE (Take the absolute value)	FUN28 D P ABS
Example		

Ladder diagram	ST

	`IF X0 THEN`
	`ABS_D( D:= R0 );`
	`END_IF`

- The instruction at left calculates the absolute value of the DR0 register, and stores it back in DR0(R1, R0).

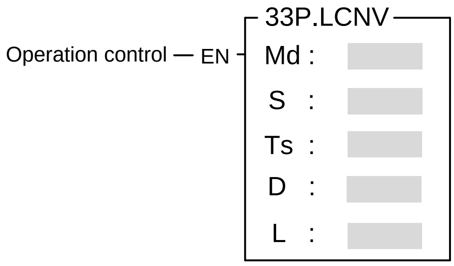


7-1-5 Linear Conversion (LCNV)

FUN33 P LCNV	Linear Conversion (LCNV)	FUN33 P LCNV

Symbol

Ladder symbol



Md: Operation mode, 0~3
S: Starting address of the source data
Ts: Starting address of the parameter table for conversion
D: Starting address to store the result
L: Quantity of conversion entry, 1~64

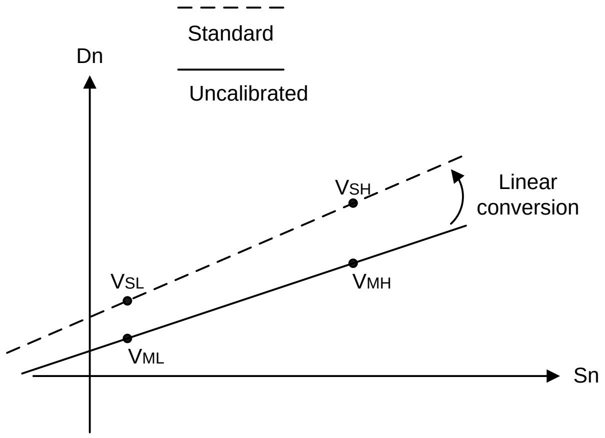
Range / Operand	HR R0 – R34767	IR R34768 – R34895	ROR R43224 – R47319	DR D0 – D11999	K	XR V,Z P0 – P9
Md					0-3	
S	○	○	○	○		○
Ts	○		○	○		○
D	○		○*	○		○
L	○		○	○	1-64	

Description

- When the analog input module being used for the analog measurement, the raw reading value of the analog input can be converted into the engineering range through this instruction for display or for proceeding control operation.
- When using temperature or analog modules for temperature or analog measurement applications, if the temperature or engineering readings measured by the PLC deviate from the results measured by standard thermometers or related standard instruments, this command can be used to make a linear correction as a correction for the actual measured value.
- When execution control "EN"=1 or from 0→1(P instruction), this instruction will perform the linear conversion operation according to the mode selection, where S is the starting address of the source data, Ts is the starting address of the conversion parameter table, D is the starting address to store the converted result, and L is the quantity of conversion entry.
- There are two expressions to meet the suitable application:

Expression 1: Two points calibration method

Fill the conversion parameter table with the low value of measurement(VML), high value of measurement (VMH), and the corresponding low value of standard (VSL), high value of standard (VSH); the converted result (Dn) will be generated from the source data(Sn) through the formula shown below:



A = (VSL - VSH / VML - VMH)×10000

B = VSL - (VML×A / 10000)

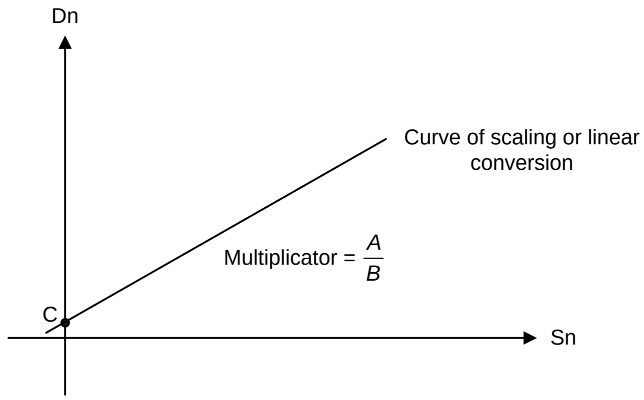
Dn = (Sn×A / 10000) + B

- The range of operands VSL, VSH, VML, VMH, Sn and Dn are between -32768 ~ 32767.
- For analog input scaling, where:
  VML=Minmum of analog input
  VMH=Maximum of analog input
  VSL=Minmum of engineering range
  VSH=Maximum of engineering range

FUN33 P LCNV	Linear Conversion (LCNV)	FUN33 P LCNV

Expression 2 : Multiplicator + Offset method

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



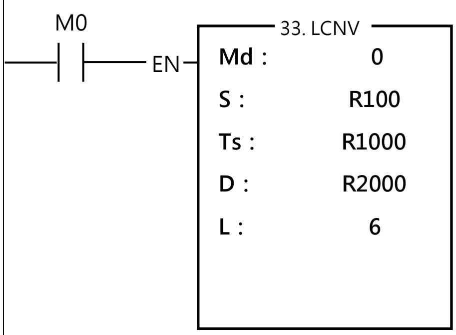
Dn =[(Sn×A) / B] + C
The range of each operand as below:
A = 1~65535
B = 1~65535
C = -32768~32767
Sn = 0~65535
Dn = -32768~32767

Description of operation mode

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

Example 1	Mode 0 of linear conversion

Ladder diagram	ST




```
LCNV( EN:= M0, Md:= 0, S:= R100,
Ts:= R1000, D=> R2000, L:= 6);
```

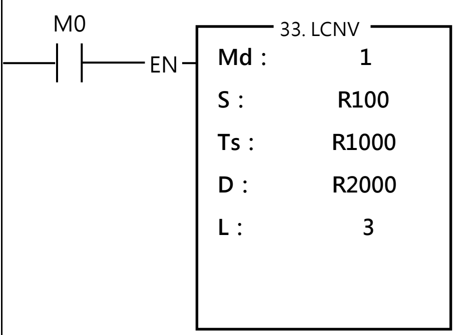
- When M0 = 1, it will perform the mode 0 operation of linear conversion, where R100 is the starting address of the source data, R1000 is the starting address of the table of the conversion parameters VML, VMH, VSL, VSH, the quantity is 6, and R2000~R2005 will store the converted results.

FUN33 P LCNV	Linear Conversion (LCNV)	FUN33 P LCNV

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

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

Example 2 Mode 1 of linear conversion

Ladder diagram	ST
	`LCNV( EN:= M0, Md:= 1, S:= R100, Ts:= R1000, D=> R2000, L:= 3);`

Description:

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

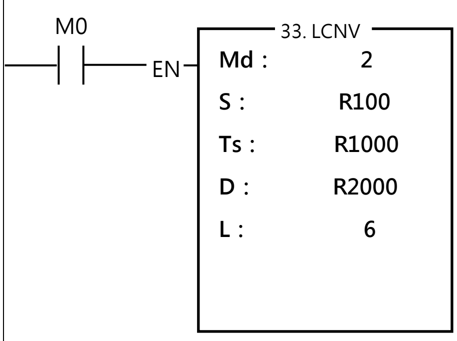
FUN33 P LCNV	Linear Conversion (LCNV)	FUN33 P LCNV

Ts

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

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

FUN33 P LCNV	Linear Conversion (LCNV)	FUN33 P LCNV
Example 3	Mode 2 of linear conversion	

Ladder diagram	ST
	`LCNV( EN:= M0, Md:= 2, S:= R100, Ts:= R1000, D=> R2000, L:= 6);`

Description:

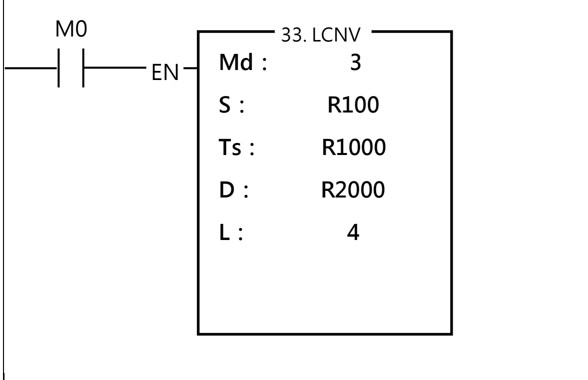
When M0 = 1, it will perform the mode 2 operation of linear conversion, where R100 is the starting address of the source data, R1000 is the starting address of the table of the conversion parameters A, B, C, the quantity is 6, and R2000~R2005 will store the converted results.

FUN33 P LCNV	Linear Conversion (LCNV)	FUN33 P LCNV

	Ts	
R1000	985	A
R1001	1000	B
R1002	20	C

	S			D
R100	1000		R2000	1005
R101	2345		R2001	2329
R102	3560	⇨	R2002	3526
R103	401		R2003	415
R104	568		R2004	579
R105	2680		R2005	2659

Example 4	Mode 3 of linear conversion

Ladder diagram	ST
	LCNV(EN:= M0, Md:= 3, S:= R100, Ts:= R1000, D=> R2000, L:= 4);

Description:

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

FUN33 P LCNV	Linear Conversion (LCNV)	FUN33 P LCNV

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

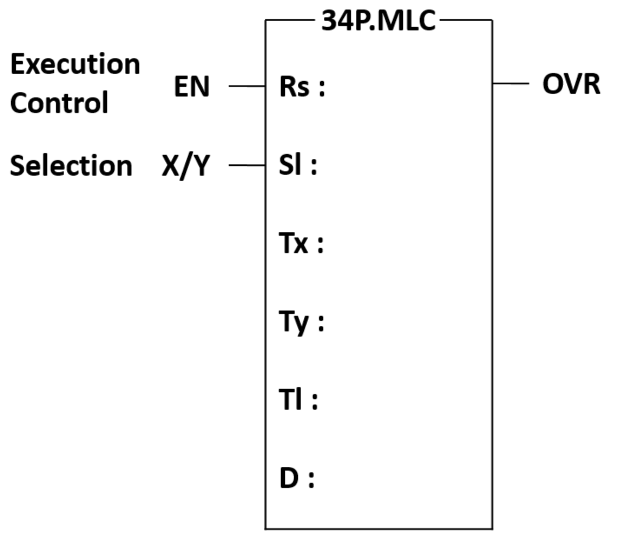
	S
R100	8192
R101	16383
R102	8190
R103	0

⇨

	D
R2000	2500
R2001	10000
R2002	900
R2003	-100

7-1-6 Multiple Linear Conversion (MLC)

FUN34P MLC	Multiple Linear Conversion (MLC)	FUN34P MLC
Symbol		



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

Range / Operand	HR	IR	OR	SR	ROR	DR	K
	R0 | R34767	R34768 | R34895	R35204 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	
Rs	○	○	○	○	○	○	
Sl	○	○	○	○	○	○	1-64
Tx	○				○	○	
Ty	○				○*	○	
Tl	○	○	○	○	○	○	2-255
D	○				○	○	

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

- When the analog input module being used for the analog measurement, the raw reading value of the analog input can be converted into the engineering range through this instruction for display or for proceeding control operation.
- When using temperature or analog modules for temperature or analog measurement applications, if the temperature or engineering readings measured by the PLC deviate from the results measured by standard thermometers or related standard instruments, this command can be used to Make a linear correction as a correction for the actual measured value.
- When execution control "EN"=1or from 0→1(P instruction), this instruction will perform the multiple linear conversion operation according to the selection of X/Y input; where Rs is the starting address of the source data, Sl is the quantity of source data for conversion, Tx is the starting address of X conversion parameter table, Ty is the starting address of Y conversion parameter table, Tl is the quantity of X/Y table, D is the starting address to store the converted result.
- When executing and selection X/Y=0, it will compare the source data with the entities of Tx table to find the corresponding location in Tx table (The entities in Tx table must be in ascending sequence), and then calculate the linear conversion according to the located section of Tx and Ty table; When executing and selection X/Y=1, it will compare the source data with the entities of Ty table to find the corresponding location in Ty table (The entities in Ty table can either be in ascending or descending sequence), and then calculate the linear conversion according to the located section of Ty and Tx table.
- When the source data is out of all entities of table, OVR=1.
- It wouldn’t execute this instruction if illegal SI or Tl.

Expression:

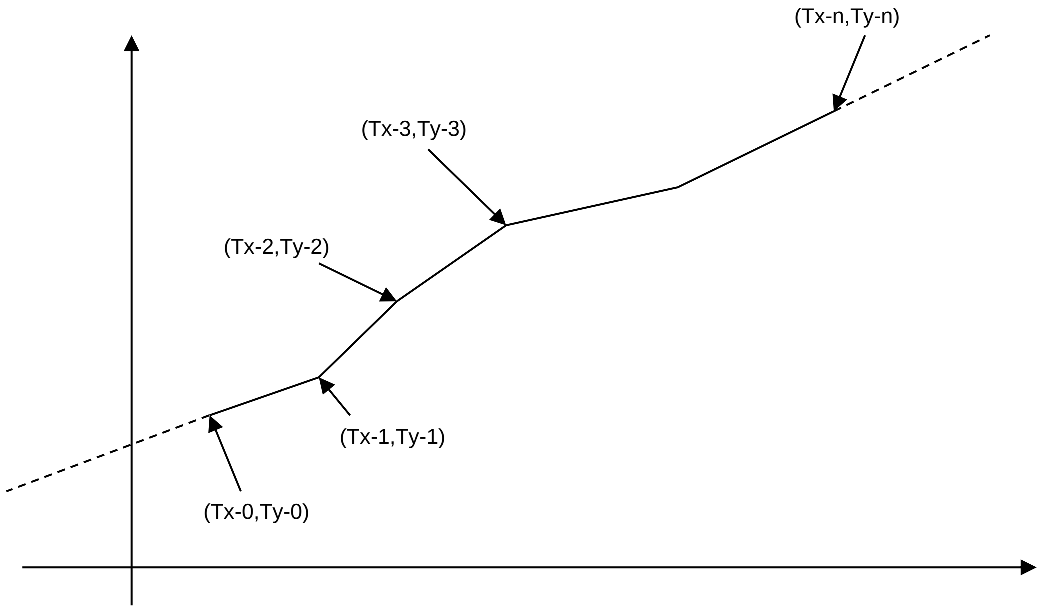
The entities of Tx conversion parameter table must be in ascending sequence to have correct linear conversion; the entities of Ty conversion parameter table can either be in ascending or descending sequence. When executing this instruction, it will search the located section by comparing entities of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

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

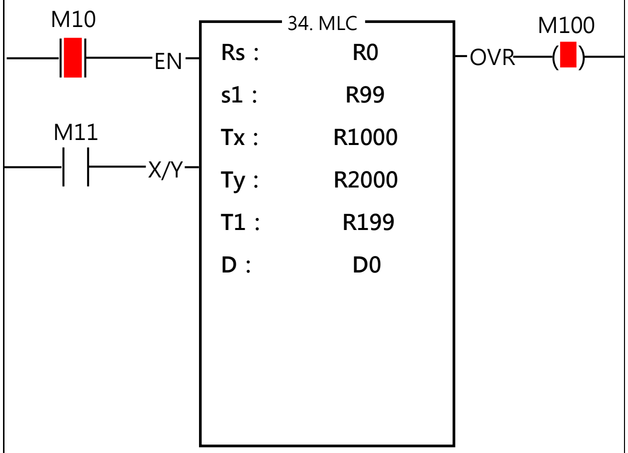
Vx = (Vy - Ty_n) × (Tx_n+1 - Tx_n / Ty_n+1 - Ty_n) + Tx_n if X/Y=1

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

Figure of multiple linear conversion:



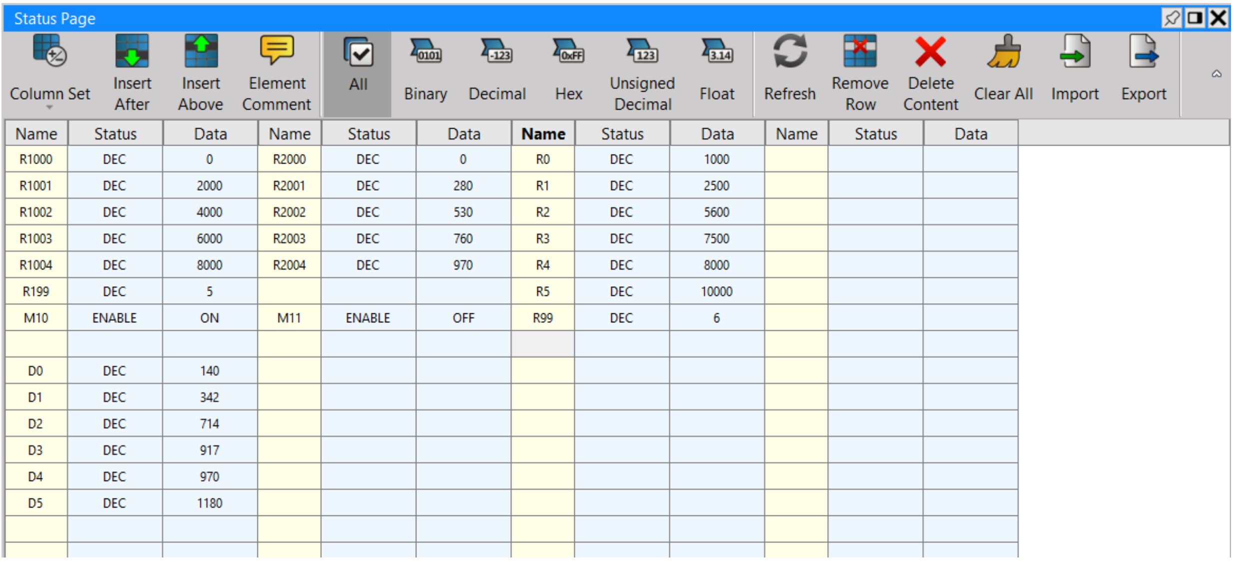
FUN34 P MLC	Multiple Linear Conversion (MLC)	FUN34 P MLC
Example 1		

Ladder diagram	ST
	MLC(EN:= M10, XY:= M11, Rs:= R0, S1:= R99, Tx:= R1000, Ty:= R2000, T1:= R199, D:= D0, OVR=> M100);

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

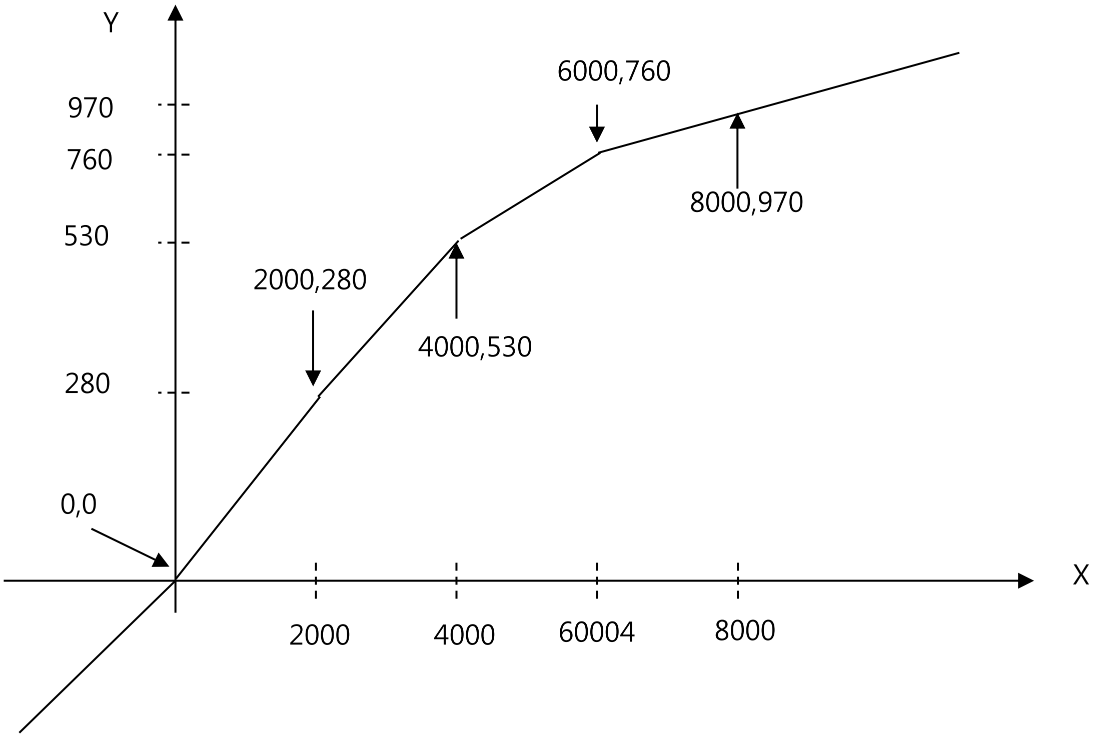
Description:

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

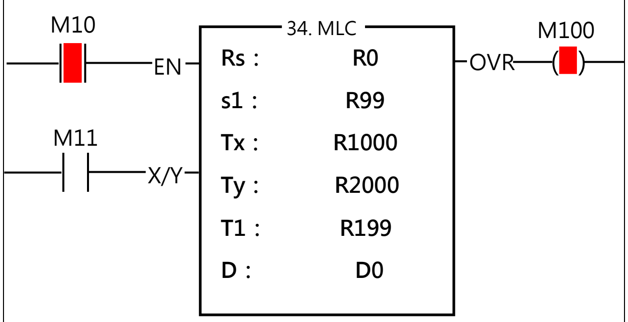


Name	Status	Data	Name	Status	Data	Name	Status	Data	Name	Status	Data
R1000	DEC	0	R2000	DEC	0	R0	DEC	1000			
R1001	DEC	2000	R2001	DEC	280	R1	DEC	2500			
R1002	DEC	4000	R2002	DEC	530	R2	DEC	5600			
R1003	DEC	6000	R2003	DEC	760	R3	DEC	7500			
R1004	DEC	8000	R2004	DEC	970	R4	DEC	8000			
R199	DEC	5				R5	DEC	10000			
M10	ENABLE	ON	M11	ENABLE	OFF	R99	DEC	6			
D0	DEC	140									
D1	DEC	342									
D2	DEC	714									
D3	DEC	917									
D4	DEC	970									
D5	DEC	1180									

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

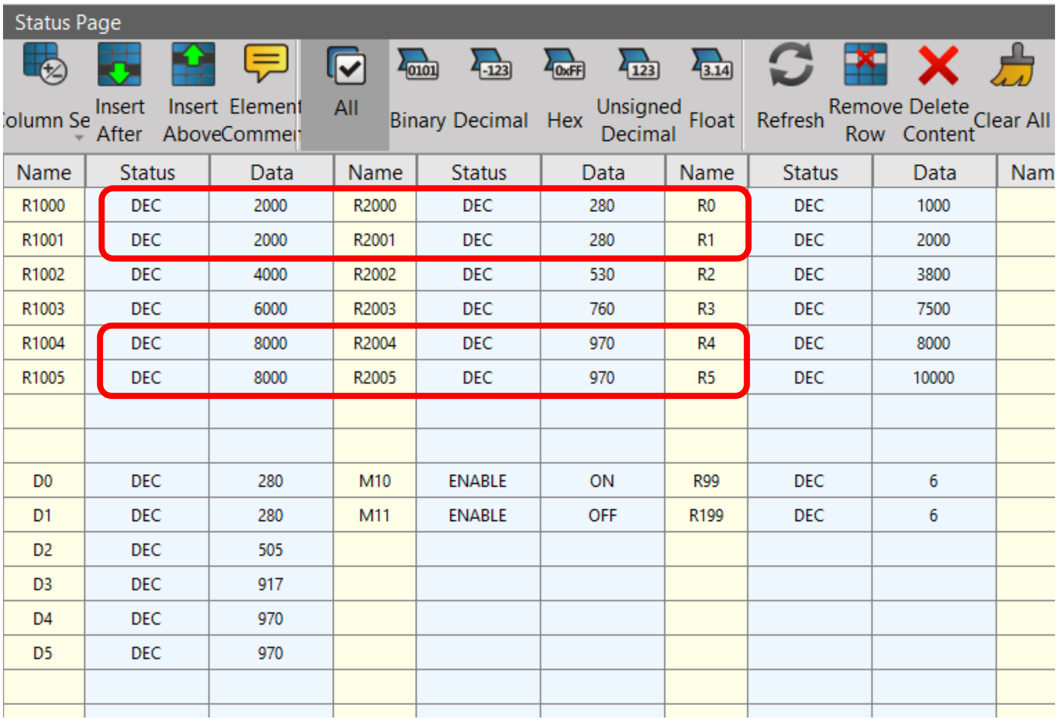


Example 2

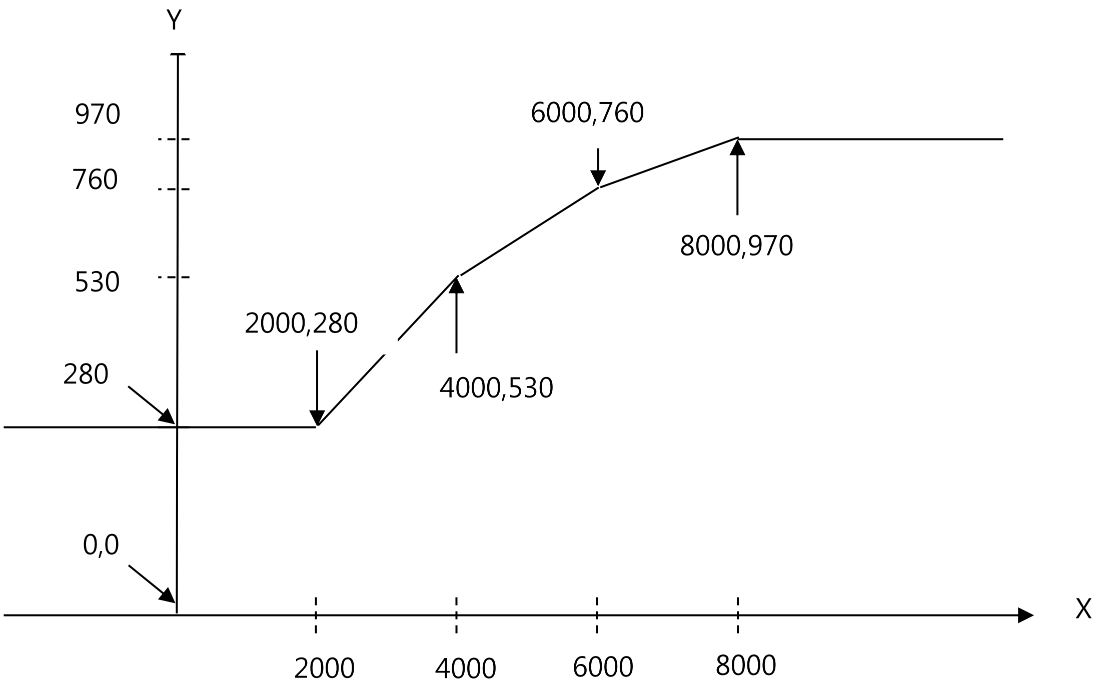
Ladder diagram	ST
	MLC(EN:= M10, XY:= M11, Rs:= R0, Sl:= R99, Tx:= R1000, Ty:= R2000, Tl:= R199, D:= D0, OVR=> M100);

Description:
When M10=1, M11=0, take R0 as the starting source data and R99 as the source data length, according to the Tx conversion table starting from R1000 and the Ty conversion table starting from R2000, and R199 as the conversion table Length, perform 5-segment linear conversion operation on source data such as R0~R5, and store the conversion results in temporary registers D0~D5. In this example, when the value of the source data is less than or equal to 2000, the corresponding value is 280; when the value of the source data is greater than or equal to 8000, the corresponding value is 970.

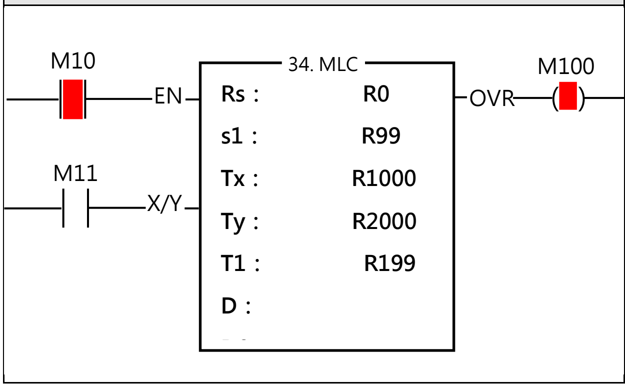
FUN34 P MLC	Multiple Linear Conversion (MLC)	FUN34 P MLC



Name	Status	Data	Name	Status	Data	Name	Status	Data	Nam
R1000	DEC	2000	R2000	DEC	280	R0	DEC	1000	
R1001	DEC	2000	R2001	DEC	280	R1	DEC	2000	
R1002	DEC	4000	R2002	DEC	530	R2	DEC	3800	
R1003	DEC	6000	R2003	DEC	760	R3	DEC	7500	
R1004	DEC	8000	R2004	DEC	970	R4	DEC	8000	
R1005	DEC	8000	R2005	DEC	970	R5	DEC	10000	
D0	DEC	280	M10	ENABLE	ON	R99	DEC	6	
D1	DEC	280	M11	ENABLE	OFF	R199	DEC	6	
D2	DEC	505							
D3	DEC	917							
D4	DEC	970							
D5	DEC	970							



FUN34 P MLC	Multiple Linear Conversion (MLC)	FUN34 P MLC
Example 3		

Ladder diagram	ST
	see below


```
MLC( EN:= M10,
     XY:= M11,
     Rs:= R0,
     S1:= R99,
     Tx:= R1000,
     Ty:= R2000,
     Tl:= R199,
     D:= D0,
```

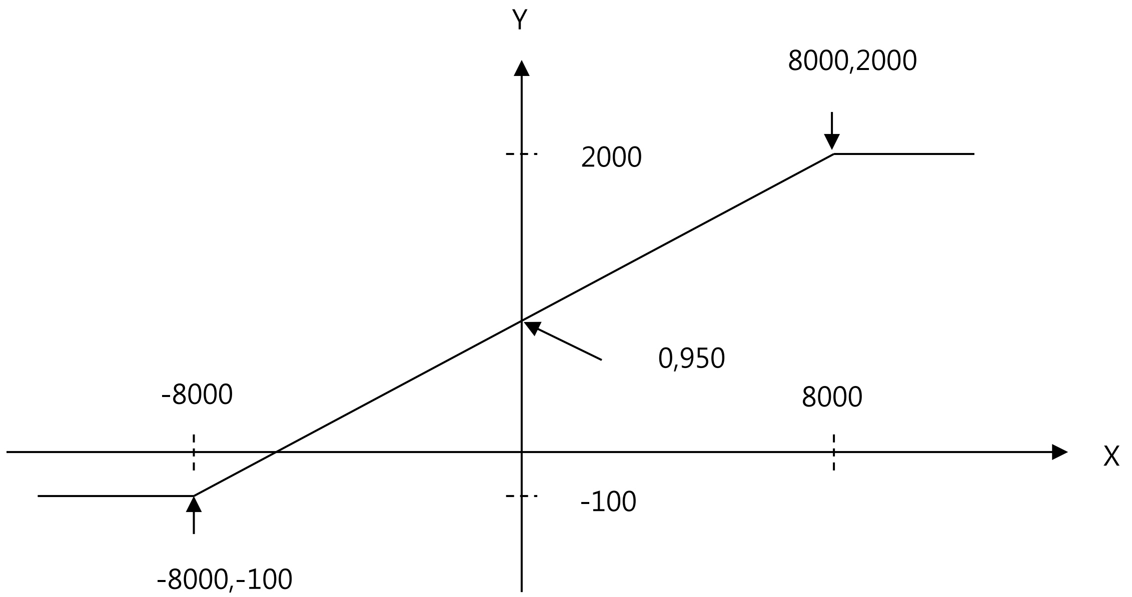
Status Page

Name	Status	Data	Name	Status	Data	Name	Status	Data	Na
R1000	DEC	-8000	R2000	DEC	-100	R0	DEC	-8100	
R1001	DEC	-8000	R2001	DEC	-100	R1	DEC	0	
R1002	DEC	8000	R2002	DEC	2000	R2	DEC	4000	
R1003	DEC	8000	R2003	DEC	2000	R3	DEC	8100	
						R4	DEC	-10000	
						R5	DEC	10000	
D0	DEC	-100	M10	ENABLE	ON	R99	DEC	6	
D1	DEC	950	M11	ENABLE	OFF	R199	DEC	4	
D2	DEC	1474							
D3	DEC	2000							
D4	DEC	-100							
D5	DEC	2000							

Description:

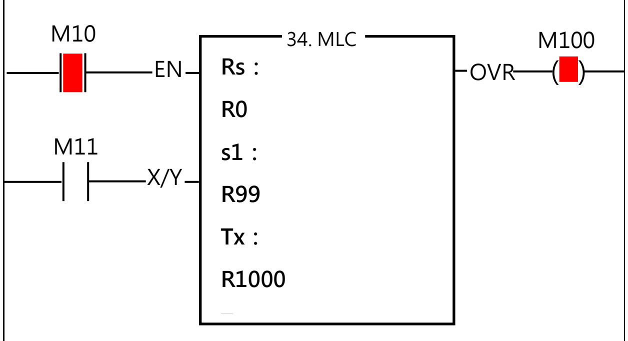
When M10=1, M11=0, R0 is the starting address of source data, R99 is the quantity of source data, R1000 is the starting address of Tx conversion parameter table, R2000 is the starting address of Ty conversion parameter table, R199 is the quantity of table; the source data R0~R5 will be calculated the linear conversion according to Tx and Ty table between three sections, then store the results into D0~D5. T In this example, when the value of the source data is -8000~8000, the corresponding

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



Example 4	

Ladder diagram	ST




```
MLC( EN:= M10,
     XY:= M11,
     Rs:= R0,
     S1:= R99,
     Tx:= R1000,
     Ty:= R2000,
     Tl:= R199,
     D:= D0,
     OVR=> M100);
```

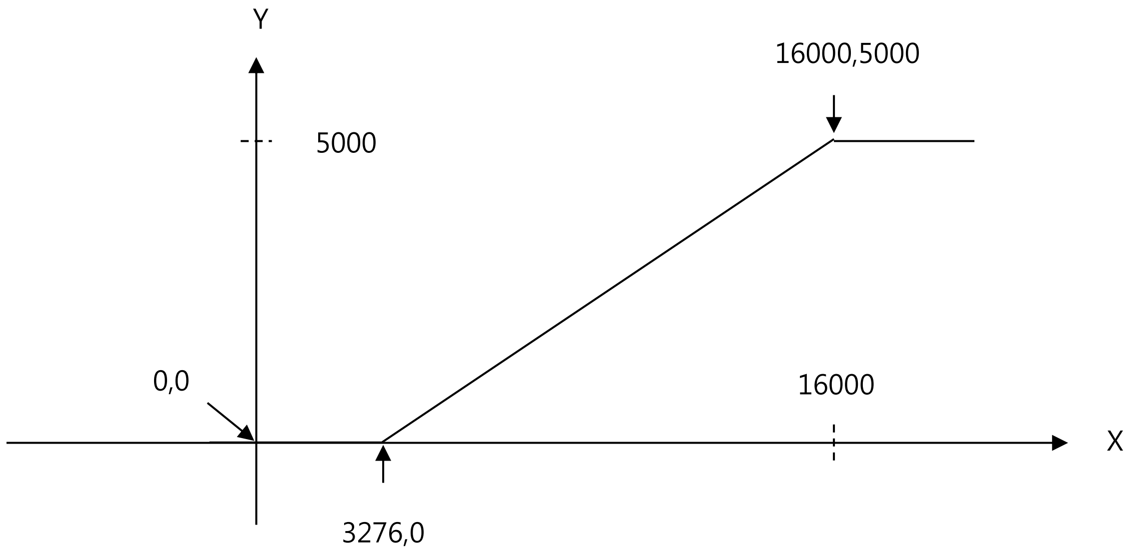
Status Page

Name	Status	Data	Name	Status	Data	Name	Status	**Data**	Nar
R1000	DEC	3276	R2000	DEC	0	R0	DEC	0	
R1001	DEC	3276	R2001	DEC	0	R1	DEC	3276	
R1002	DEC	16000	R2002	DEC	5000	R2	DEC	4095	
R1003	DEC	16000	R2003	DEC	5000	R3	DEC	9638	
						R4	DEC	16000	
						R5	DEC	16380	
D0	DEC	0	M10	ENABLE	ON	R99	DEC	6	
D1	DEC	0	M11	ENABLE	OFF	R199	DEC	4	
D2	DEC	321							
D3	DEC	2500							
D4	DEC	5000							
D5	DEC	5000							

Description:

When M10=1, M11=0, R0 is the starting address of source data, R99 is the quantity of source data, R1000 is the starting address of Tx conversion parameter table, R2000 is the starting address of Ty conversion parameter table, R199 is the quantity of table; the source data R0~R5 will be calculated the linear conversion according to Tx and Ty table between three sections, then store the results into D0~D5.T In this example, when the value of the source data is 3276~16000, the corresponding value is 0~5000 according to the linear conversion shown in the figure below; when the value of the source data is ≧16000, the corresponding value is 5000; all are 0.

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

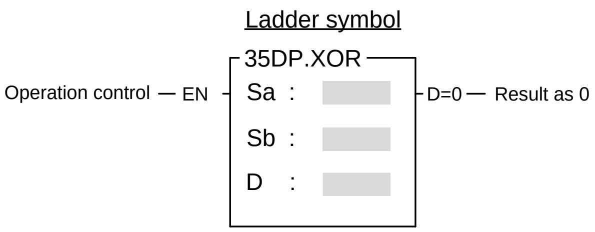


7-2 ogical Operation Instructions (FUN35 ~ 36)

7-2-1 EXCLUSIVE OR (XOR)

FUN35 D P XOR	EXCLUSIVE OR (XOR)	FUN35 D P XOR

Symbol



Sa: Source data a for exclusive or operation
Sb: Source data b for exclusive or operation
D: Register storing XOR results
Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16-bit +/-number	V,Z P0-P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

Description

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

FUN35 D P XOR	EXCLUSIVE OR (XR)	FUN35 D P XOR
Example		

Ladder diagram	ST
	

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

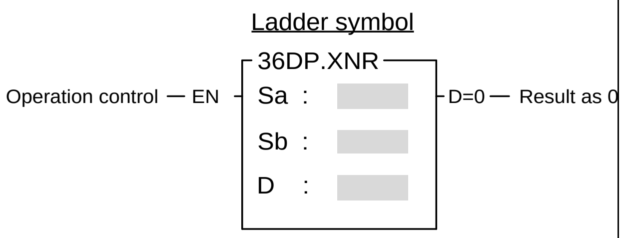
Sa	R0	1	0	1	1	1	0	1	1	0	1	1	0	1	1	0	1
Sb	R1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0

⇩X0 = ↑

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

7-2-2 EXCLUSIVE NOR (XNR)

FUN36 D P XNR	EXCLUSIVE NOR (XNR)	FUN36 D P XNR
Symbol		



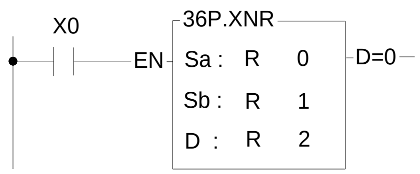
Sa: Data a for XNR operation
Sb: Data b for XNR operation
D: Register storing XNR results
Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WY29584	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34895	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	16-bit +/-number	V,Z P0-P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

Description	

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

FUN36 D P XNR	EXCLUSIVE NOR (XNR)	FUN36 D P XNR
Example		

Ladder diagram	ST
	

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

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

⇩ X0 =

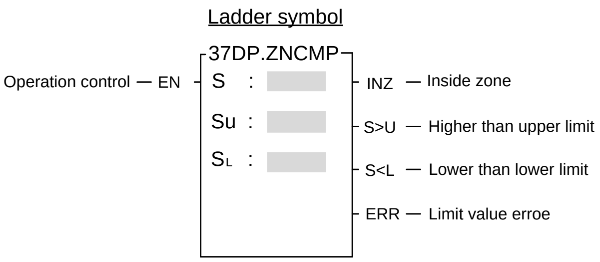
D	R2	1	0	1	0	1	0	1	0	0	0	1	1	0	1	0	0

7-3 Comparison Instructions (FUN37)

7-3-1 ZONE COMPARE (ZNCMP)

FUN37 D P ZNCMP	ZONE COMPARE	FUN37 D P ZNCMP

Symbol



S: Register for zone comparison
SU: The upper limit value
SL: The lower limit value
S, SU, SL may combine with V, Z, P0~P9 to serve indirect address application

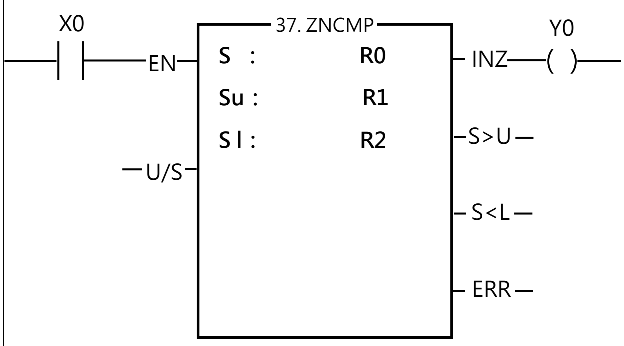
Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16-bit +/-number	V,Z P0-P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
SU	○	○	○	○	○	○	○	○	○	○	○	○	○	○
SL	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Description

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

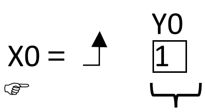
FUN37 D P ZNCMP	ZONE COMPARE	FUN37 D P ZNCMP
Example		

- The instruction compares the value of R0 with the upper and lower limit zones formed by R1 and R2. If the values of R0~R2 are as shown in the diagram at bottom left, then the result can then be obtained as shown in the diagram below.
  If want to get the status of out side the zone, you can use OUT NOT Y0.

Ladder diagram	ST
	

S	R0	200	
S_U	R1	300	(Upper limit value)
S_L	R2	100	(Lower limit value)

Before-execution



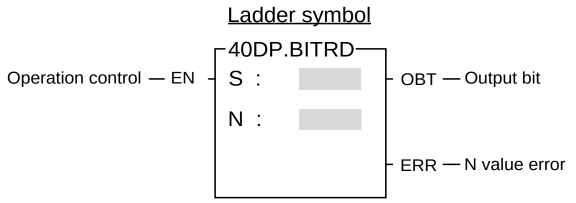
Results of execution

7-4 Data Movement Instructions（FUN40 ~ 50）

7-4-1 BIT READ (BITRD)

FUN40 D P BITRD	BIT READ	FUN40 D P BITRD

Symbol



S: Source data to be read

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

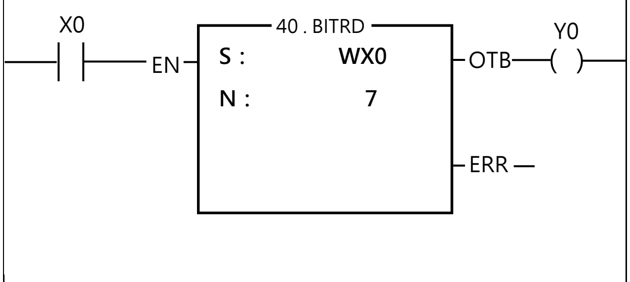
S, N may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +/-number	V,Z P0-P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D	○	○	○	○	○	○	○	○	○	○	○	○	0-31	○

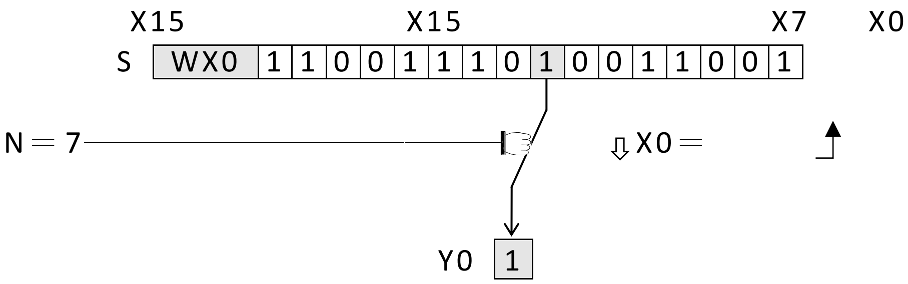
Description

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

FUN40 D P BITRD	BIT READ	FUN40 D P BITRD
Example		

Ladder diagram	ST
	

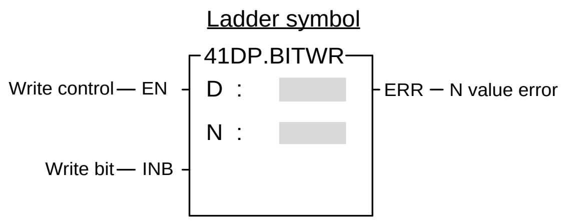
The instruction reads the 7th bit (X7) status from WX0 (X0~X15) and output to Y0. The results are as follows:



7-4-2 BIT WRITE (BITWR)

FUN41 D P BITWR	BIT WRITE	FUN41 D P BITWR

Symbol



D: Register for bit write

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

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

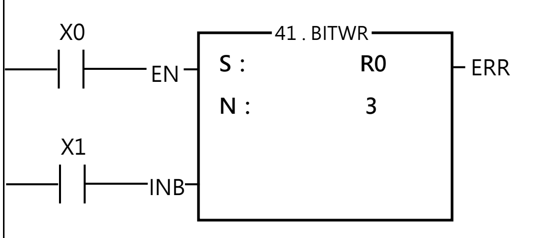
Range / Ope-rand	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	0 0 | OR | 15 31	V,Z P0-P9
D	○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○

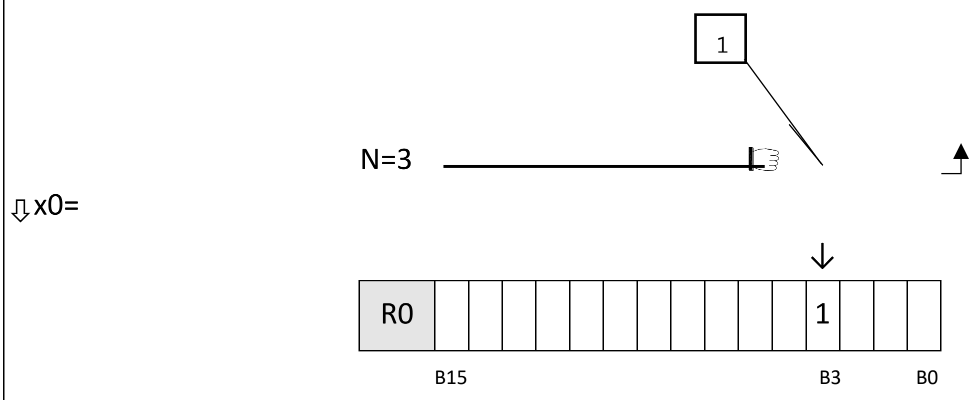
Description

- When write control "EN" = 1 or changes from 0 to 1 (P instruction), will write the write bit (INB) into the Nth bit of register D.

FUN41 D P BITWR	BIT WRITE	FUN41 D P BITWR

- When the operand is 16-bit, the effective range of N is 0~15. For 32-bit (D instruction) operand it is 0~31. N beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.
- The instruction writes the status of the write bit INB into B3 of R0. Assuming X1 = 1, the result will be as follows:

Ladder diagram	ST
	

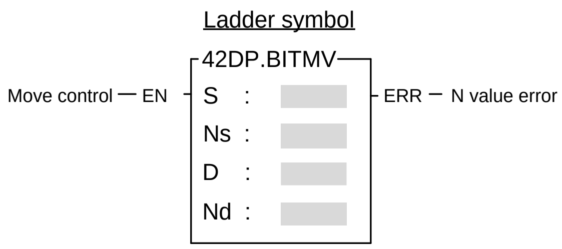


Bits other than B3 remain unchanged

7-4-3 BIT MOVE (BITMV)

FUN42 D P BITMV	BIT MOVE	FUN42 D P BITMV

Symbol



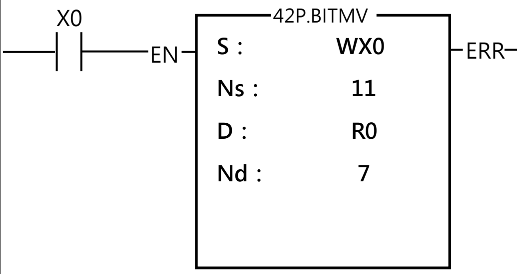
S: Source data to be moved
Ns: Assign Ns bit within S as source bit
D: Destination register to be moved
Nd: Assign Nd bit within D as target bit
S, Ns, D, Nd may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +/-number	V,Z P0-P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ns	○	○	○	○	○	○	○	○	○	○	○	○	0-31	○
D		○	○	○	○	○	○		○	○*	○*	○		○
Nd	○	○	○	○	○	○	○	○	○	○	○	○	0-31	○

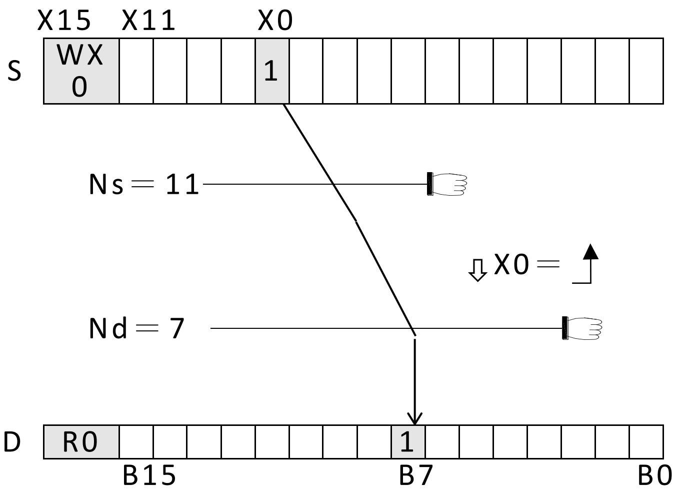
Description

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

FUN42 D P BITMV	BIT MOVE	FUN42 D P BITMV
Example		

Ladder diagram	ST
	

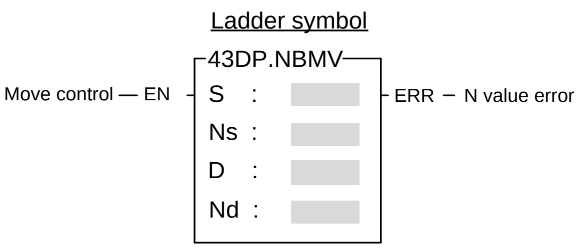
- The instruction at left moves the status of B11 (X11) within S into the B7 position within D. Except bit B7, other bits within D does not change.



7-4-4 NIBBLE MOVE (NBMV)

FUN43 D P NBMV	NIBBLE MOVE	FUN43 D P NBMV

Symbol



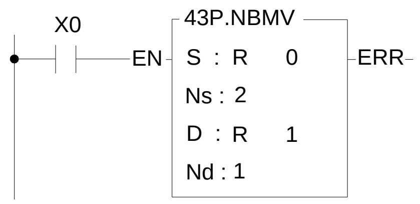
S: Source data to be moved
Ns: Assign Ns nibble within S as source nibble
D: Destination register to be moved
Nd: Assign Nd nibble within D as target nibble
S, Ns, D, Nd may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WY29584	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34895	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	16/32-bit +/-number	V,Z P0-P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ns	○	○	○	○	○	○	○	○	○	○	○	○	0-7	○
D		○	○	○	○	○	○		○	○*	○*	○		○
Nd	○	○	○	○	○	○	○	○	○	○	○	○	0-7	○

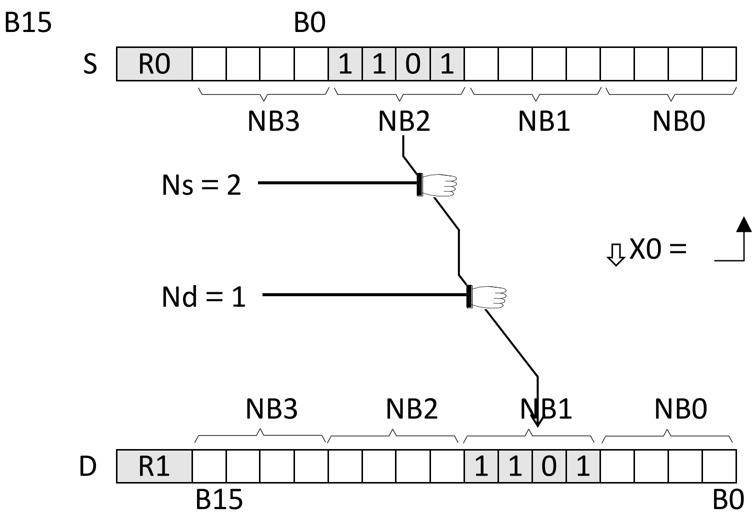
Description

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

FUN43 D P NBMV	NIBBLE MOVE	FUN43 D P NBMV
Example		

Ladder diagram	ST
	

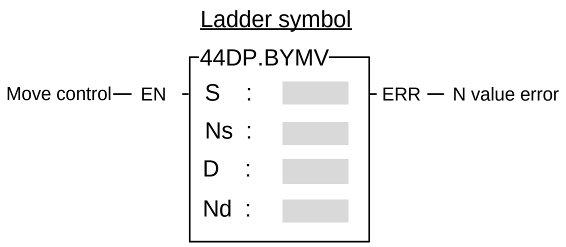
- The instruction moves the third nibble NB2 (B8~B11) within S to the first nibble NB1 (B4~B7) within D. Other nibbles within D remain unchanged.



7-4-5 BYTE MOVE (BYMV)

FUN44 D P BYMV	BYTE MOVE	FUN44 D P BYMV

Symbol



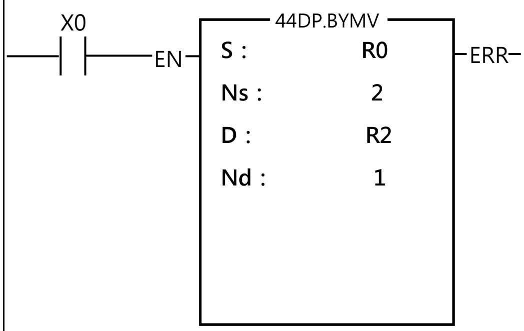
S: Source data to be moved
Ns: Assign Ns byte within S as source byte
D: Destination register to be moved
Nd: Assign Nd byte within D as target byte
S, Ns, D, Nd may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WY29584	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34895	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	16/32-bit +/-number	V,Z P0-P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ns	○	○	○	○	○	○	○	○	○	○	○	○	0-3	○
D		○	○	○	○	○	○		○	○*	○*	○		○
Nd	○	○	○	○	○	○	○	○	○	○	○	○	0-3	○

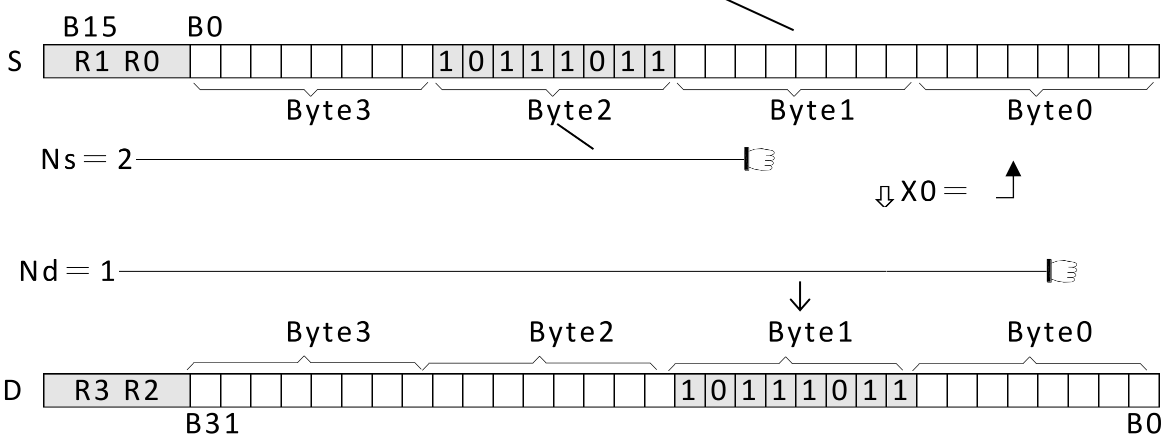
Description

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

FUN44 D P BYMV	BYTE MOVE	FUN44 D P BYMV
Example		

Ladder diagram	ST
	

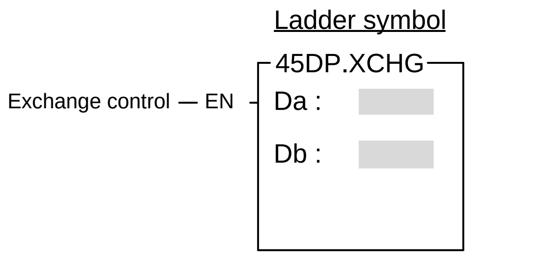
- The instruction moves the third byte (B16~B23) within S (32 bit register composed of R1R0), to the first byte within D (32 bit register composed of R3R2). Other bytes within D remain unchanged.



7-4-6 EXCHANGE (XCHG)

FUN45 D P XCHG	EXCHANGE	FUN45 D P XCHG

Symbol



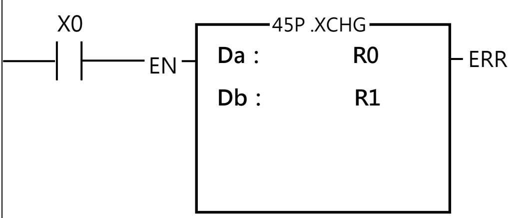
Da: Register a to be exchanged
Db: Register b to be exchanged
Da, Db may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	WY0 ∣ WY1008	WM0 ∣ WY29584	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	V,Z ∣ P0-P9
Da	○	○	○	○	○	○	○	○*	○*	○	○
Db	○	○	○	○	○	○	○	○*	○*	○	○

Description

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

Example

Ladder diagram	ST
	

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

FUN45 D P XCHG	EXCHANGE	FUN45 D P XCHG

B15 B0

Da	R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Db	R1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

⇩ X0 = ↑

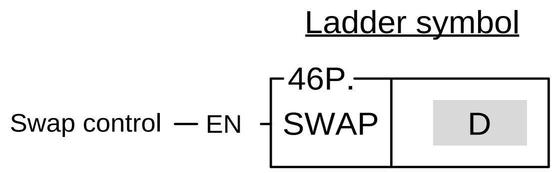
B15 B0

Da	R0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Db	R1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

7-4-7 BYTE SWAP (SWAP)

FUN46 P SWAP	BYTE SWAP	FUN46 P SWAP

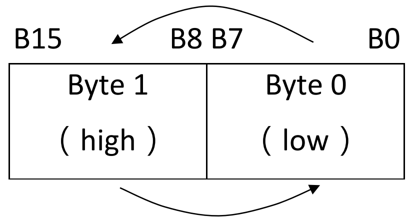
Symbol



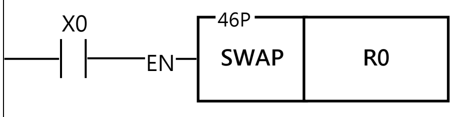
D: Register for byte data swap
D may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	V,Z P0-P9
D	○	○	○	○	○	○	○	○*	○*	○	○

Description



Example	

Ladder diagram	ST
	IF X0 THEN SWAP(D:= R0); END_IF

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

Byte1 Byte0

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

⇩ X0 = ↑

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

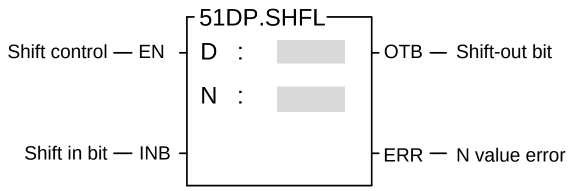
7-5 Shifting/Rotating Instructions (FUN51 ~ 54)

7-5-1 SHIFT LEFT (SHFL)

FUN51 D P SHFL	SHIFT LEFT	FUN51 D P SHFL

Symbol

Ladder symbol



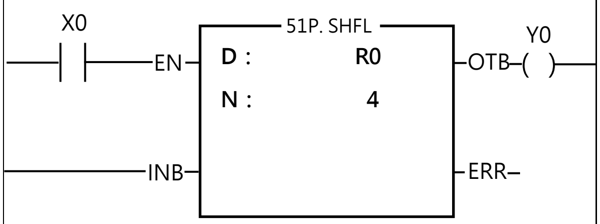
D: Register to be shifted
N: Number of bits to be shifted
N, D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WY29584	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34895	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	1 ∣ 16 or 1 ∣ 32	V,Z P0-P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

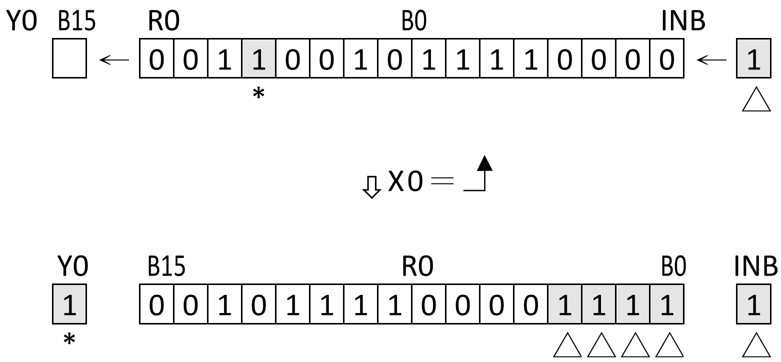
Description

- When shift control "EN" = 1 or has a transition from 0 to 1 (P instruction), will shift the data of the D register towards the left by N successive bits (in ascending order). After the lowest bit B0 has been shifted left, its position will be replaced by shift-in bit INB, while the status of shift-out bits B15 or B31 (D instruction) will appear at shift-out bit "OTB".
- If the operand is 16 bits, the effective range of N is 1~16. For 32 bits (D instruction) operand, it is 1~32. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.

FUN51 D P SHFL	SHIFT LEFT	FUN51 D P SHFL
Example		

Ladder diagram	ST
	

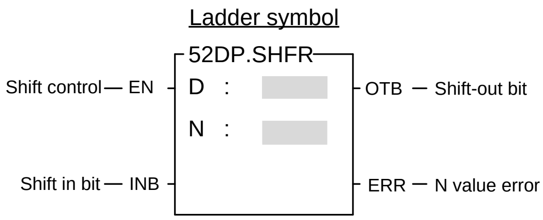
- The instruction shifts the data in register R0 towards the left by 4 successive bits. The results are shown below.



7-5-2 SHIFT RIGHT (SHFR)

FUN52 D P SHFR	SHIFT RIGHT	FUN52 D P SHFR

Symbol



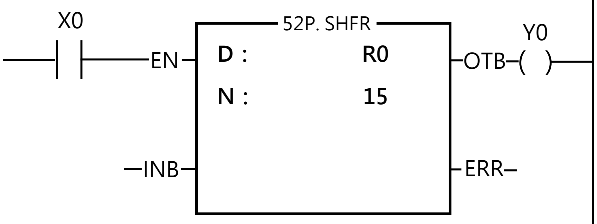
D: Register to be shifted
N: Number of bits to be shifted
D, N may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0–WX1008	WY0–WY1008	WM0–WY29584	WS0–WS3088	T0–T1023	C0–C1279	R0–R34767	R34768–R34895	R35024–R35151	R35280–R43223	R43224–R47319	D0–D11999	1–16 or 1–32	V,Z P0-P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Description

- When shift control "EN" = 1 or has a transition from 0 to 1 (P instruction), will shift the data of D register towards the right by N successive bits (in descending order). After the highest bits, B15 or B31 (D instruction) have been shifted right, their positions will be replaced by the shift-in bit INB, while shift-out bit B0 will appear at shift-out bit "OTB".
- If the operand is 16 bits, the effective range of N is 1~16. For 32 bits (D instruction) operand, it is 1~32. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.

FUN52 D P SHFR	SHIFT RIGHT	FUN52 D P SHFR
Example		

Ladder diagram	ST
	

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

INB		B15								R0						B0		Y0
0	→	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	→
△			*															

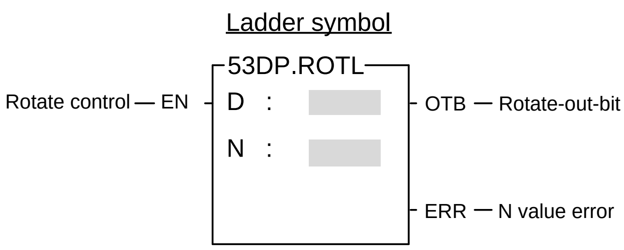
⇩X0 = ↑

INB	B15							R0							B0	Y0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
△	△	△	△	△	△	△	△	△	△	△	△	△	△	△		*

7-5-3 ROTATE LEFT (ROTL)

FUN53 D P ROTL	ROTATE LEFT	FUN53 D P ROTL

Symbol



D: Register to be rotated

N: Number of bits to be rotated

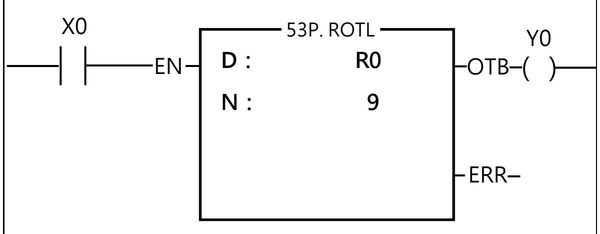
D, N may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ~ WX1008	WY0 ~ WY1008	WM0 ~ WY29584	WS0 ~ WS3088	T0 ~ T1023	C0 ~ C1279	R0 ~ R34767	R34768 ~ R34895	R35024 ~ R35151	R35280 ~ R43223	R43224 ~ R47319	D0 ~ D11999	1 ~ 16 or 1 ~ 32	V,Z P0-P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

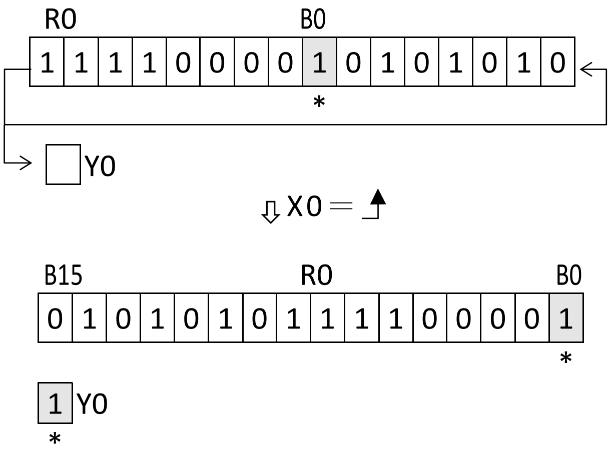
Description

- When rotate control "EN" = 1 or has a transition from 0 to 1 (P instruction), will rotate the data of D register towards the left by N successive bits (in ascending order, ie. in a 16-bit instruction, B0→B1, B1→B2, ..., B14→B15, B15→B0. In a 32-bit instruction, B0→B1, B1→B2, ..., B30→B31, B31→B0). At the same time, the status of the rotated out bits B15 or B31 (D instruction) will appear at rotate-out bit "OTB".
- If the operand is 16 bits, the effective range of N is 1~16. For 32 bits (D instruction) operand, it is 1~32. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.

FUN53 D P ROTL	ROTATE LEFT	FUN53 D P ROTL
Example		

Ladder diagram	ST
	

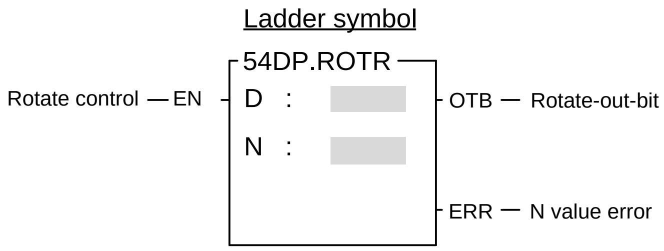
- The instruction rotates data from the R0 register towards the left 9 successive bits. The results are shown below.



7-5-4 ROTATE RIGHT (ROTR)

FUN54 D P ROTR	ROTATE RIGHT	FUN54 D P ROTR

Symbol



D: Register to be rotated

N: Number of bits to be rotated

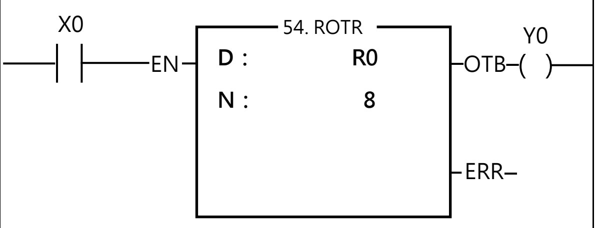
D, N may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ~ WX1008	WY0 ~ WY1008	WM0 ~ WY29584	WS0 ~ WS3088	T0 ~ T1023	C0 ~ C1279	R0 ~ R34767	R34768 ~ R34895	R35024 ~ R35151	R35280 ~ R43223	R43224 ~ R47319	D0 ~ D11999	1~16 or 1~32	V,Z P0-P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

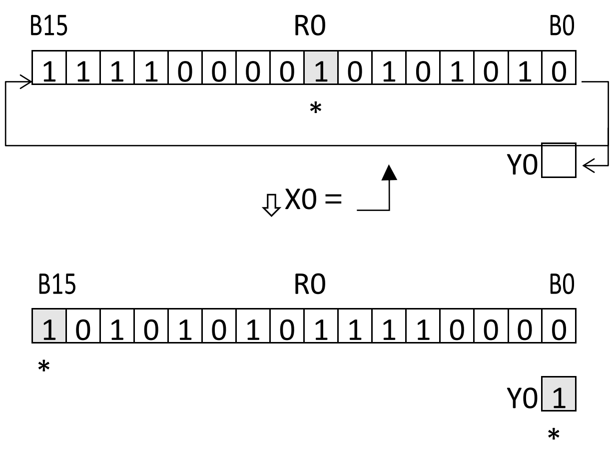
Description

- When rotate control "EN" = 1 or has a transition from 0 to 1(P instruction), will rotate the bit data of D register towards the right by N successive bits (in descending order, ie. in a 16-bit instruction, B15→B14, B14→B13, ..., B1→B0, B0→B15. In a 32-bit instruction, B31→B30, B30→B29, ..., B1→B0, B0→B31). At the same time, the status of the rotated out B0 bits will appear at the rotate-out bit "OTB".
- If the operand is 16 bits, the effective range of N is 1~16. For 32 bits (D instruction) operand, it is 1~32. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.

FUN54 D P ROTR	ROTATE RIGHT	FUN54 D P ROTR
Example		

Ladder diagram	ST
	

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

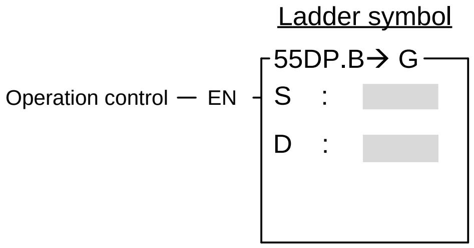


7-6 Code Conversion Instructions (FUN55 ~ 64)

7-6-1 INARY-CODE TO GRAY-CODE CONVERSION (B→G)

FUN55 DP B→G	BINARY-CODE TO GRAY-CODE CONVERSION	FUN55 DP B→G

Symbol



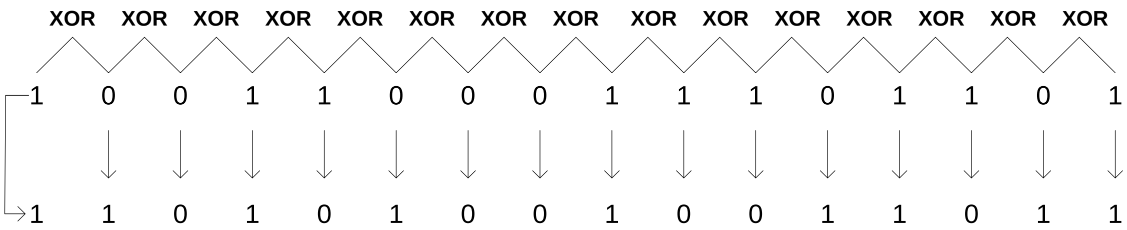
S: Starting of source

D: Starting addr ess of destination
S, D operand can combine V, Z, P0~P9 for index addressing

Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +/-number	V,Z P0-P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
D		○	○	○	○	○	○		○	○*	○*	○		○

Description

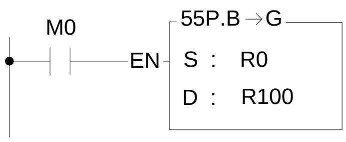
- When the execution control "EN"=1 or from 0→1 (P instruction), convert the binary code of the S register to Gray code.
- When the conversion bit is less than 16 bits, a temporary register is needed to store the conversion result. When it is greater than or equal to 16 bits, two registers are required (D instruction).
- The conversion method shown as below



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

Example1	

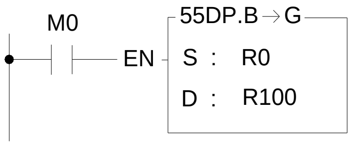
When M0 is from OFF→ON, convert R0 (binary code) into Gray code, and then store it in R100.

Ladder diagram	ST
	IF R_TRIG(S:= M0) THEN BintoGray(S:= R0, D:= R100); END_IF

R0＝1001010101010011B ➔ R100＝1101111111111010B

Example2	When M0 =1, it will perform the 32-bit code conversion

When M0 is ON, convert DR0 (binary code) to Gray code, and then store it in DR100.

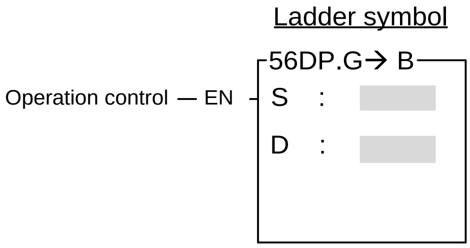
Ladder diagram	ST
	IF R_TRIG(S:= M0) THEN BintoGray(S:= DR0, D:= DR100); END_IF

DR0 = 00110111001001000010111100010100B ➔ DR100=00101100101101100011100010011110B

7-6-2 GRAY-CODE TO BINARY-CODE CONVERSION (G→B)

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

Symbol



S: Starting of source

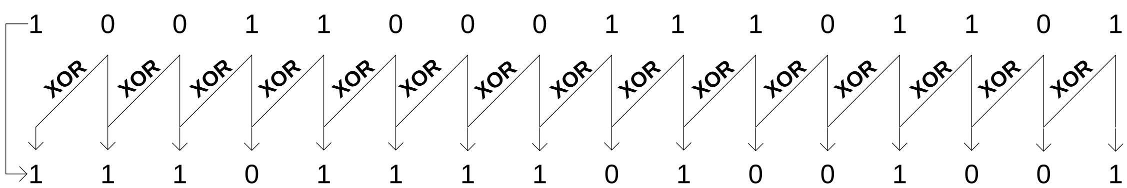
D: Starting address of destination

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

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WM29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +/-number	V,Z P0-P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
D		○	○	○	○	○	○		○	○*	○*	○		○

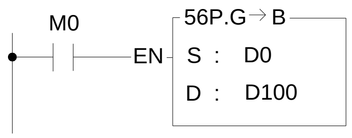
Description

- When the execution control "EN"=1 or from 0→1 (P instruction), convert the binary code of the S register to Gray code.
- When the conversion bit is less than 16 bits, a temporary register is needed to store the conversion result. When it is greater than or equal to 16 bits, two registers are required (D instruction).
- The conversion method shown as below:



FUN56 D P G→B	GRAY-CODE TO BINARY-CODE CONVERSION	FUN56 D P G→B
Example1		

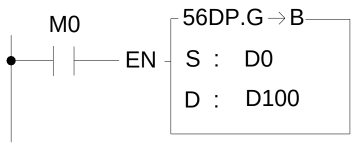
When M0 is from OFF→ON, convert D0 (binary code) into Gray code, and then store it in D100.

Ladder diagram	ST
	IF R_TRIG(S:= M0) THEN GraytoBin(S:= D0, D:= D100); END_IF

D0＝1001010101010011B ➔D100＝1110011001100010B

Example2	When M0 =1, it will perform the 32-bit code conversion

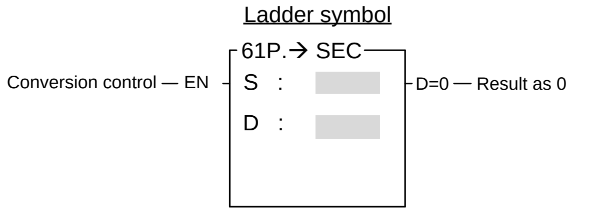
When M0 is ON, convert DD0 (binary code) to Gray code, and then store it in DD100

Ladder diagram	ST
	IF R_TRIG(S:= M0) THEN GraytoBin_D(S:= D0, D:= D100); END_IF

DD0 = 00110111001001000010111100010100B ➔ DD100 = 00100101110001111100101000011000B

7-6-3 HOUR：MINUTE：SECOND→SECOND

FUNCTION		
FUN61 P →SEC	HOUR：MINUTE：SECOND→SECOND	FUN61 P →SEC
Symbol		



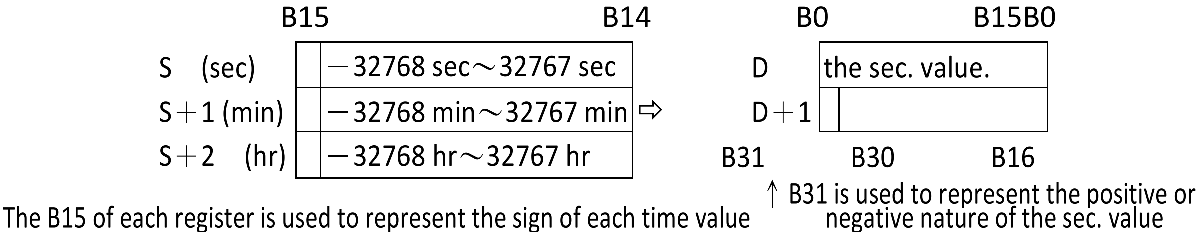
S: Starting calendar data register to be converted
D: Starting register storing results

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999
S	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○

Description	

FUN61 P →SEC	HOUR：MINUTE：SECOND→SECOND	FUN61 P →SEC

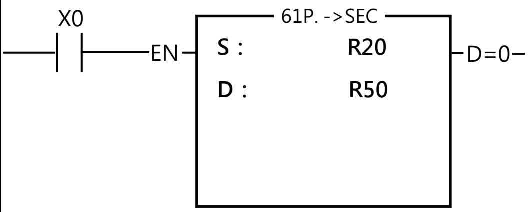
- When conversion control "EN" = 1 or has a transition from 0 to 1 (P instruction), will convert the hour: minute: second data of S~S+2 into an equivalent value in seconds and store it into the 32-bit register formed by combining D and D+1. If the result = 0, then set the "D = 0" flag as 1.
- Among the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

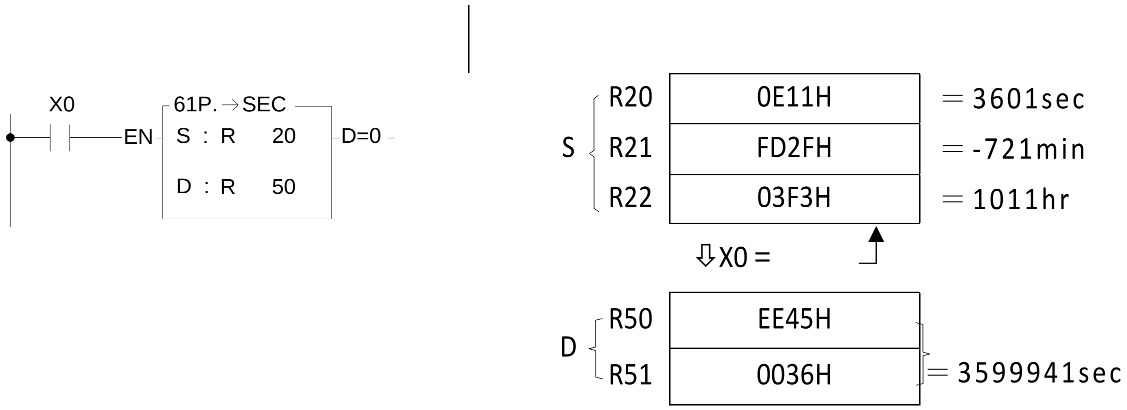


- Any [hour: minute: second] time data will be automatically merged and used except when accessing with FUN61 or 62 instructions. Other instructions will regard it as an individual general register and will not be automatically merged and used, there is no relationship between the 3 registers, so you can operate on any data of hours, minutes, and seconds separately, and the results will not affect each other.

FUN61 P →SEC	HOUR：MINUTE：SECOND→SECOND	FUN61 P →SEC
Example		

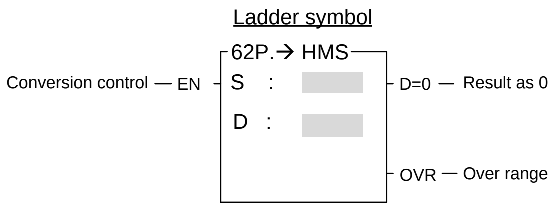
- The example program at below converts the hour: minute: second data formed by R20~R22 into their equivalent value in seconds then stored in the 32-bit register formed by R50~R51. The results are shown below:

Ladder diagram	ST
	IF R_TRIG(S:= X0) THEN ToSEC(S:= R20, D:=R50); END_IF



7-6-4 SECOND→HOUR：MINUTE：SECOND

FUN62 P →HMS	SECOND→HOUR：MINUTE：SECOND	FUN62 P →HMS
Symbol		



S: Starting register of second to be converted
D: Starting register storing result of conversion (hour : minute : second)

Range / Ope-Rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	-117968399 | 117964799
S	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○	

FUN62 P →HMS	SECOND→HOUR：MINUTE：SECOND	FUN62 P →HMS
Description		

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

S
S＋1 | Second |
B31 B16
↑
The bit B31 of the second register is used as the sign bit of the second value.

⇨

D (sec)	－59 sec～59 sec
D ＋ 1	－59 min～59 min
D ＋ 2	－32768 hr～32767 hr

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

- As shown in the diagram above, after convert to hour : minute : second value, the minute : second value can only be in the range of -59 to 59, and the hour number can be in the range of -32768 to 32767 hours. Because of this, the maximum limit of D is -32768 hours, -59 minutes, -59 seconds to 32767 hours, 59 minutes, 59 seconds, the corresponding second value of S which is in the range of -117968399 to 117964799 seconds. If the S value exceeds this range, this instruction cannot be carried out, and will set the over range flag "OVR" to 1. If S = 0 then result is 0 flag "D = 0" will be set to 1.

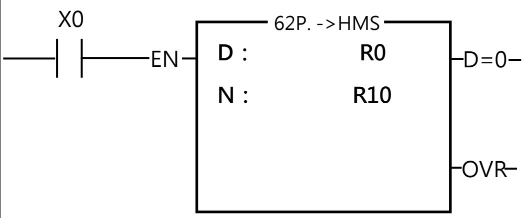
FUN62 P →HMS	SECOND→HOUR：MINUTE：SECOND	FUN62 P →HMS
Example		

- The program in the diagram below is an example of this instruction. Please note that the contents of the registers are denoted by hexadecimal, and on the right is its equivalent value in decimal notation.

R0	5D17H	6315287sec
R1	0060H	

⇩X0 = ↑

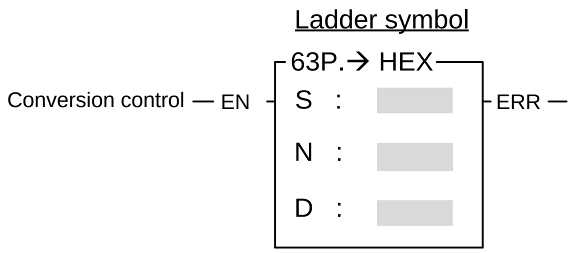
R10	002FH	47sec
R11	000EH	14min
R12	06DAH	1754hr

Ladder diagram	ST
	IF R_TRIG(S:= X0) THEN ToHMS(S:= R0, D:=R10); END_IF

7-6-5 CONVERSION OF ASCII CODE TO HEXADECIMAL VALUE (ASCII→HEX)

FUN63 P →HEX	CONVERSION OF ASCII CODE TO HEXADECIMAL VALUE	FUN63 P →HEX

Symbol



S: Starting source register.
N: Number of ASCII codes to be converted to hexadecimal values.
D: The starting register that stores the result (hexadecimal value).
S, N, D, can associate with V, Z, P0~P9 to do the indirect addressing application.

Range / Operand	WX	WY	WM	WS	TM	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WM29584	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34895	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	16-bit +number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	1 ~	○
D		○	○	○	○	○	○		○	○*	○*	○		○

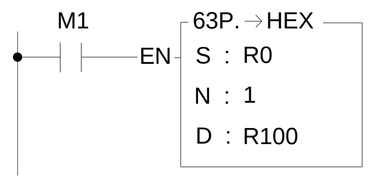
Description

- When conversion control "EN" =1 or changes from 0→1 (P instruction), it will convert the N successive hexadecimal ASCII character('0'~'9','A'~'F') convey by 16-bit registers (Low Byte is effective) into hexadecimal value, and store the result into the register starting with D. Every 4 ASCII code is stored in one register. The nibbles of register, which does not involve in the conversion of ASCII code will remain unchanged.
- The conversion will not be performed when N is 0 or greater than 511.
- When there is ASCII error (neither 30H ~ 39H nor 41H ~ 46H), the output "ERR" is ON.
- The main purpose of this command is to convert the ASCII numbers received by communication ports 1~2 from the external ASCII peripherals (transmitting values to the PLC in ASCII codes) into hexadecimal values that can be directly processed by the CPU.

FUN63 P →HEX	CONVERSION OF ASCII CODE TO HEXADECIMAL VALUE	FUN63 P →HEX

Example1	When M1 from OFF→ON, ASCII code converted to hexadecimal value.

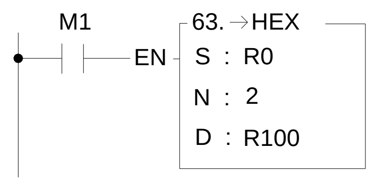
When M1 is from OFF→ON, convert ASCII code to hexadecimal value
Convert the ASCII code of R0 to hexadecimal value and store it in Nibble 0 of R100
(Nibble1~Nibble3 remain unchanged)

Ladder diagram	ST
	IF R_TRIG(S:= M1) THEN ToHEX(S:= R0, N:= 1, D:= R100); END_IF

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

Example2	When M1 is ON, ASCII code converted to hexadecimal value.

When M1 is ON, convert ASCII code to hexadecimal value
Convert the ASCII codes of R0 and R1 into hexadecimal values and store them in the low bytes of R100 (the high bytes remain unchanged)

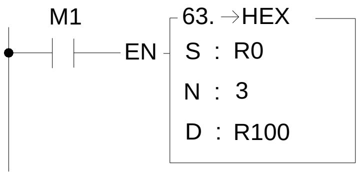
Ladder diagram	ST
	IF M1 THEN ToHEX(S:= R0, N:= 2, D:= R100); END_IF

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

Example3	

When M1 is ON, convert ASCII code to hexadecimal value

Convert the ASCII codes of R0~R2 into hexadecimal values and store them in R100 (Nibble 3 remains unchanged)

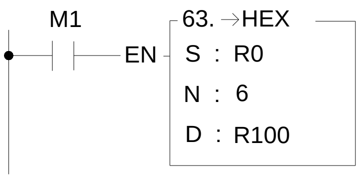
Ladder diagram	ST
	IF M1 THEN ToHEX(S:= R0, N:= 3, D:= R100); END_IF

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

Example4	When M1 is ON, ASCII code converted to hexadecimal value

When M1 is ON, convert ASCII code to hexadecimal value

Convert the ASCII codes of R0~R5 into hexadecimal values and store them in R100~R101

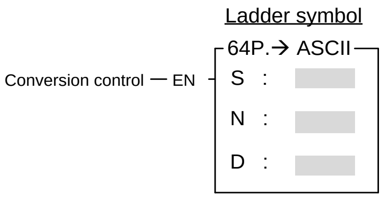
Ladder diagram	ST
	IF M1 THEN ToHEX(S:= R0, N:= 6, D:= R100); END_IF

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

7-6-6 CONVERSION OF HEXADECIMAL VALUE TO ASCII CODE (HEX→ASCII)

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

Symbol



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

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ｜ WX1008	WY0 ｜ WY1008	WM0 ｜ WM29584	WS0 ｜ WS3088	T0 ｜ T1023	C0 ｜ C1279	R0 ｜ R34767	R34768 ｜ R34895	R35024 ｜ R35151	R35280 ｜ R43223	R43224 ｜ R47319	D0 ｜ D11999	16-bit + number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	1~511	○
D		○	○	○	○	○	○		○	○*	○*	○		○

Description

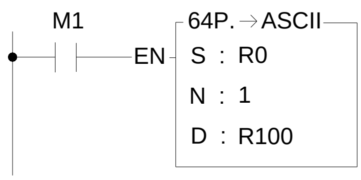
- When conversion control "EN" =1 or changes from 0→1 (P instruction), will convert the N successive nibbles of hexadecimal value in registers start from S into ASCII code, and store the result to low byte (high byte remain unchanged) of the registers which start from D.
- The conversion will not be performed when the value of N is 0 or greater than 511.
- The main purpose of this instruction is to convert the numerical value data, which PLC has processed, to ASCII code and transmit to ASCII peripherals by communication port1 or communication port 2.

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

Example1

When M1 is from OFF→ON, the converted hexadecimal value is ASCII code
Convert the Nibble 0 of R0 to ASCII code and store it in R100 (the high byte remains unchanged).

Ladder diagram	ST

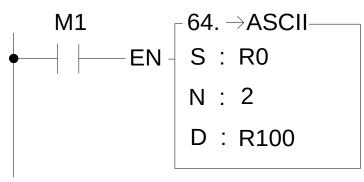
	`IF R_TRIG( S:= M1 ) THEN`
	`ToASCII ( S:= R0, N:= 1, D:= R100);`
	`END_IF`

R0＝0009H ➔ R100＝0039H（9）

Example2 When M1 is ON, it converts hexadecimal value to ASCII code.

When M1 is ON, convert the hexadecimal value to ASCII code
Convert NB0~NB1 of R0 into ASCII codes and store them in R100~R101 (the high byte remains unchanged).

Ladder diagram	ST

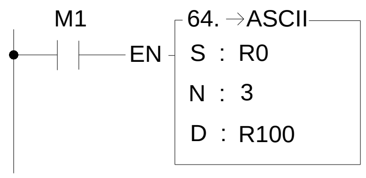
	`IF M1 THEN`
	`ToASCII ( S:= R0, N:= 2, D:= R100);`
	`END_IF`

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

Example3 When M1 is ON, it converts hexadecimal value to ASCII code.

When M1 is ON, convert the hexadecimal value to ASCII code
Convert NB0~NB2 of R0 into ASCII codes and store them in R100~R102

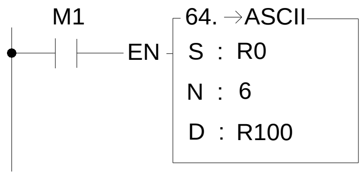
Ladder diagram	ST

	`IF M1 THEN`
	`ToASCII ( S:= R0, N:= 3, D:= R100);`
	`END_IF`

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

FUN 64 P →ASCII	CONVERSION OF HEXADECIMAL VALUE TO ASCII CODE	FUN 64 P →ASCII
Example4	When M1 is ON, it converts hexadecimal value to ASCII code.	

When M1 is ON, convert the hexadecimal value to ASCII code
Convert NB0~NB5 of R0~R1 to ASCII code and store in R100~R105

Ladder diagram	ST
	IF M1 THEN ToASCII (S:= R0, N:= 6, D:= R100); END_IF

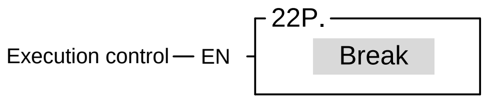
R0＝3456H	➔	R100＝0031H（1）
R1＝0012H		R101＝0032H（2）
		R102＝0033H（3）
		R103＝0034H（4）
		R104＝0035H（5）
		R105＝0036H（6）

7-7 Flow Control Instructions II (FUN22、FUN65 ~ 71)

7-7-1 Break

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

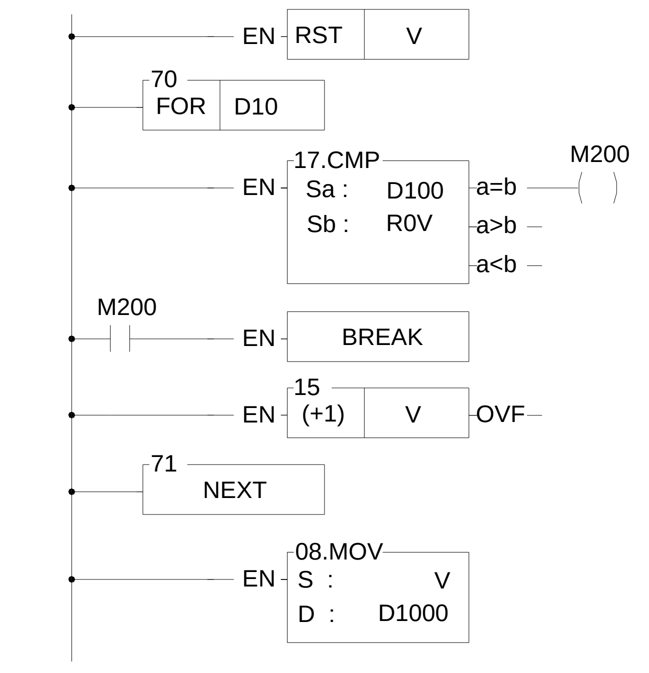
Ladder symbol



Description	

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

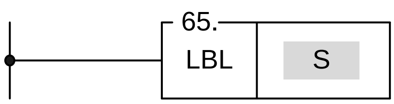
FUN22 P BREAK	BREAK FROM FOR AND NEXT LOOP (BREAK)	FUN22 P BREAK
Example		

Ladder diagram	ST
	see below


```
V := 0;
FOR D10 := 0 TO 10 BY 1  DO
IF D100 = R0V THEN
M200 := TRUE;
END_IF
IF M200 THEN
BREAK;
END_IF
V := V + 1;
END_FOR
D1000 := V;
```

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

7-7-2 LABEL (LBL)

FUN65 LBL	LABEL	FUN65 LBL
Symbol		
Ladder symbol 	S: Alphanumeric, 1~6 characters	
Description	※Only supported in the main program and subroutine	

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

FUN65 LBL	LABEL	FUN65 LBL

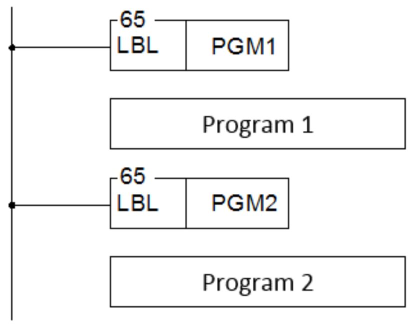
Reserved words	Interrupt
X0+I~X7+I (INT0~INT7) X0–I~X7–I (INT0–~INT7–)	Interrupt service program name of external X0~X7
HSC0I~HSC7I	Interrupt service routine name of HSC0 ~ HSC7
STM0I (1MS), STM1I (1MS), STM2I (1MS), STM3I (1MS), LTM0I (10MS), LTM1I (10MS), LTM2I (10MS), LTM3I (10MS)	1mS, 10mS, 2 kinds of timer interrupt service program name in PLC
HSTAI (ATMRI), HST0I~HST3I	Label for high-speed fixed timer interrupt service routine. In units of 0.1mS
COCPUI, LHMI, RHM0I, RHM1I, RHM2I, RHM3I, RHM4I, RHM5I	Labels for the pulse output command finished interrupt service routine.

Unless the program you marked is indeed the service program corresponding to the above interrupt, the above name can be used, and it cannot be used elsewhere. Otherwise, when an interrupt occurs, the PLC will execute the general program you marked as an interrupt program, resulting in errors or crash.

FUN65 LBL	LABEL	FUN65 LBL
Example		

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

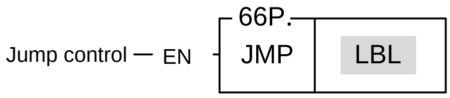
Ladder diagram	ST

	`LBL ( PGM1)`
	`Program 1`
	`LBL ( PGM2)`
	`Program 2`

7-7-3 JUMP (JMP)

FUN66 P JMP	JUMP	FUN66 P JMP
Symbol		

Ladder symbol

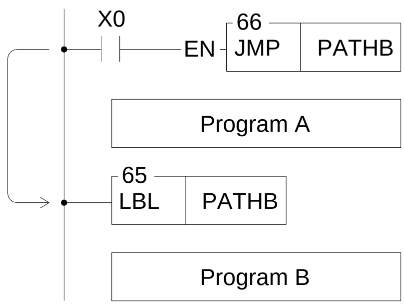


LBL: The program label to be jumped

Description

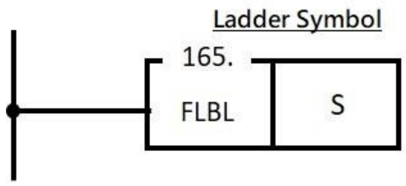
- When jump control “EN” = 1 or changes from 0→1 (P instruction), PLC will jump to the location behind the marked label and continuous to execute the program.
- This instruction is especially suit for the applications where some part of the program will be executed only under certain condition. This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

FUN66 P JMP	JUMP	FUN66 P JMP
Example		

Ladder diagram	ST
	IF X0 THEN JMP_66(“PATHB”) END_IF Program A LBL(“PATHB”) Program B

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

7-7-4 FUNTION BLOCK LABEL

FUN165 FLBL	FUNTION BLOCK LABEL	FUN165 FLBL
Symbol		
	S: English/ Digit 1~6	
Discription	※It only supported in the function block diagram.	

- This command labels a specific address in the program, so the program function block diagram jumps (FJUMP) to the address where the label is located for execution. If there is no need for flow control, such as jumping or calling, it can also be labeled to annotate the program to facilitate program identification or improve readability.
- This instruction is only used as a program address label for process control or annotation. The instruction itself will not perform any action. Whether there is this instruction in the program, the execution result will not be affected by it.
- The label can consist of 1 to 6 non-repeating arbitrary English letters or numbers.

FUN165 FLBL	FUNTION BLOCK LABEL	FUN165 FLBL

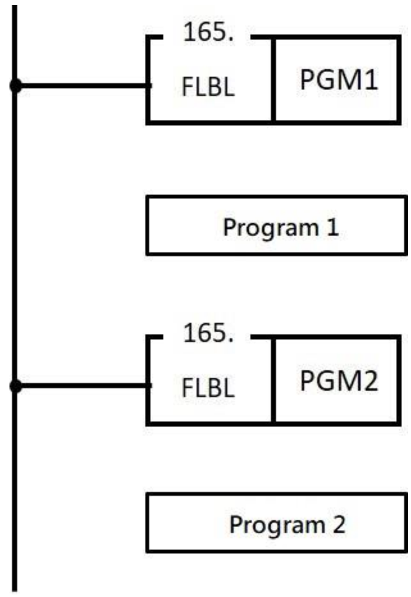
Reserved words	Discription
X0+I~X7+I (INT0 ~ INT7) X0–I~X7–I (INT0– ~ INT7–)	Labels for external input (X0~X7) interrupt service routine
HSC0I~HSC7I	Labels for high speed counter HSC0 ~ HSC7 interrupt service routine
STM0I (1MS), STM1I (1MS), STM2I (1MS), STM3I (1MS), LTM0I (10MS), LTM1I (10MS), LTM2I (10MS), LTM3I (10MS)	Labels for 1ms, 10ms PLC internal timer interrupt service routine
HSTAI (ATMRI), HST0I~HST3I	Labels for high-speed fixed timer interrupt service routine In units of 0.1mS
COCPUI, LHMI, RHM0I, RHM1I, RHM2I, RHM3I, RHM4I, RHM5I	Labels for expansion module event interrupt

Unless the program you labeled is indeed the service program corresponding to the above interrupts, the above labels can be used and cannot be used elsewhere. Otherwise, the PLC will execute the general program you labeled as an interrupt program when an interrupt occurs, resulting in errors or crashes.

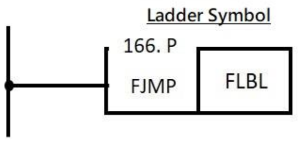
Example	

FUN165 FLBL	FUNTION BLOCK LABEL	FUN165 FLBL

The illustration below is an example of a label only used as a program comment (not called or jumped to this mark). As for applying the label in jump control, please refer to the description of the JMP instruction. Please refer to the order's CALL Description when the label is a subroutine name.

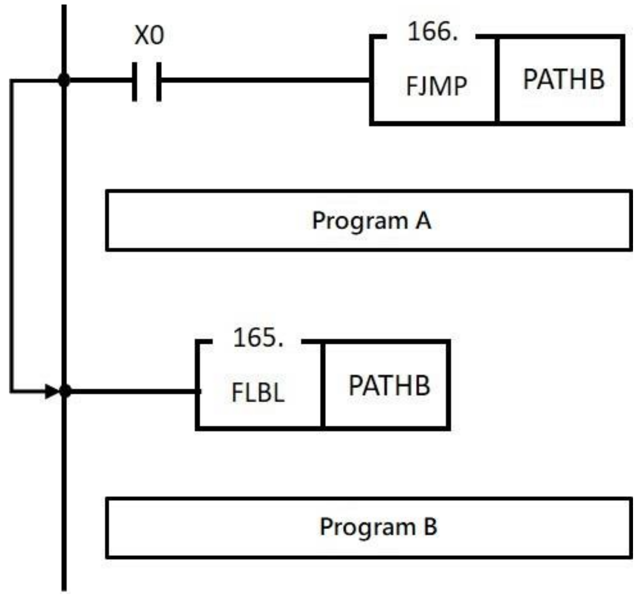
Ladder diagram	ST
 	LBL_F (“PGM1”) Program 1 LBL_F(“PGM2”) Program 2

7-7-5 FUNTION BLOCK JUMP

FUN166 P FJMP	FUNTION BLOCK JUMP	FUN166 P FJMP
Symbol		
	FLBL：The program label to be jumped	
Discription	※It's only supported in the function block diagram.	

- When the function block jump controls "EN"=1 or from 0→1 (P command), PLC directly jumps to the position labeled FLBL and continues to execute the program.
- This command is especially suitable for the application that only needs to execute a particular part of the program when a specific situation occurs, and in the application of multiple outputs of the coil and then use the input control to select and execute a particular section of the program—usually not managed to save time.
- This command can jump back (that is, the FLBL address of the jump back is smaller than the address of the FJMP command). Still, it should be noted that if the leap back causes the scan time to extend beyond the time set by the Watchdog Timer, the PLC will generate WDT Interrupted, stops running, and issues an error signal.
- Function block jump commands are limited to the same function block diagram.

FUN166 P FJMP	FUNTION BLOCK JUMP	FUN166 P FJMP
Example		

Ladder diagram	ST
	IF X0 THEN FJMP_166("PATHB") END_IF Program A LBL("PATHB") Program B

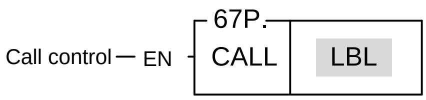
In the figure above, when X0=1, the execution will jump directly from where the JMP command is located to the site where the FLBL name is PATHB so that program A is skipped and all instructions in A are not executed. The list related to program A Points or register status remains unchanged (as if there is no A program).

7-7-6 CALL

FUN67 P CALL	CALL	FUN67 P CALL

Symbol

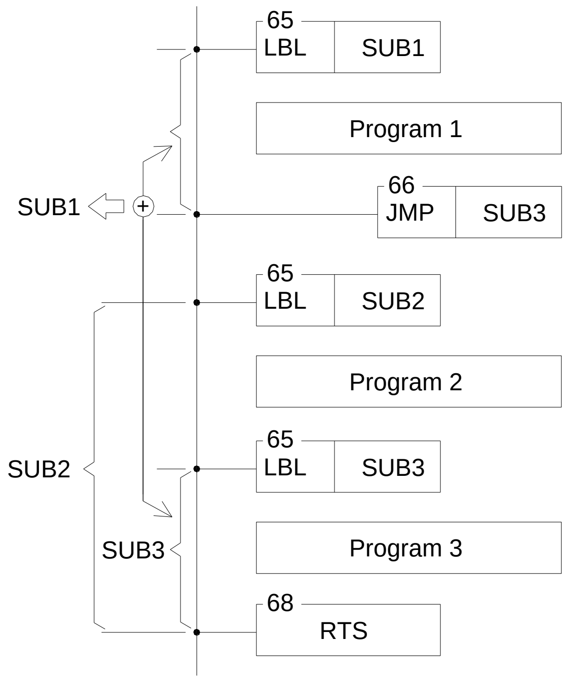
Ladder symbol



LBL: The subroutine label name to be called.

Description

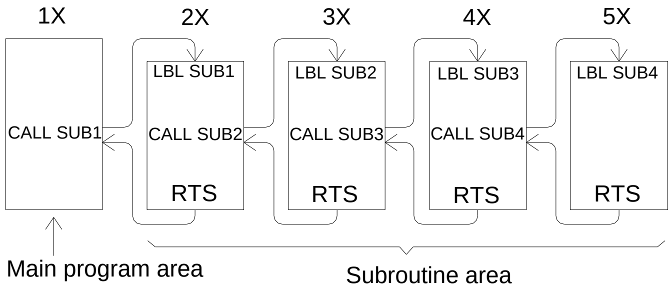
- When call control "EN" = 1 or changes from 0→1 (P instruction), PLC will call (perform) the subroutine bear the same label name as the one being called. When execute the subroutine, the program will execute continuous as normal program does but when the program encounters the RTS instruction then the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

Ladder diagram	ST
	LBL("SUB1") Program 1 JMP_66("SUB3") LBL("SUB2") Program 2 LBL("SUB3") Program 3 RETURN;

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

FUN67 P CALL	CALL	FUN67 P CALL

- When main program called a subroutine, the subroutine also can call the other subroutines (so called the nested subroutines) for up to 32 levels at the most (include the interrupt routine).

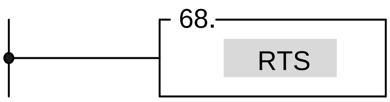


- Interrupt service programs (HSC0I~HSC7I, HST0I~HST3I, PSO0I~PSO3I, X0+I~X15+I / INT0~INT15, X0-I~X15-I / INT0-~INT15-, HSTAI / ATMRI, STM0I~STM3I, LTM0I~LTM3I, COCPUI, LHMI, RHM0I~RHM5I) are also a kind of subroutine. It is also placed in sub program area. However, the calling of interrupt service program is triggered off by the signaling of hardware to make the CPU perform the corresponding interrupt service program (which we called as the calling of the interrupt service program). The interrupt service program can also call subroutine or interrupted by other interrupts with higher priority. Since it is also a subroutine (which occupied one level), please refer to RTI instruction for explanation.

7-7-7 RETURN FROM SUBROUTINE (RTS)

FUN68 RTS	RETURN FROM SUBROUTINE	FUN68 RTS
Symbol		

Ladder symbol



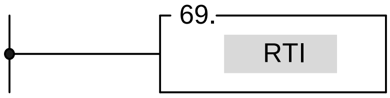
Description

- This instruction is used to represent the end of a subroutine. Therefore, it can only appear within the subroutine area. Its input side has no control signal, so there is no way to serially connect any contacts. This instruction is self sustain, and is directly connected to the power line.
- When PLC encounter this instruction, it means that the execution of a subroutine is finished. Therefore, it will return to the address immediately after the CALL instruction, which were previously executed and will continue to execute the program.
- If the above instructions are used in the subroutine and causing the subroutine not to execute the RTS instruction, then PLC will halt the operation and set the DR35361 'Bit9 (System Stack Error) to 1. Therefore, no matter what the flow is going, it must always ensure that any subroutine must be able to execute a matched RTS instruction.
- For the usage of the RTS instruction please refer to instructions for the CALL instruction.

7-7-8 RETURN FROM INTERRUPT (RTI)

FUN69 RTI	RETURN FROM INTERRUPT	FUN69 RTI
Symbol		

Ladder symbol



Description

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

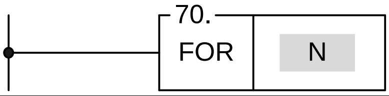
FUN69 RTI	RETURN FROM INTERRUPT	FUN69 RTI
Description		

- If there is an interrupt occurred while CPU is handling the higher priority (such as hardware high speed counter interrupt) or same priority interrupt program (please refer to Chapter 10 for priority levels), the PLC will not execute the interrupt program for this interrupt until all the higher priority programs were finished.
- If the RTI instruction cannot be reached and performed in the interrupt service routine, may cause a serious CPU shut down. Consequently, no matter how you control the flow of program, it must be assured that the RTI instruction will be executed in any interrupt service program.
- For the detailed explanation and example for the usage of interrupts, please refer to Chapter 5 for explanation.

7-7-9 FOR

FUN70 FOR	FOR	FUN70 FOR
Symbol		

Ladder symbol



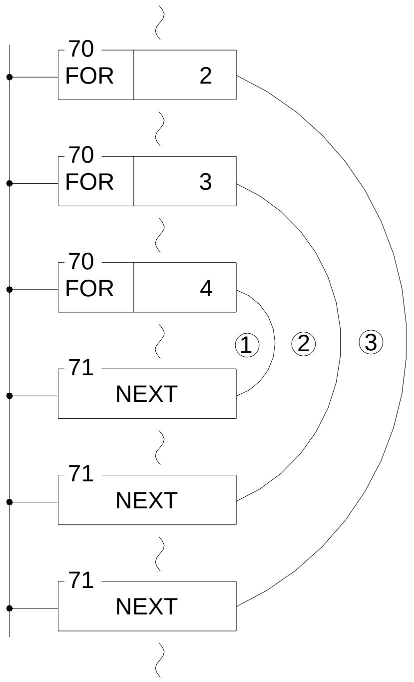
N: Number of times of loop execution

Range / Ope-Rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	1 | 16838
N	○	○	○	○	○	○	○	○	○	○	○	○	○

Description	

- This instruction has no input control, is connected directly to the power line, and cannot be in series with any conditions.
- The programs within the FOR and NEXT instructions form a program loop (the start of the loop program is the next instruction after FOR, and the last is the instruction before NEXT). When PLC executes the FOR instruction, it first records the N value after that instruction (loop execution number), then for N times successively execution from start to last of the programs in the loop. Then it jumps out of the loop, and continues executes the instruction immediately after the NEXT instruction.
- The loop can have a nested structure, i.e., the loop includes other loops, like an onion. 1 loop is called a level, and there can be a maximum of 32 levels. The FOR and NEXT instructions must be used in pairs. The first FOR instruction and the last NEXT instruction are the outermost (first) level of a nested loop. The second FOR instruction and the second last NEXT instruction are the second level, the last FOR instruction and the first NEXT instruction form the loop's innermost level.

FUN70 FOR	FOR	FUN70 FOR
Example		

<table>
<tr><th>Ladder diagram</th><th>ST</th></tr>
<tr><td></td><td>

```
FOR 0 TO 2 BY 1  DO
~
FOR 0 TO 3 BY 1  DO
~
FOR 0 TO 4 BY 1  DO
~
END_FOR
~
END_FOR
~
END_FOR
```

</td></tr>
</table>

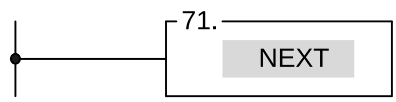
- In the example in the diagram, loop will be executed 4 ×3 ×2 = 24 times, loop will be executed 3 ×2 = 6 times, and loop will be executed 2 times.
- If there is a FOR instruction and no corresponding NEXT instruction, or the FOR and NEXT instructions in the nested loop have not been used in pairs, or the sequence of FOR and NEXT has been misplaced, then a syntax error will be generated and this program may not be executed.
- Do not use JMP command to jump out of the loop, otherwise the PLC system stack will be destroyed, the program flow will be disordered, and it may cause a serious crash.
- The effective range of N is 1~16383 times. Beyond this range PLC will treat it as 1. Care should be taken, if the amount of N is too large and the loop program is too big, a WDT may occur.

7-7-10 NEXT

FUN71 NEXT	LOOP END	FUN71 NEXT

Symbol

Ladder symbol

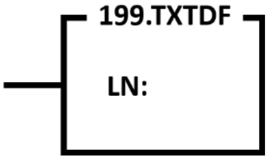


Description

- This instruction and the FOR instruction together form a program loop. The instruction itself has no input control, is connected directly to the power line, and cannot be in series with any conditions.
- When PLC has not yet entered the loop (has not yet executed to the FOR instruction, or has executed but then jumped out), but the NEXT instruction is reached, then PLC will not take any action, just as if this instruction did not exist.
- For the usage of this instruction please refer to the explanations for the FOR instruction on the preceding page.

7-7-11 Ladder Program Block Close-out Function (TXTDF)

FUN199 TXTDF	Ladder Program Block Close-out Function (TXTDF)	FUN199 TXTDF
Symbol		

Ladder Symbol 	LN: Text definition description
LN is available for inputting 1~200 bits	

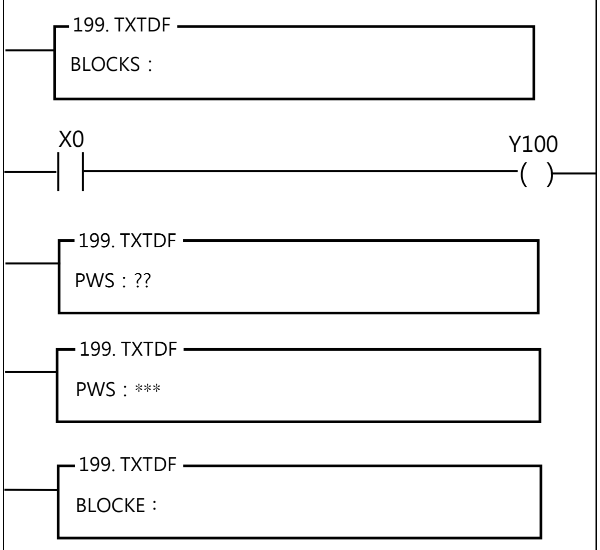
Description	

- By logging in a special keyword with the ladder FUN199.TXTDF command, you may use the block close-out function. Through such function, you may protect the ladder program in the Block Diagram easily.
- You may import 1~200 bits in Parameter LN for describing the text definition. Currently, the following words are retained and you need to prevent these bits from conflicting with each other when using.

FUN199 TXTDF	Ladder Program Block Close-out Function (TXTDF)	FUN199 TXTDF

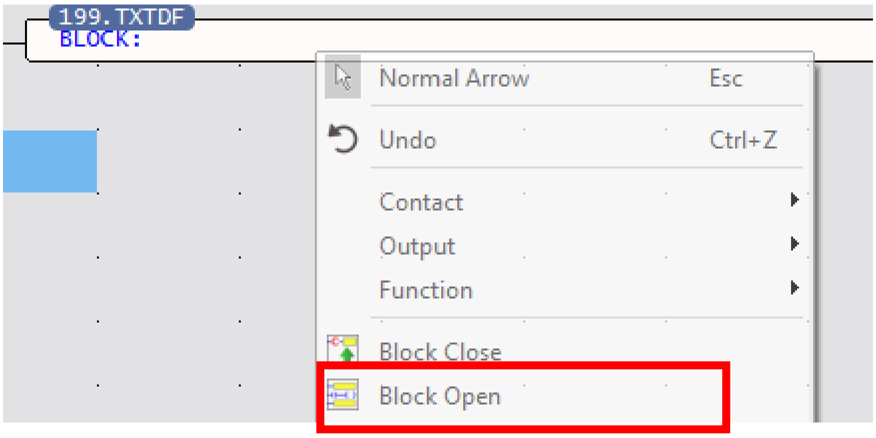
Reserved words	Description	Notes
BLOCKS:NAME	Block Diagram starting network commands	
BLOCKS:	Block Diagram starting network commands	
PSW:?	To open Block Diagram, you need to input password.	Effective when logged in the Block Diagram
PSWC:***	To open Block Diagram, you need to input password and it will be shown as *.	Effective when logged in the Block Diagram
PSW:CLOSE	Such block cannot enter the open state.	Effective when logged in the Block Diagram
BLOCKDSP:OPEN	When file is opened, this block enters the display state.	Effective when logged in the Block Diagram
BLOCKE:	Block Diagram end network commands.	

Example	

Ladder diagram	ST
	

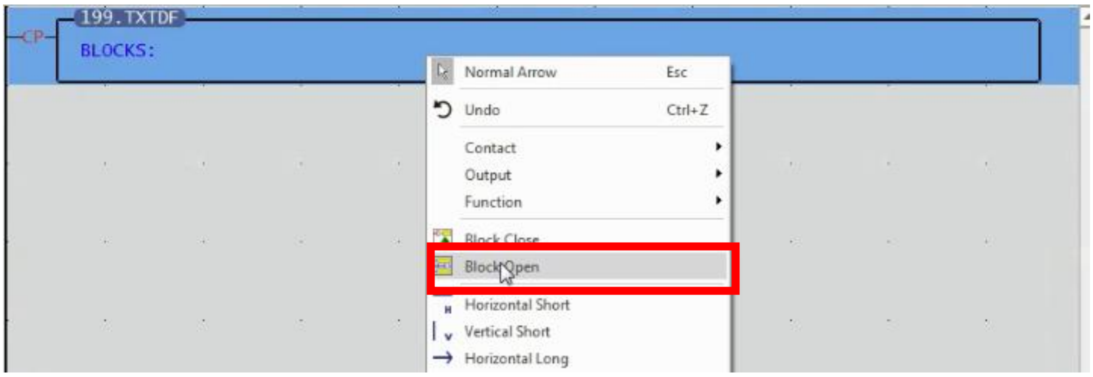
Lock-up process

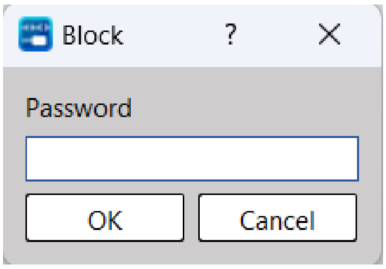
Per the example indicated in the diagram above, after clicking the first line, *i.e.,* the red box, with right key, you may select closing the program block per the figure indicated below:



FUN199 TXTDF	Ladder program block close-out function (TXTDF)	FUN199 TXTDF

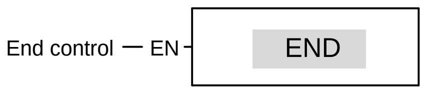
Unlock steps:
Assuming PSWC:*** in the sample program, *** is set to 123 (displayed as *** in the Ladder), then right-click on the program, choose to Block open, and then enter the password.
As shown below:





7-7-12 PROGRAM END

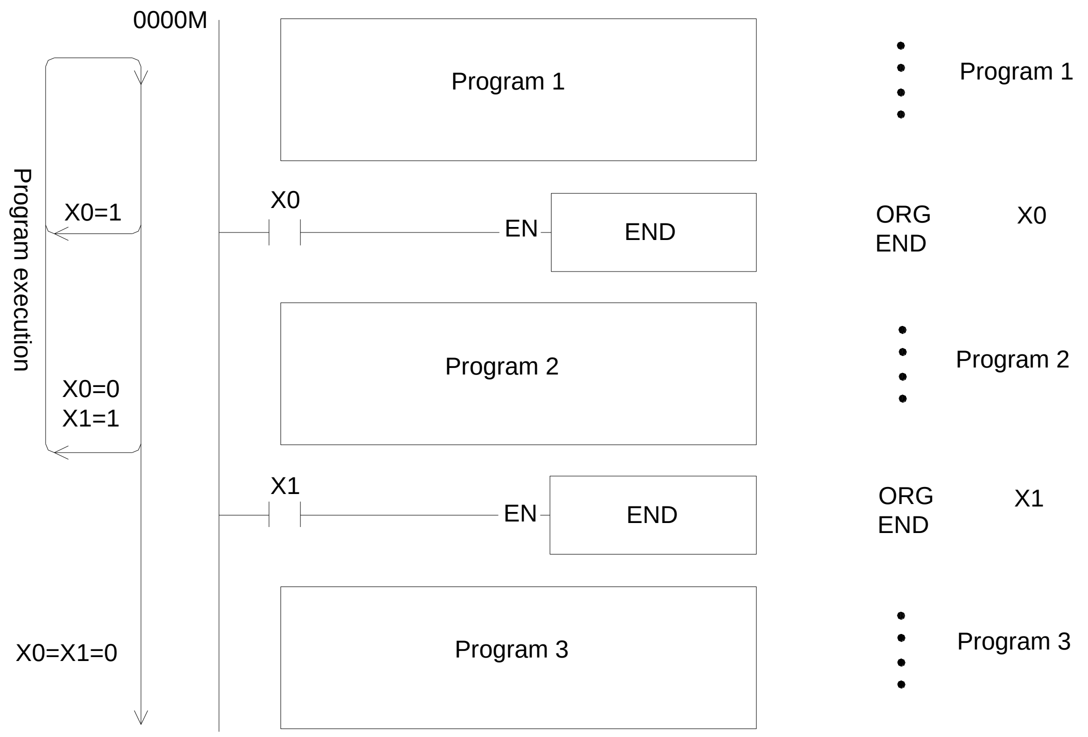
END	PROGRAM END	END
Symbol		

Ladder symbol	No operand
	

Description

- When end control "EN" = 1, this instruction is activated. Immediately end this program scan, all the programs after the END instruction will not be executed. When "EN" = 0, this instruction is ignored, and programs after the END instruction will continue to be executed as the END instruction is not exist.
- This instruction may be placed more than one point within a program, and its input (end control "EN") controls the end point of program execution. It is especially useful for debugging and for testing.
- It’s not necessary to put any END instructions in the main program, CPU will automatically restart to start point when reach the end of main program.

END	PROGRAM END	END
Example		

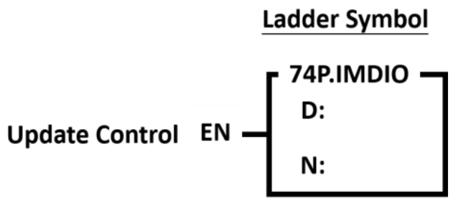


7-8 I/O Instructions (FUN74~86)

7-8-1 IMMEDIATE I/O REFRESH (IMDIO)

FUN74 P IMDIO	IMMEDIATE I/O REFRESH	FUN74 P IMDIO

Symbol



D: The starting address of the I/O point to be updated
N: The number of I/O points to be updated

Range / Ope-Rand	X	Y	K
	Xn	Yn	1 | 36
D	○	○	
N			○

Description

- The input/output signal update of the PLC system usually grabs all the input signals at one time before the program is executed, and then starts to scan the program. After all the scans are over, all the output results are sent to the output point at one time. In this way, the input action to the output response is at least there will be one scan time delay (maximum 2 scan times). The method of this instruction is to immediately grab or send the input signal or output signal specified by the Instruction when encountering this Instruction, so that the most immediate and fast input/output response can be obtained.

FUN74 P IMDIO	IMMEDIATE I/O REFRESH	FUN74 P IMDIO

- When update control "EN" = 1 or changes from 0 to 1 (P instruction), update the status of N input points or output points (i.e., D~D+N-1) starting from input point or output point designated by D.
- The I/O points of the immediate I/O update of the PLC are limited to the I/O points on the host computer. The following table shows the allowable real-time I/O numbers of MA and ME/MS hosts:

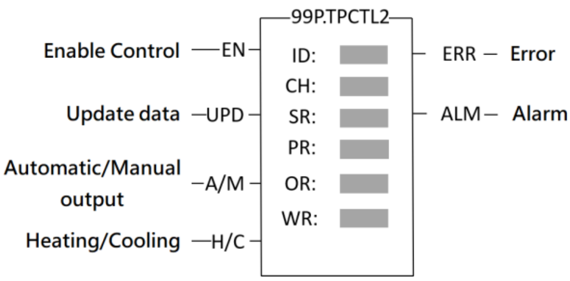
I/Oports Legal ports	MA	ME/MS
Input	X0 ~ X15	X0 ~ X7
Output	Y0 ~ Y15	Y0 ~ Y15

- If the range of the real-time I/O ports in the program exceeds the input point or output ports number of the host (for example, D=X7, N=9 in the program, it means that 9 input point signals such as X7~X15 should be captured immediately, and assuming that The Model is ME/MS model, the maximum input point is X7, obviously X15 has exceeded the input point number of the host), then the PLC will not be able to run.
- When this instruction is executed, although the PLC will immediately capture or send out the real-time input/output signal, the delay of the hardware or software components on the input point or the action delay of the output point (Action response time of output components such as relays or transistors) still exists, please pay special attention.

7-9 PID Control (FUN38, FUN99)

7-9-1 PID Temperature Control Instruction 2 (TPCTL 2)

FUN99 P TPCTL2	PID TEMPERATURE CONTROL INSTRUCTION 2	FUN99 P TPCTL2
Symbol		



ID: The number of the expansion module to perform temperature control
CH: Expansion module channel that performs temperature control
SR: Program-controlled setting start register
RP: Gain setting start register
OR: Output start register
WR: Work start register

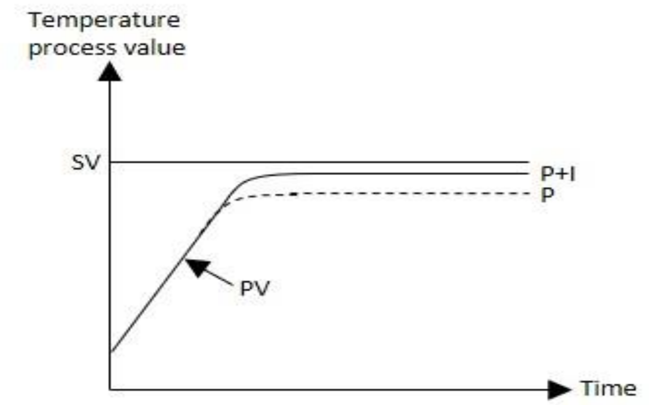
範圍 / 運算元	HR	IR	OR	SR	ROR	DR	K
	R0 ~ R34767	R34768 ~ R34895	R35024 ~ R35151	R35280 ~ R43223	R43224 ~ R47319	D0 ~ D11999	
ID							0 ~ 127
CH	○	○	○	○	○	○	0~63
SR	○				○	○	
PR	○				○*	○	
OR	○				○*	○	
WR	○				○*	○	

Discription	

- PID temperature control (FUN99) uses the temperature module and the temperature planning form to measure the current external temperature value as a Process Variable (referred to as PV) and the Set Point (Abbreviated as SP) set by the user and program-controlled variables through the software PID mathematical formula to obtain the appropriate output control value to control the temperature within the temperature range expected by the user.
- Convert the numerical result after PID operation into time-proportional ON/OFF (PWM) output, and control the heating or cooling circuit connected in series with the SSR through the transistor-type contact output so that a very accurate and inexpensive control result can be obtained.
- EN: Execute temperature control when ON, stop when OFF
- UPD: When ON, the parameters will be updated to the specified channel of the module
- A/M: PID manual mode, if enabled, the output will be in manual control mode, and the MOUT value will be automatically copied to MV instead of using the PID calculation result as the output.
- H/C: Perform heating or cooling control

PID control

- The PID control system is independently operated by the modules, and the PLC scan cycle will not be increased due to multiple modules performing PID at the same time.
- Each channel can perform its own PID calculation. The temperature control mode needs to be set to PID control. The temperature control can be performed more efficiently by using the proportional item (P), integral action (I) and differential action (D). Use demand to carry out P, PI, PD, PID control.
- Proportional item, the size of the output volume (MV) will become an output ratio with the error (E) between the measured value (PV) and the set value (SV), and the proportional item will fluctuate greatly when it is set. On the contrary, the fluctuation is small.
- Integral time, increase or decrease the output according to the error (E) between the measured value (PV) and the set value (SV), so as to reduce the steady-state error generated by the P action, the integral time setting; the smaller it is, the greater the fluctuation and the faster the rise, otherwise the smaller and the slower, the range is 0~3600s, if the integral time is 0, the integral control will not be performed.

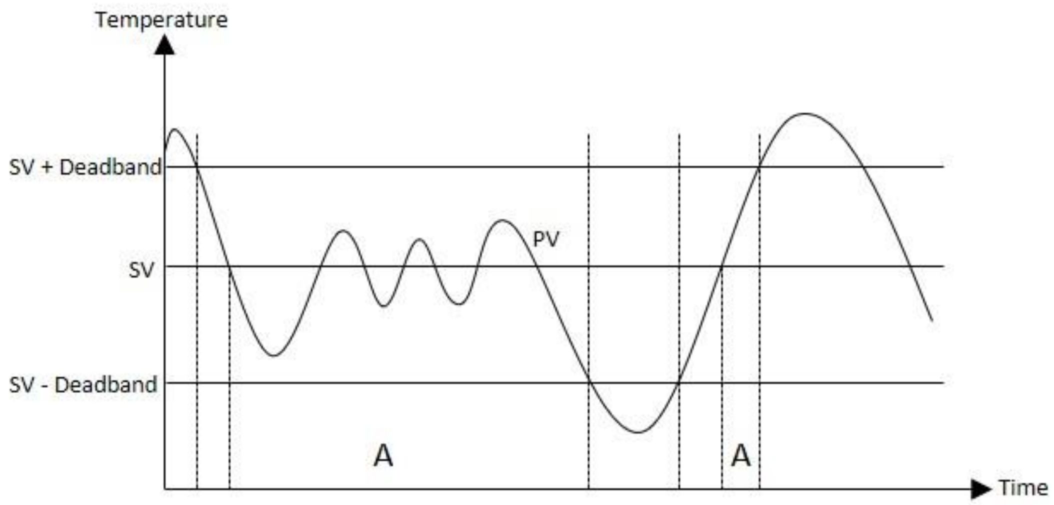


- Derivative time, increase or decrease the output according to the change rate of the error (E) between the measured value (PV) and the set value (SV), even if there is a sudden change due to the influence of noise, or on the control overshoot can return to a stable state in a short time through the derivative action. The smaller the derivative time setting, the smaller the fluctuation and the slower the response, otherwise the larger the faster, the range is 0~3600s. If the derivative time is 0, the derivative control is not performed.

FUN99 P TPCTL2	PID TEMPERATURE CONTROL INSTRUCTION 2	FUN99 P TPCTL2

SR Parameter	Word Size	Description
TS	1	Time cycle size, the unit is 0.1s (0.1s~30.0s)
SV	2	Set value, the unit is 0.1 degree
DEAD BAND	1	Reach the dead zone near the SV, the range is 0.1%~10.0%
DOUT	1	Output points
PERIOD	1	PWM period, the unit is 1s
Out mode	1	0, PWM Output 1, else

- PID_Deadband: The setting range is 0~10.0% (input range). In PID control, this area is a deviation (E) inactive area. When the , temperature program control value (PV) enters the dead zone at the beginning, it will still be normal. When the PID operation passes through the set value (SV), then the E will be substituted into the formula with 0, and the normal straight-line PID operation will resume after passing through this area. For example, E in area A in the figure is regarded as 0.



FUN99 P TPCTL2	PID TEMPERATURE CONTROL INSTRUCTION 2	FUN99 P TPCTL2

PR	Word Size	Description
Kp	2 (floating point)	Proportional term, real number
Ti	1	Integration time, 0~3600s
Td	1	Differential time, 0~3600s
Bias	2 (floating point)	Output deviation value, real number
High output limit	2 (floating point)	Output upper limit
Low output limit	2 (floating point)	Output lower limit
PID Method	1	0: Standard PID 1: Minimum transcendence method
AT	1	Whether AT is enabled
MAUTO	1	Does MOUT value change with MV

- Kp, Ti, Td: PID parameters, which can be adjusted after specifying or turning on AT automatic generation.
- Bias：The output bias value, the user can use it to increase or decrease the output value, but it will still be limited by the setting of the output range.
- High/Low output limit：Limit the output range, set the upper and lower limits of PID output, if the output lower limit is greater than or equal to the output upper limit, an error alarm will be issued.
- PID Method: Select a suitable PID algorithm
- AT: Whether to enable Autotuning to obtain PID control parameters
- MAUTO: Copy MV value to MOUT

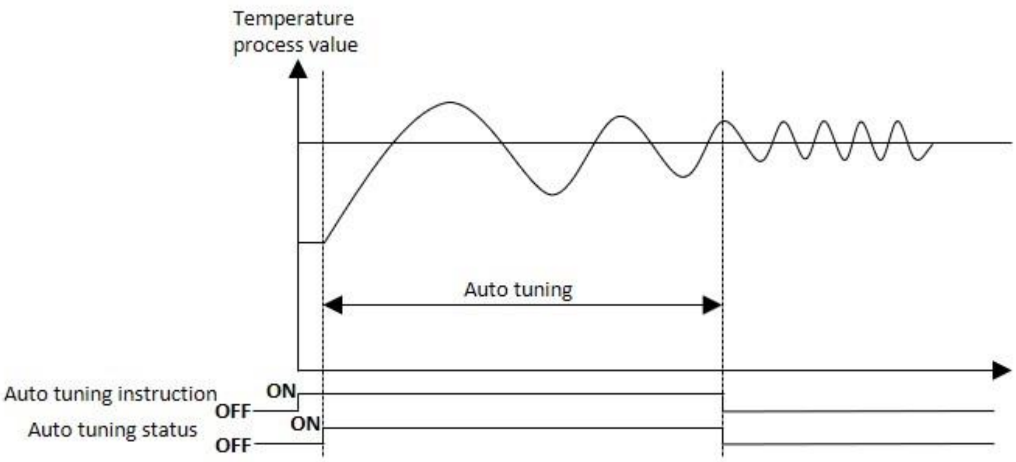
OR	Word Size	Description
MV	2 (floating point)	Output value return
MOUT	2 (floating point)	MV manual output value setting

FUN99 P TPCTL2	PID TEMPERATURE CONTROL INSTRUCTION 2	FUN99 P TPCTL2

WR	Word Size	Description
PID Operation Status	1	=0, Idle =1, Working =2, Error =3, AT now
AT Working Status	1	=0, Idle =1, Running =2, Error =3, Finish =4, Time out
PV	2	Programmed Value Return

Auto tuning

- This function can automatically calculate the appropriate proportional item (P), integral time (I) and differential time (D) PID parameters according to the control system environment. It can only be used after selecting the PID control mode and starting to perform temperature control. Temporarily Calculate through several waveforms obtained after ON/OFF control to obtain the best PID parameters. After the end, the parameters are automatically written into the respective memory of the PID and converted to PID control mode for temperature control.



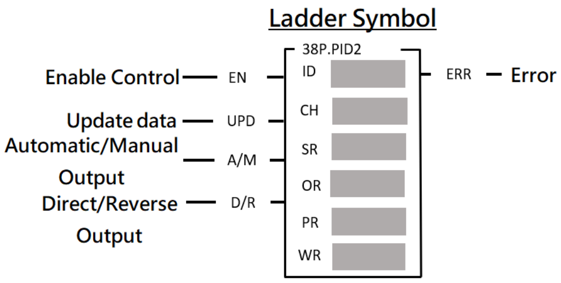
FUN99 P TPCTL2	PID TEMPERATURE CONTROL INSTRUCTION 2	FUN99 P TPCTL2

- During the period of auto tuning, the output upper limit and output lower limit will be referred to as the reference basis for the output, and the setting of the output period must not be 0 to perform auto tuning.
- If the SV setting exceeds the temperature range value, auto tuning will not be executed.
- If auto tuning has not been completed after 2 hours, an auto tuning timeout error will be issued.
- Channels that are set to off cannot perform the auto tuning function.
- If you change the setting values of SV, dead zone, TC module correction, output upper limit, output period, control mode and closed channel during auto tuning, auto tuning will stop and the error relay will be ON.
- Execution method: Through temperature control instruction
- Ending method: Auto tuning completes the report

7-9-2 General-Purpose PID 2 Instruction

FUN38 PID2	PID 2	FUN38 PID2

Symbol



ID: The number of the expansion module used as the input signal

CH: Channel number for input signal

SR: Program-controlled setting value starting register number, a total of 8 registers are occupied

OR: PID output register number

PR: The starting register number of the parameter setting value, a total of 7 registers are occupied

WR: The starting number of the working register used by this command, occupying 5 registers in total, and other places cannot be reused.

Operand \ Range	HR R0 | R34767	IR R34768 | R34895	SR R35280 | R43223	ROR R43224 | R47319	DR D0 | D11999	K
ID						0 ~ 127
CH	○	○	○	○	○	0 ~ 63
SR	○			○	○	
OR	○			○*	○	
PR	○			○	○	
WR	○			○	○	

Description

FUN38 PID2	PID 2	FUN38 PID2

- The general-purpose PID2 command (FUN38) regards the currently measured external analog input value as a process variable (Process Variable, referred to as PV). It sets the set point (SP) set by the user and the programmed variable through the software. After the PID mathematical calculation, the appropriate output control value is obtained through the D/A analog output module or reprocessed through other interfaces to control the controlled program within the user's wanted setting range.
- The digital PID calculation formula is as follows：

Mn = [KpxEn] $\sum_{0}^{n}[\mathrm{Kp} \times Ti \times Ts \times En]$ - [KpxTdx(PVn-PVn-1)/Ts] + Bias

Mn = ： Control output at "n" time

Kp ：Proportional term real number（range：±($1.8*10^{-38}$~$3.4*10^{38}$））

Ti ： Integral time constant（range：0~3600，equivalent to 0~3600 Repeats/Seconds）

Td ： Differential time constant（range：0~3600，equivalent to 0~3600 Seconds）

PVn ： Program-controlled variable value at "n" time

PV n-1 ： "n" last programmed variable value

En ： Error at "n" time = set value (SP) - program variable value at "n" time (PVn)

Ts ： Interval time between PID operations (range: 1~300, unit: 0.1S)

Bias ： Bias output (range: ± (1.8*10-38~3.4*1038)

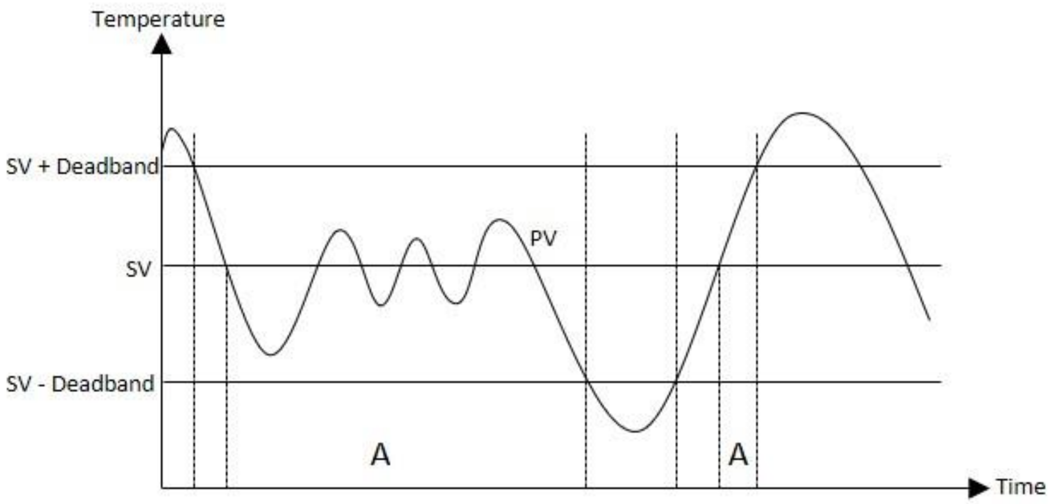
FUN38 PID2	PID 2	FUN38 PID2

- When the control selection "A/M"=0, it means the manual control mode, the PID calculation result will not be used, and the manual output value MOUT will be automatically copied to MV.
- When the control selection "A/M"=1, it means automatic control mode, the MV value is calculated by PID, if MAUTO=1, the MV value will be automatically copied to MOUT.
- When the control selection "A/M"=1 and the operation direction "D/R"=1, the program control is forward PID control; that is, when the error (SP-PVn) is positive, the control output of the PID operation result: The larger the value is; when the error is negative, the control output of the PID calculation result is smaller.
- When the control selection "A / M" = 1 and the operation direction "D/R" = 0, the program control is reverse PID control; that is, when the error (SP-PVn) is positive, the control output of the PID operation result: The smaller it is; when the error is negative, the control output of the PID operation result is larger.
- When the program control setting value or parameter setting value is wrong, the PID instruction will not be executed, and the error indicator "ERR"=1 is set.
- If you need to update the parameters, after updating the contents of the relevant registers, turn UPD OFF->ON to update the parameters.

FUN38 PID2	PID 2	FUN38 PID2

SR Parameter	Word Size	Description
TS	1	Time cycle size, the unit is 0.1s (0.1s~30.0s)
SV	2	Set value, the unit is 0.1 degree
DEAD BAND	1	Reach the dead zone near the SV, the range is 0.1%~10.0%

- PID_Deadband: The setting range is 0~10.0% (input range). In PID control, this area is a deviation (E) inactive area. When the , temperature program control value (PV) enters the dead zone at the beginning, it will still be normal. When the PID operation passes through the set value (SV), then the E will be substituted into the formula with 0, and the normal straight-line PID operation will resume after passing through this area. For example, E in area A in the figure is regarded as 0.



FUN38 PID2	PID 2	FUN38 PID2

PR	Word Size	Description
Kp	2 (floating point)	Proportional term, real number
Ti	1	Integration time, 0~3600s
Td	1	Differential time, 0~3600s
Bias	2 (floating point)	Output deviation value, real number
High output limit	2 (floating point)	Output upper limit
Low output limit	2 (floating point)	Output lower limit
PID Method	1	0: Standard PID 1: Minimum transcendence method
BUM	2	Smooth transfer enables (1 is enabled, 0 is disabled), it is a function of smooth transfer when manual to automatic control mode
AT	1	Whether AT is enabled
MAUTO	1	Does MOUT value change with MV

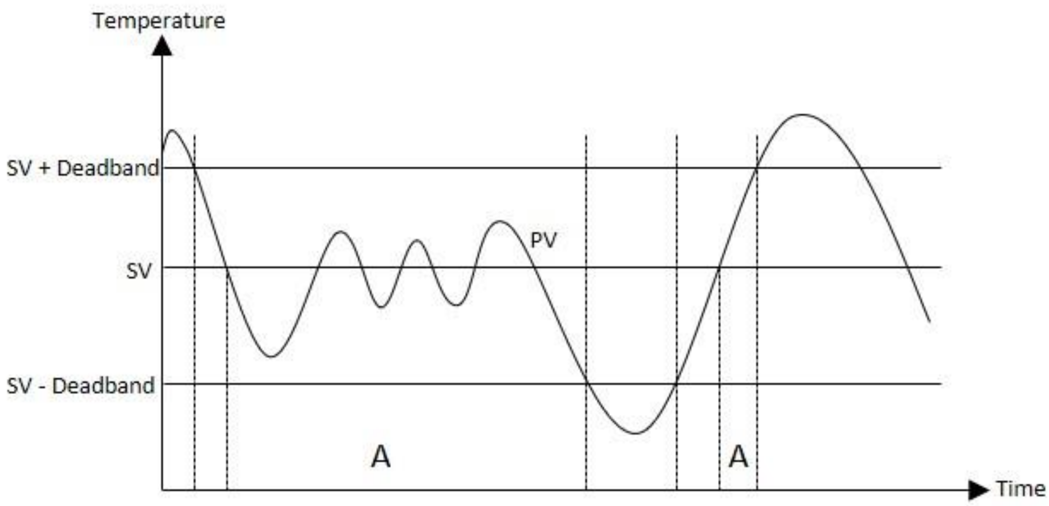
- Kp, Ti, Td: PID parameters, which can be adjusted after specifying or turning on AT automatic generation.
- Bias：The output bias value, the user can use it to increase or decrease the output value, but it will still be limited by the setting of the output range.
- High/Low output limit：Limit the output range, set the upper and lower limits of PID output, if the output lower limit is greater than or equal to the output upper limit, an error alarm will be issued.
- PID Method: Select a suitable PID algorithm
- AT: Whether to enable Autotuning to obtain PID control parameters
- MAUTO: Copy MV value to MOUT

OR	Word Size	Description
MV	2 (floating point)	Output value return
MOUT	2 (floating point)	MV manual output value setting

FUN38 PID2	PID 2	FUN38 PID2

SR Parameter	Word Size	Description
TS	1	Time cycle size, the unit is 0.1s (0.1s~30.0s)
SV	2	Set value, the unit is 0.1 degree
DEAD BAND	1	Reach the dead zone near the SV, the range is 0.1%~10.0%

- PID_Deadband: The setting range is 0~10.0% (input range), in the PID control, this zone is a deviation (E) ineffective zone. After the temperature program control value (PV) starting to enter the dead zone, the normal PID operation will continue until the set value (SV) is crossed. At this time, E will be substituted into the calculation formula with 0, and the normal straight PID operation will resume after crossing this zone, such as the area A in the figure, all E are regarded as 0.



FUN38 PID2	PID 2	FUN38 PID2

WR	Word Size	Description
PID Operation Status	1	=0, Idle =1, Working =2, Error =3, AT now
AT Working Status	1	=0, Idle =1, Running =2, Error =3, Finish =4, Time out
PV	2	Programmed Value Return

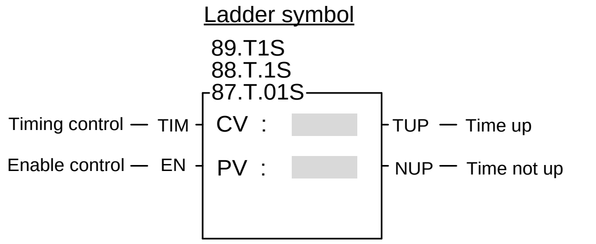
OR	Word Size	Description
MV	2 (floating point)	Output value return
MOUT	2 (floating point)	MV manual output value setting

7-10 Cumulateive Timer Instruction (FUN87~89)

7-10-1 ACCUMULATIVE TIMER (10ms, 100ms, 1s)

FUN87 T.01S FUN88 T.1S FUN89 T1S	ACCUMULATIVE TIMER (0.01s, 0.1s, 1s)	FUN87 T.01S FUN88 T.1S FUN89 T1S

Symbol



CV : Register storing elapse time (current value)

PV : Preset value of timer

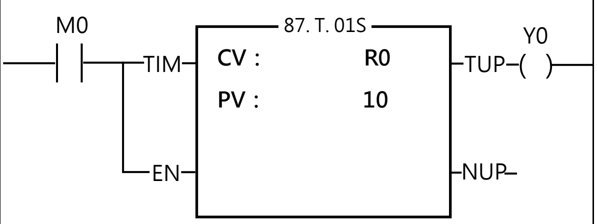
Range / Ope-Rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	0 | 32627 or 0 | 214783647
CV		○	○	○	○	○	○	○	○	○*	○*	○	○
PV	○	○	○	○	○	○	○	○	○	○		○	

Description

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

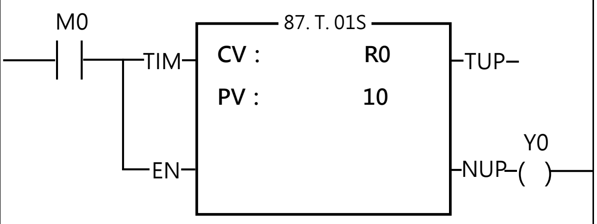
- The operation for this instruction is the same as that for the basic timer (T0~T1023), except that the basic timer only has a "timing control" input - when its input is 1 it starts timing, and when input is 0 it get clear. Every time the input changes, it starts timing again and is unable to accumulate. Timing with this instruction is only permissible when enable control "EN" = 1. With this instruction, when timing control "TIM" is 1, it is the same as a basic timer, but when "TIM" is 0, it does not clear, but keeps the current value. If the timer need to clear, then change enable control "EN" to 0. When timing control "TIM" is once again to be 1, it will continue to accumulate from the previous value when the timer last paused. In addition, this instruction also has two outputs: Time to “TUP” (when time up it is 1, usually it is 0) and Time not to “NUP” (usually it is 1, when time is up it is 0). Users can utilize input and output combinations to produce timers with various different functions.

Example 1	ON DELAY DE-ENERGIZING Timer

Ladder diagram	ST
	

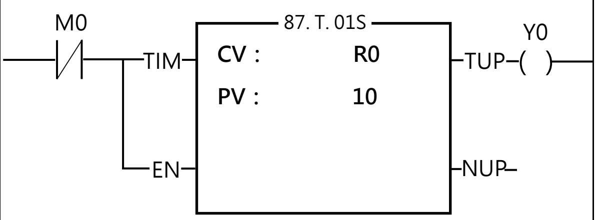
- This timer's output (Y0 in this example) is normally not energized. When this timer's input control (X0 in this example) is activated (ON), only after delay by 10 sec will output Y0 become energized (ON).

Example 2	ON DELAY DE-ENERGIZING Timer

Ladder diagram	ST
	

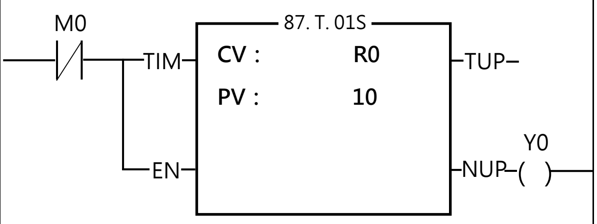
- The output Y0 of this timer is usually energized. When this timer's input control X0 is on, only after delay by 10 sec will the output become de-energized (OFF).

Example 3	OFF DELAY ENERGIZING Timer

Ladder diagram	ST
	

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

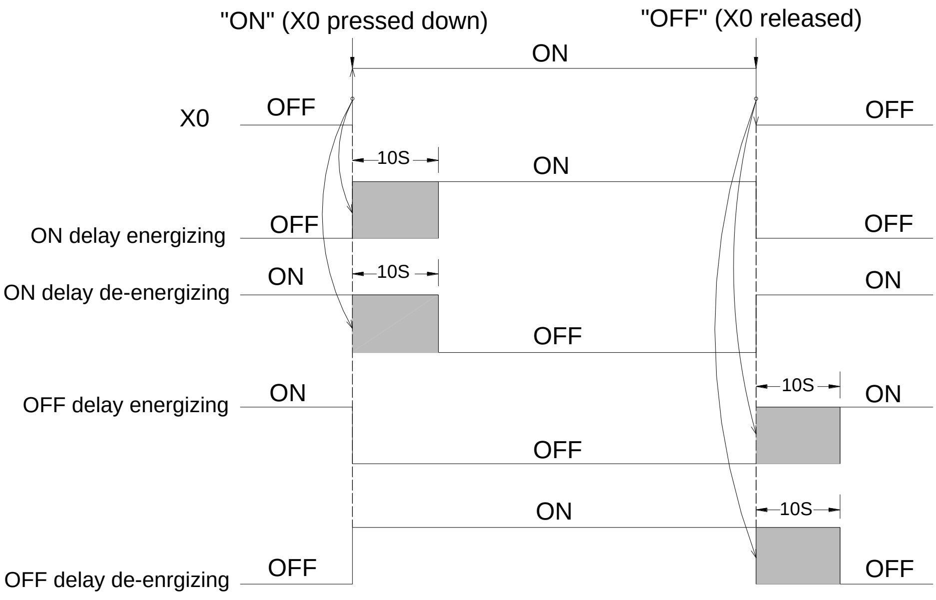
Example 4	OFF DELAY ENERGIZING Timer

Ladder diagram	ST
	

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

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

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

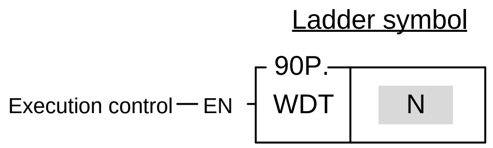


7-11 Watchdog Timer Instructions (FUN90~91)

7-11-1 Watchdog Timer (WDT)

FUN90 P WDT	WATCHDOG TIMER	FUN90 P WDT

Symbol



N: The watchdog time.
Its value can only be 50, 60, 70...990,
The unit is 10MS, that is, the set time range is (50~990) x10MS, that is, 0.05 seconds to 9.9 seconds.

Description

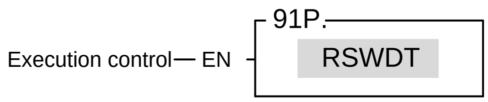
- When the execution control "EN"=1 or from 0→1 (P instruction), change the setting time of the monitoring timer to NX10MS. Once set, WATCHDOG TIMER (WDT) will use this as the timing time, if the scan time exceeds the set time, the PLC will stop and not execute.
- The watchdog timer is normally implemented 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.
- Once the set value is set, it will be saved forever, and there is no need to set it once for each scan, so this command should be used practically P instruction.
- The WDT time is set at 0.25 seconds.
- For the working principle of WDT, please refer to the FUN91 (RSWDT) instruction.

7-11-2 RESET WATCHDOG TIMER (RSWDT)

FUN91 P RSWDT	RESET WATCHDOG TIMER	FUN91 P RSWDT

Symbol

Ladder symbol



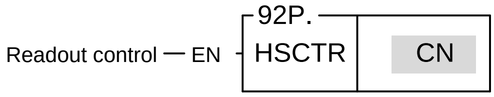
This instruction has no operand.

Description

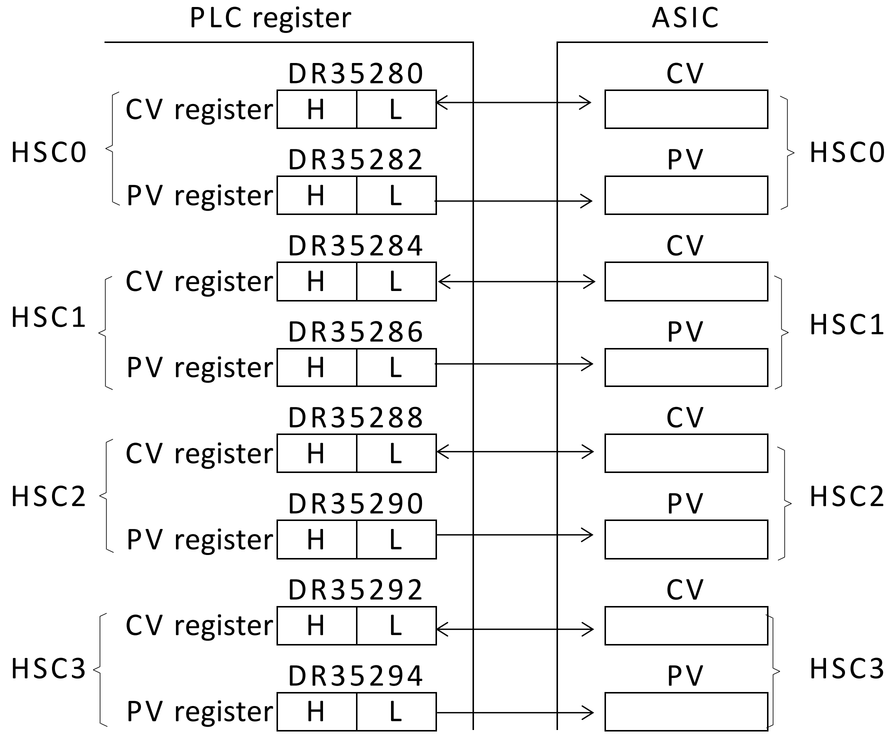
- When the execution control "EN"=1 or from 0→1 (P instruction), the WDT timer is cleared (that is, the WDT starts counting from 0 again).
- The function of WATCHDOG TIMER has been described in FUN90 (WDT command), and its principle is as follows:
- WATCHDOG TIMER are generally hardware ONE-SHOT timers (you cannot use software to do this, otherwise if the CPU crashes, the timer will be invalid, of course it cannot be protected), the so-called one-shot means That is, as long as you trigger the timer once, the timer value will be cleared to 0 immediately and restarted. If you do not trigger the WDT after it starts timing, the WDT timing will continue to increase to the set value N, and then the WDT will act and stop the PLC. If you trigger the WDT once before the WDT timing N has reached, the WDT will never happen, and the PLC uses this principle to ensure system security, because the PLC generally enters the program scan and I/O update WDT is triggered once during system service (HOUSEKEEPING). If the system is normal and the scan time does not exceed the set time N of WDT, there must be time to clear WDT and make it inactive. However, if the CPU is damaged, WDT cannot be triggered. Or the scan time is too long to trigger the WDT within N time, the WDT will act and turn off the PLC.
- In some applications, you have set the WDT time (FUN90), and your program scans the time in some cases, and it may temporarily exceed the set time of WDT, which is expected and allowed by you. Of course, you don’t want the PLC to stop because of this. At this time, you can use this command to trigger WDT to avoid WDT from happening. This is the main purpose of this command.

7-12 High Counting/Timing Instruction (FUN92~93)

7-12-1 Hareware High Speed Counter Current Value Access

<table>
<tr><td>FUN92D P
HSCTR</td><td colspan="2">Hareware High Speed Counter Current Value (CV) Access</td><td>FUN92D P
HSCTR</td></tr>
<tr><td>Symbol</td><td colspan="3">*When the high-speed counter is used as 32bits, it can only count down, and the PV can only be set to 0.</td></tr>
<tr><td colspan="2">Ladder symbol

</td><td colspan="2">CN : Hardware high speed counter number
0 : HSC0
1 : HSC1
2 : HSC2
3 : HSC3
4 : HSC4
5 : HSC5
6 : HSC6
7 : HSC7</td></tr>
<tr><td>Description</td><td colspan="3"></td></tr>
</table>

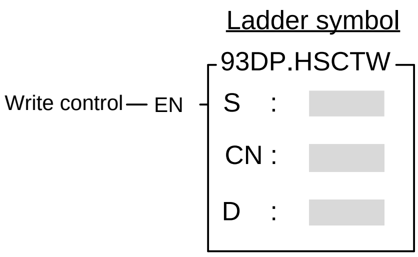
- The HSC0 ~ HSC3 counters of M-Series PLC are 4 sets of 32bit high speed counter with the variety counting modes such as up/down pulse. All the 4 high speed counters are built in the ASIC hardware and could perform count, compare, and send interrupt independently without the intervention of the CPU. 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.



FUN92D P HSCTR	Hareware High Speed Counter Current Value (CV) Access	FUN92D P HSCTR

- When access control "EN" =1 or changes from 0→1(P instruction), will gets the CV value of HSC designated by CN from ASIC and puts into the HSC corresponding CV register (i.e. the CV of HSC0 will be read and put into DR35280 or the CV of HSC1 will be read and put into DR35284).
- 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 8 "The high speed counter and high speed timer of M-Series PLC".

7-12-2 Hardware High Speed Counter Current Value and Preset Value Writing

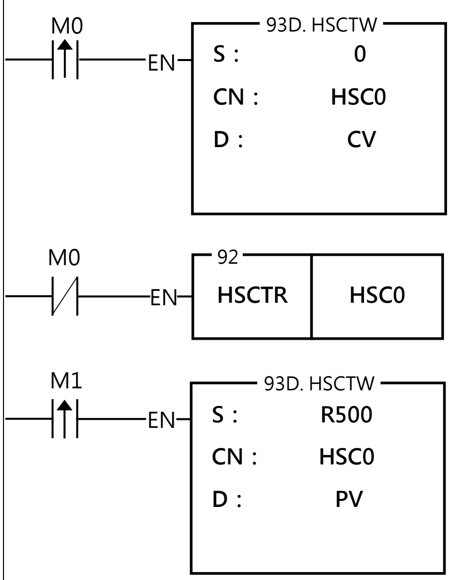
<table>
<tr><td>FUN93D P
HSCTW</td><td colspan="2">Hardware High Speed Counter Current Value and Preset Value Writing</td><td>FUN93D P
HSCTW</td></tr>
<tr><td>Symbol</td><td colspan="3">*When the high-speed counter is used as 32bits, it can only count down, and the PV can only be set to 0.</td></tr>
<tr><td colspan="2">
</td><td colspan="2">CN：Hardware high speed counter to be written
0：HSC0
1：HSC1
2：HSC2
3：HSC3
4：HST4
2：HSC2
3：HSC3
4：HST4
D: Write target (0 represents CV, 1 r epresents PV)</td></tr>
<tr><td>Description</td><td colspan="3"></td></tr>
</table>

FUN93D P HSCTW	Hardware High Speed Counter Current Value and Preset Value Writing	FUN93D P HSCTW

- Please refer to FUN92 for the relationship between the CV or PV values of HSC0~HSC7 in the ASIC and the corresponding CV registers and PV registers inside the PLC.
- When the writing control "EN"=1 or from 0→1 (P command), write the contents of the CV register or PV register of the high-speed counter designated by the PLC internal CN to the ASIC correspondingly CV or PV of HSC.
- General applications often need to write PV, that is, write your preset set value to the PV in ASIC. When the count value reaches your set value, the counter will immediately send an interrupt. Through the interrupt service program, you It can be used for various precise counting or positioning control.
- M SERIES PLC will automatically read the value of the current value register CV of HSC0~HSC3 inside the ASIC at that time when the power is off, and then write it into the CV register of HSC0~HSC3 inside the PLC (with power-off hold function), and when the PLC is powered on again, it will reversely write the CV registers inside the PLC back to the CV registers inside the ASIC. The content value of the register will automatically return to the value before the last power failure, but if your control application needs to be cleared to 0 or start counting from a specific value when the power is restored, you must use this instruction to do ASIC internal write to the CV value of HSC.
- For detailed applications, please refer Chapter 7 “The high-speed counter and high speed timer of M-Series PLC”.

FUN93D P HSCTW	Hardware High Speed Counter Current Value and Preset Value Writing	FUN93D P HSCTW
Example		

Ladder diagram	ST




```
IF R_TRIG( S:= M0) THEN
HSCTW( S:= 0, CN:= HSC_HSC0, D:=
HSC_CV);
END_IF
IF M0 = FALSE THEN
HSCTR( CN:= 0);
END_IF
IF R_TRIG( S:= M1) THEN
HSCTW( S:= R500, CN:= HSC_HSC0, D:=
HSC_PV);
END_IF
```

FUN93D P HSCTW	Hardware High Speed Counter Current Value and Preset Value Writing	FUN93D P HSCTW

As the program in this 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 DR35282, 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.

7-13 Slow Up/Slow Down (FUN95~98)

7-13-1 TRACKING TYPE RAMP FUNCTION FOR D/A OUTPUT

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

```
                   ┌ 98.RAMP2  ┐
Execution EN ──────┤ Om :      ├── ACC
                   │ Ta :      │
                   │ Td :      ├── DEC
                   │ Rt :      │
                   │ Rc :      │
                   │ WR :      │
                   └───────────┘
```

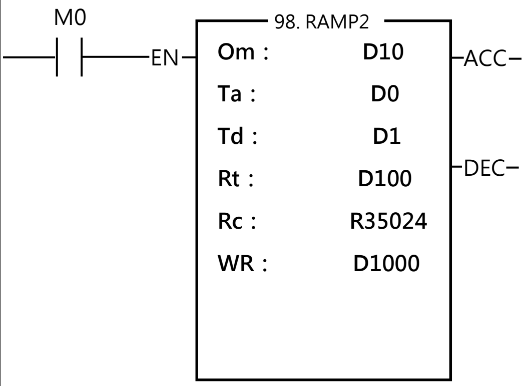
OM: Maximum output; range from 0~65535
TA: The acceleration time for the output, from 0 up to maximum; Range from 0~65000, unit is in mS.
TD: The deceleration time for the output, from maximum down to 0; Range from 0~65000, unit is in mS.
RT: Register of target output; Range from 0~65535
RC: Register of current output, it is used for analog output
WR: Starting address of working registers, it needs 4 registers

Operand \ Range	HR	OR	ROR	DR	K
	R0 │ R3839	R35024 │ R3967	R5000 │ R8071	D0 │ D3999	16-Bit
Om	○	○	○	○	0~65535
Ta	○	○	○	○	0~65000
Td	○	○	○	○	0~65000
Rt	○	○	○	○	
Rc	○	○	○	○	
WR	○	○	○*	○	

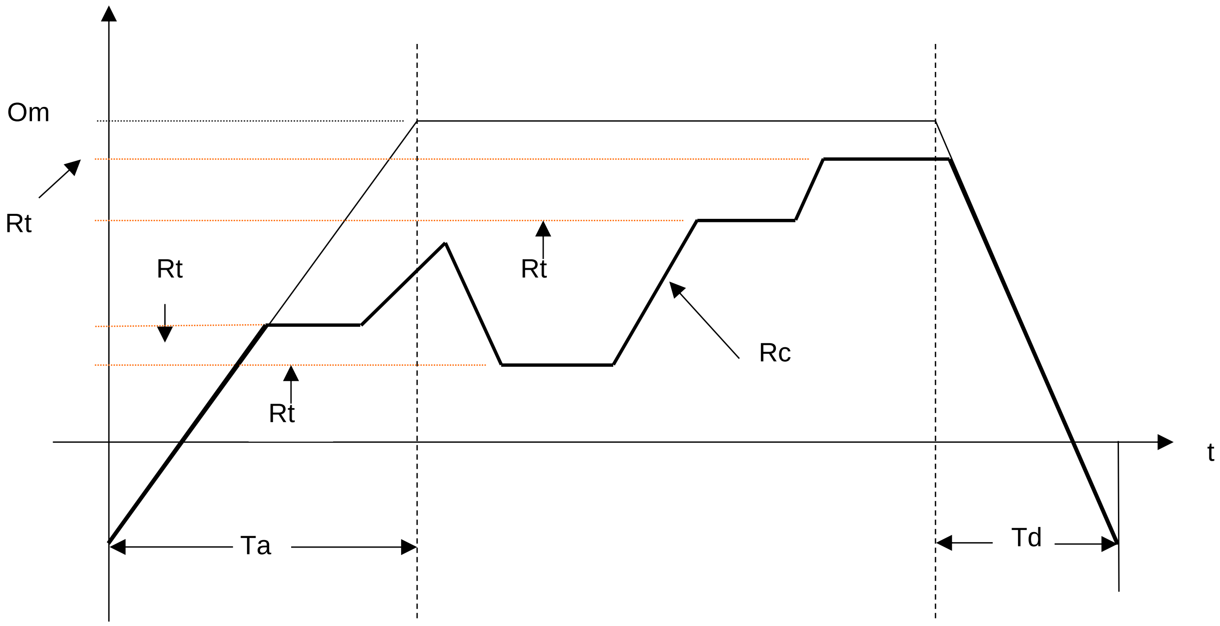
FUN98 RAMP2	TRACKING TYPE RAMP FUNCTION FOR D/A OUTPUT	FUN98 RAMP2
Description		

- When execution 〝EN〞=0, current output value (Rc) will be 0 immediately; the output indicators ACC=0 and DEC=0.
- When execution 〝EN〞=1, this instruction being executed; it will output current value (Rc) first, and then compare the target output value (Rt) with current output value (Rc) every scan; if the target output value is greater than current output value, the current output will be increased according to the rate, which is decided by the settings of acceleration time (Ta) and maximum output (Om), till current output value is equal to the target output value (ACC=1 during this time); if the target output value is less than current output value, the current output will be decreased according to the rate, which is decided by the settings of deceleration time (Td) and maximum output (Om), till current output value is equal to the target output value (DEC=1 during this time).
- If the setting value of target output (Rt) is greater than maximum output(Om), the output value will be clamped by the maximum value.
- It can have smooth activity for acceleration and deceleration control via the execution of this instruction by using current output value (Rc) for analog output (R35024~R35151).。
- The setting value of target output (Rt) needs to stay two scan times at least for proper operation.
- It needs 4 registers for working, they can not be repeated in use.
- This instruction is for positive value operation, but it also can have negative output by short and easy application program for help. Please see example 2.

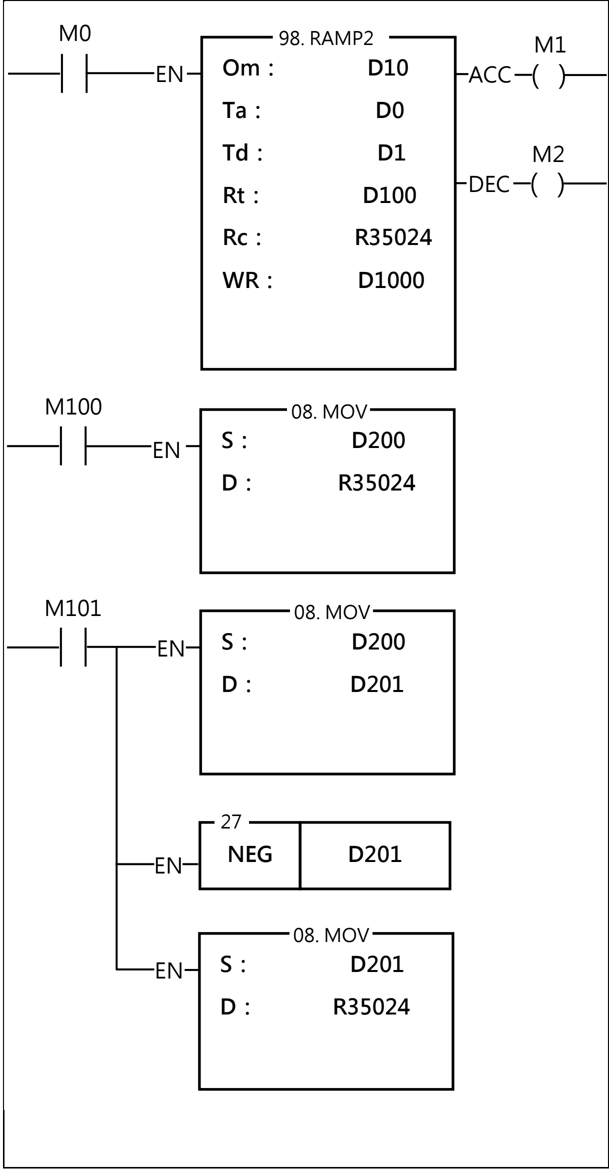
FUN98 RAMP2	TRACKING TYPE RAMP FUNCTION FOR D/A OUTPUT	FUN98 RAMP2
Example 1	Positive output for ACC/DEC control	

Ladder diagram	ST
	RAMP2(EN:= M0, Om:= D10, Ta:= D0, Td:= D1, Rt:= D100, Rc:= R35024, WR:= D1000, ACC=> M0, DEC=> M1);

D10: Setting of maximum output, it is 16383
D0: The acceleration time for the output from 0 up to maximum, it is 30000mS
D1: The deceleration time for the output from maximum down to 0, it is 20000mS
D100: Setting of target output value, it is 8192
R35024: Register of current output, it is used for D/A output
D1000~D1003: Working registers
Description:
When M0=0, current output value is 0 immediately (No ramp). When M0=1, it will output the value of R35024 first; and then compare the target output value (D100) with current output value (R35024) every scan; if D100 > R35024, the current output value of R35024 will be increased according to the rate of 16383/30000 (Om=16383, Ta=30000), till R35024=D100 (ACC=1 during this time); if D100 < R35024, the current output value of R3904 will be decreased according to the rate of 16383/20000 (Om=16383, Td=20000), till R35024=D100 (DEC=1 during this time).



FUN98 RAMP2	TRACKING TYPE RAMP FUNCTION FOR D/A OUTPUT	FUN98 RAMP2
Example 2	Both positive and negative output for ACC/DEC control	

Ladder diagram	ST
	

D10 ：Setting of maximum output, it is 8191

D0 ：The acceleration time for the output from 0 up to maximum, it is 20000mS

D1 ：The deceleration time for the output from maximum down to 0, it is 10000mS

D100：Setting of target output value, it is 0

D200：Register of current output, it is used for analog output

D1000~D1003：Working registers

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

Description：

Description: When M0=0, current output value is 0 immediately (No ramp).
When M0=1, it will output the value of D200 first; and then compare the target output value (D100) with current output value (D200) every scan; if D100 > D200, the current output value of D200 will be increased according to the rate of 8191/20000 (Om=8191, Ta=20000), till D200=D100 (ACC=1 during this time); if D100 < D200, the current output value of D200 will be decreased according to the rate of 8191/10000 (Om=8191, Td=10000), till D200=D100 (DEC=1 during this time).
M100=1, positive output control; M101=1, negative output control.
The target output (D100) is always positive value from 0~65535.

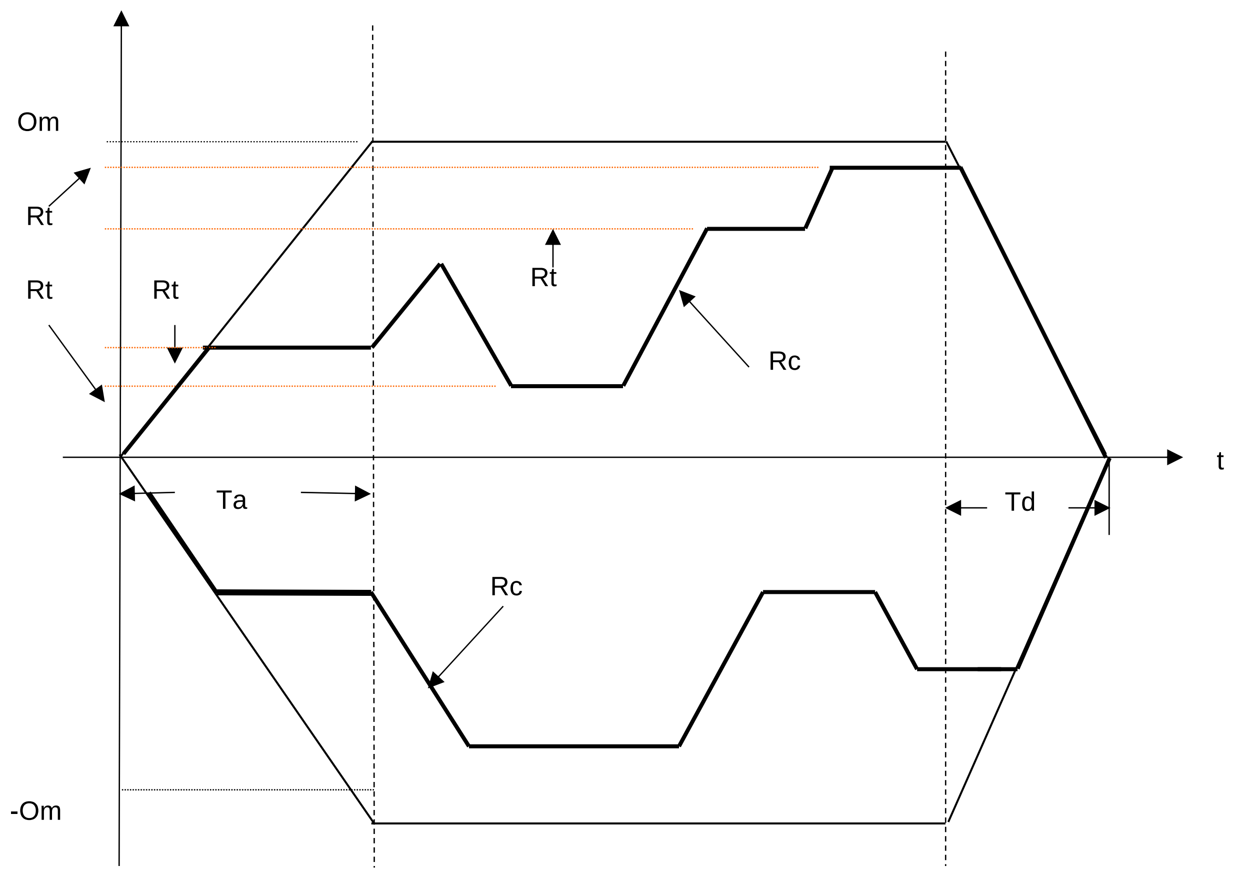
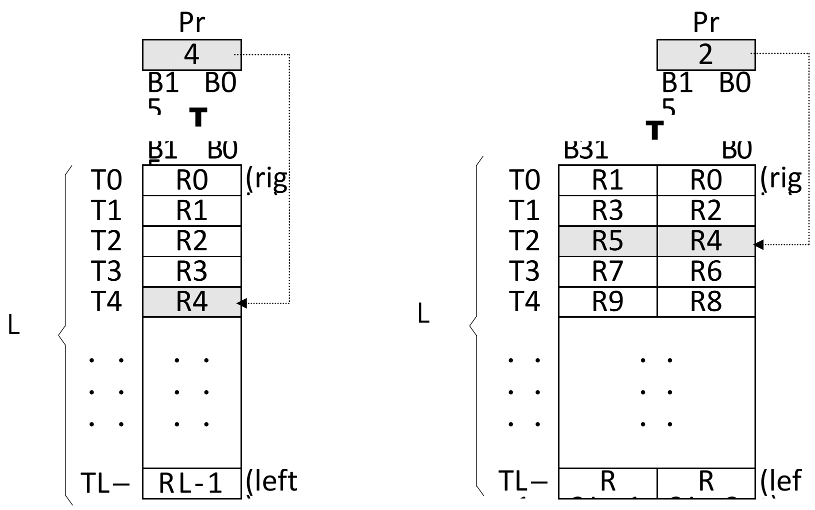


Table Instructions

- 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 T0 to TL-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.



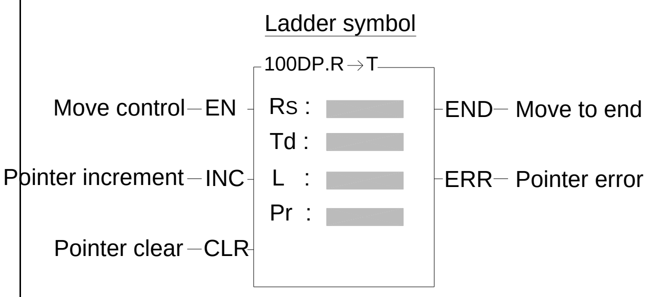
16bit table 32bit table

7-14 Table Instruction (FUN100~114)

7-14-1 REGISTER TO TABLE MOVE

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

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

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WM29584	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34895	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	16/32bit +/- number	V、Z P0~P9
Rs	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Td		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	2~2048	
Pr		○	○	○	○	○	○		○	○*	○*	○		

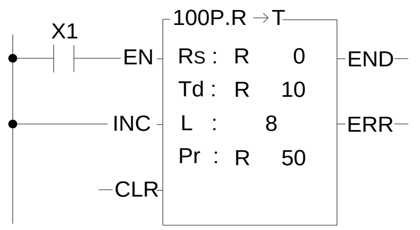
Description

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

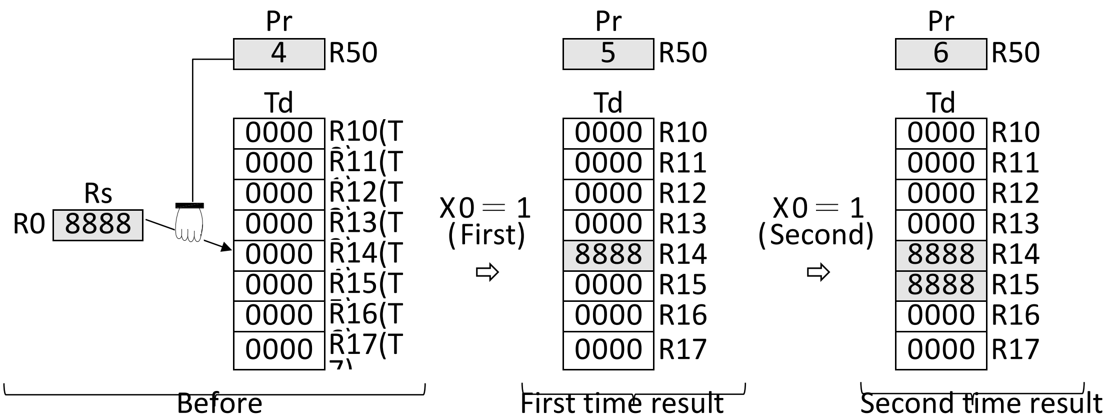
FUN100 D P R→T	REGISTER TO TABLE MOVE	FUN100 D P R→T

- 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.

Example

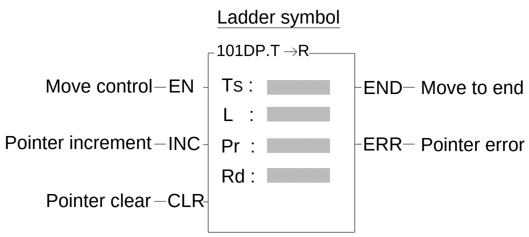
Ladder diagram	ST
	

- 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.



7-14-2 TABLE TO REGISTER MOVE

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



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

Ope-rand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WM29584	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34895	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	16/32bit +/- number	V、Z P0~P9
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
L							○				○*	○		
Pr		○	○	○	○	○	○		○	○*	○*	○	2~2048	
Rd		○	○	○	○	○	○		○	○*	○*	○		○

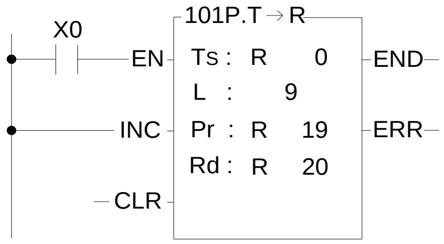
Description	

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

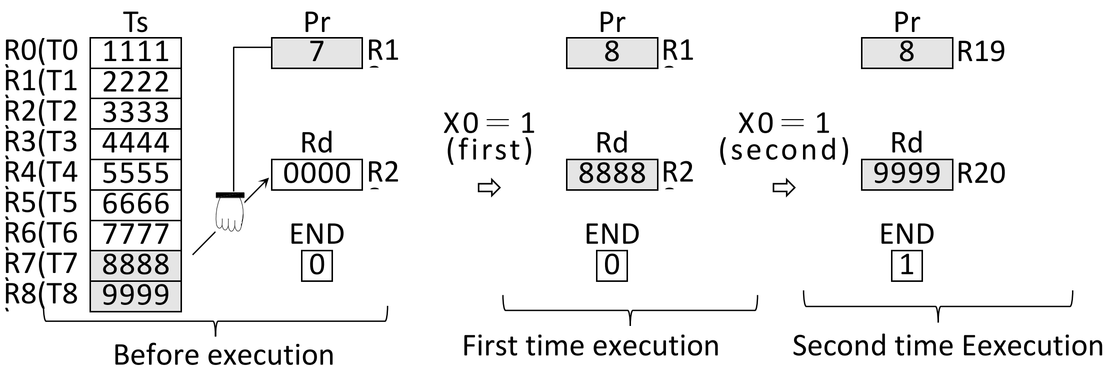
FUN101 D P T→R	TABLE TO REGISTER MOVE	FUN101 D P T→R

- 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.

Example

Ladder diagram	ST
	

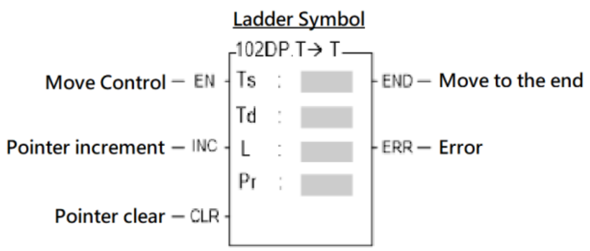
- In the example at left, at the very beginning Pr = 7 and Ts and Rd are as shown at left in the diagram below. When X0 have a transition from 0→1 twice, the results are shown at right in the diagram below.
- At the second time execution, the pointer has already reached to the end so there will be no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



7-14-3 TABLE TO TABLE MOVE

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

Symbol



Ts: the starting number of the register in the source list
Td: the starting number of the register of the destination list
L: list length (Ts and Td)
Pr: index register number
Ts, Td can be combined with V, Z, P0~P9 for indirect addressing applications.

範圍 / 運算元	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 │ WX1008	WY0 │ WY1008	WM0 │ WM29584	WS0 │ WS3088	T0 │ T1023	C0 │ C1279	R0 │ R34767	R34768 │ R34895	R35024 │ R35151	R35280 │ R43223	R43224 │ R47319	D0 │ D11999	2 │ 2048	V、Z P0~P9
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
Td		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	○	
Pr		○	○	○	○	○	○		○	○*	○*	○		

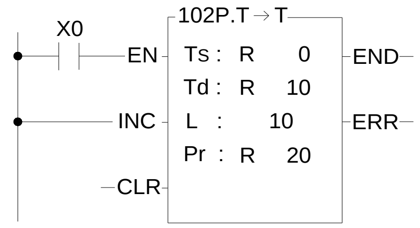
Description

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

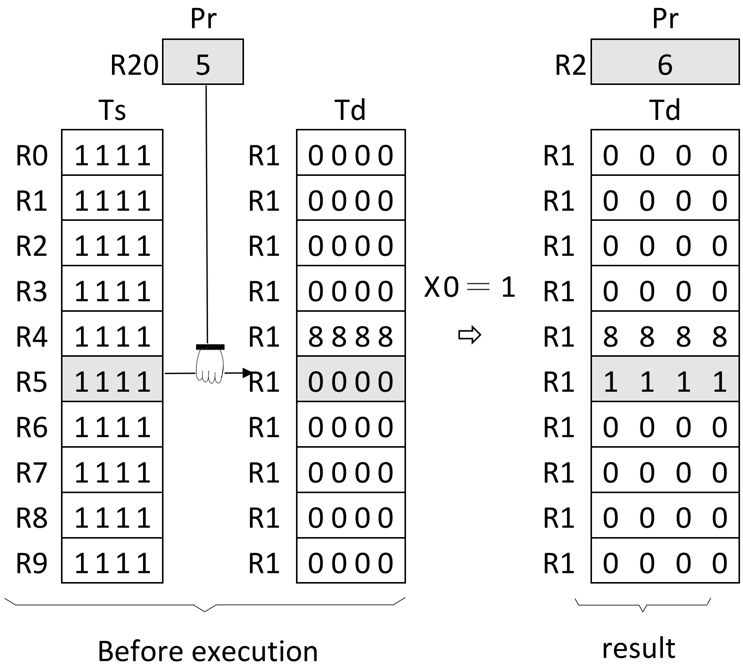
FUN102 D P T→T	(TABLE TO TABLE MOVE)	FUN102 D P T→T

- 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.

Example

Ladder diagram	ST
	

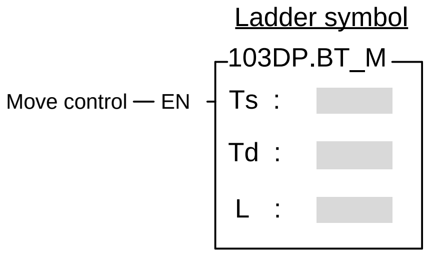
- The diagram at left below is the status before execution. When X0 from 0→1, the content of R5 in Ts table will copy to R15 and pointer R20 will be increased by 1.



7-14-4 BLOCK TABLE MOVE (BT_M)

FUN103 D P BT_M	BLOCK TABLE MOVE	FUN103 D P BT_M

Symbol



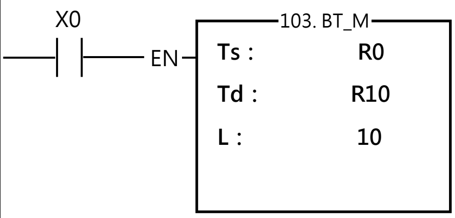
Ts : Starting register for source table
Td : Starting register for destination table
L : Lengths of source and destination tables
Ts, Td may combine with V, Z, P0~P9 to serve indirect

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WY29584	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34895	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	2 ∣ 256	V,Z ∣ P0-P9
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
Td		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	○	

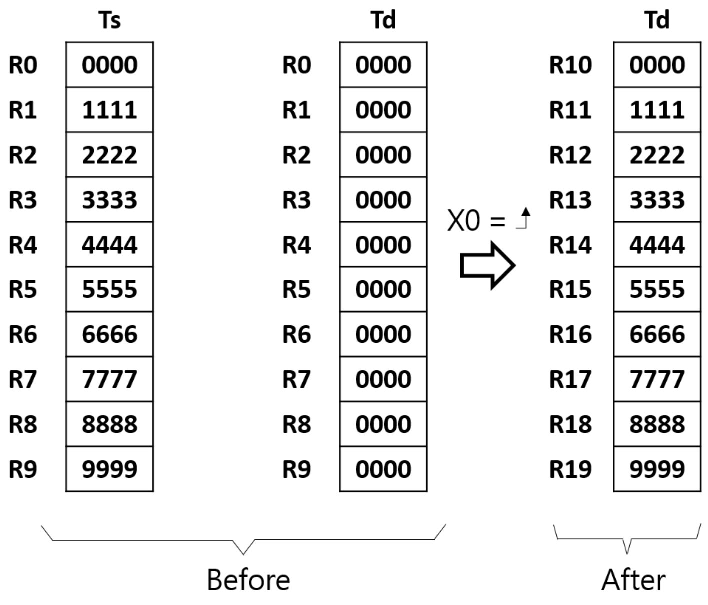
Description

- In this instruction the source table and destination table are the same length. When this instruction was executed all the data in the Ts table is completely copied to Td. No pointer is involved in this instruction.
- When move control "EN" = 1 or have a transition from 0 to 1 (P instruction), all the data from source table Ts (length L) is copied to the destination table Td, which is the same length.
- One table is completely copied every time this instruction is executed, so if the table length is long, it will be very time consuming. In practice, P instruction should be used to avoid time waste caused by each scan repeating the same movement action.

FUN103 D P BT_M	BLOCK TABLE MOVE	FUN103 D P BT_M
Example		

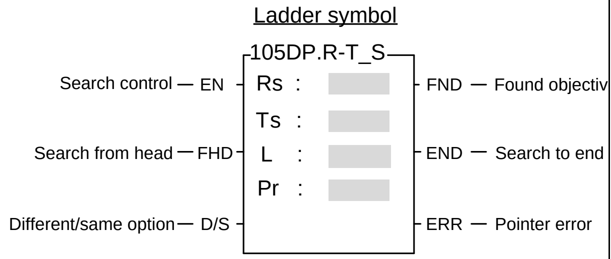
Ladder diagram	ST
	IF X0 THEN BT_M(Ts:= R0, Td:= R10, L:= 10); END_IF

- The program example in the above figure assumes that the state of the TS and TD lists is as shown in the left figure before execution. When X0 changes from 0 to 1, the execution result as shown in the right figure below can be obtained:



7-14-5 REGISTER TO TABLE SEARCH

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



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

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 │ WX1008	WY0 │ WY1008	WM0 │ WM29584	WS0 │ WS3088	T0 │ T1023	C0 │ C1279	R0 │ R34767	R34768 │ R34895	R35024 │ R35151	R35280 │ R43223	R43224 │ R47319	D0 │ D11999	16/32-bit +/- number	V、Z P0~P9
Rs	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
L							○				○*	○	2~256	
Pr		○	○	○	○	○	○		○	○*	○*	○		

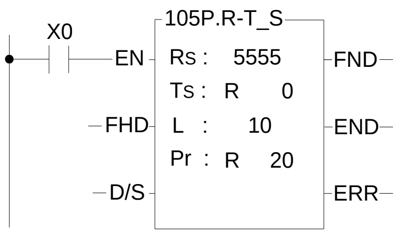
FUN105 D P R-T_S	REGISTER TO TABLE SEARCH	FUN105 D P R-T_S

Description

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

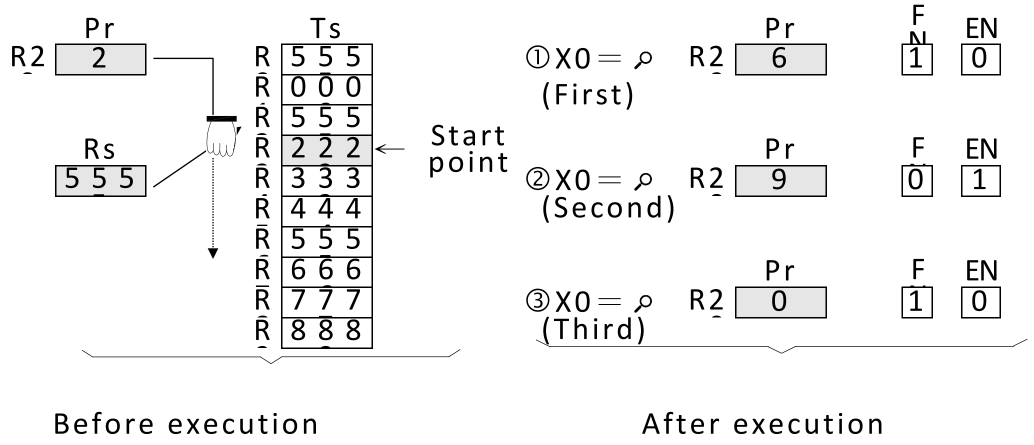
Example

Ladder diagram	ST

	`IF X0 THEN`
	`T_Search( FHD:= FALSE, DS:= FALSE, Rs:= 5555, Ts:= R0, L:= 10, Pr:= R20, FND=> M0, END=> M1, ERR=> M2);`
	`END_IF`

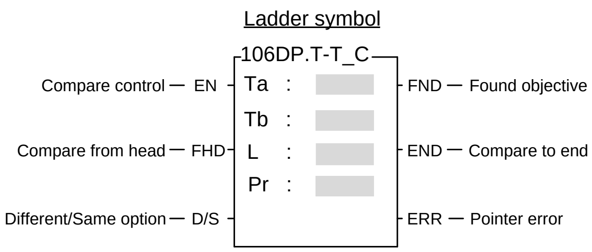
- The instruction at left is searching the table for a register with the value 5555 (because D/S = 0, it is searching for same value). Before execution, the pointer point to R2, but the starting point of the search is Pr + 1 (i.e. it starts from R3). After X0 has transition from 0→1 3 times, the results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

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



7-14-6 TABLE TO TABLE COMPARE

FUN106 DP T-T_C	TABLE TO TABLE COMPARE	FUN106 DP T-T_C
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

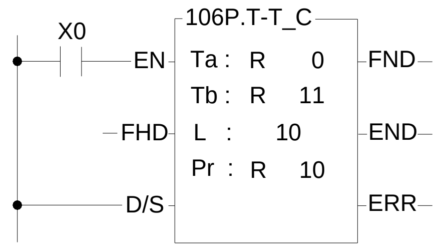
Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WM29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	2 | 256	V、Z P0~P9
Ta	○	○	○	○	○	○	○	○	○	○	○	○		○
Tb	○	○	○	○	○	○	○	○	○	○	○	○		○
L							○				○*	○	○	
Pr		○	○	○	○	○	○		○	○*	○*	○		

Description	

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

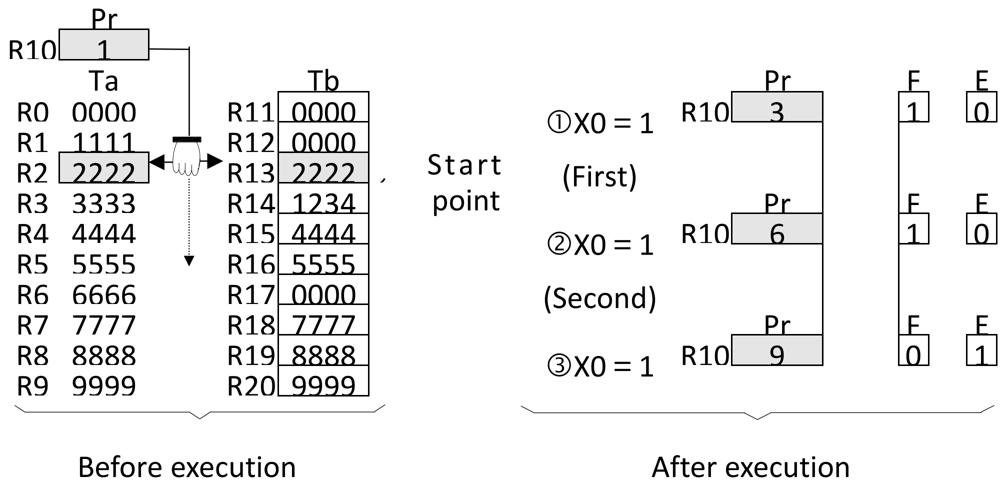
- When comparison control "EN" = 1 or has a transition from 0 to 1(P instruction), then starting from the first register in the tables Ta and Tb (when "FHD" = 1 or Pr value has reached L-1) or starting from the next pair of registers (Tapr+1 and Tbpr+1) pointed by Pr ("FHD" = 0, while Pr is less than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

Example

Ladder diagram	ST
	IF X0 THEN T_Compare(FHD:= FALSE, DS:= TRUE, Ta:= R0, Tb:= R11, L:= 10, Pr:= R10, FND=> M0, END=> M1, ERR=> M2); END_IF

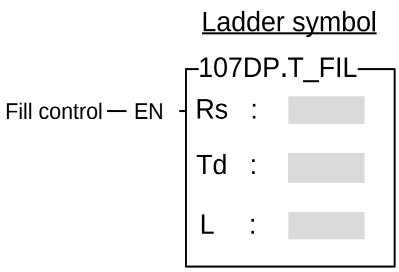
- The instruction at right starts from the register next to the register pointed by the pointer (because "FHD" is 0) to search for register pairs with different data (because "D/S" is 1) within the 2 tables. At the very beginning, Pr points to Ta1 and Tb1. There are 3 different pairs of data at the position 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.

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



7-14-7 TABLE FILL (T_FIL)

FUN107 D P T_FIL	TABLE FILL	FUN107 D P T_FIL
Symbol		



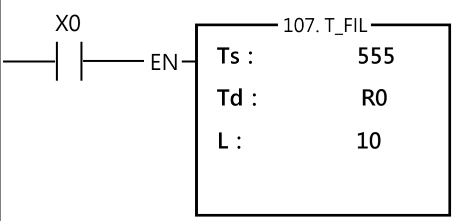
Rs : Source data to fill, can be a constant or a register
Td : Starting register of destination table
L :Table length
Rs, Td may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +- numbers	V,Z P0-P9
Rs	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Td		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	2-256	

Description	

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

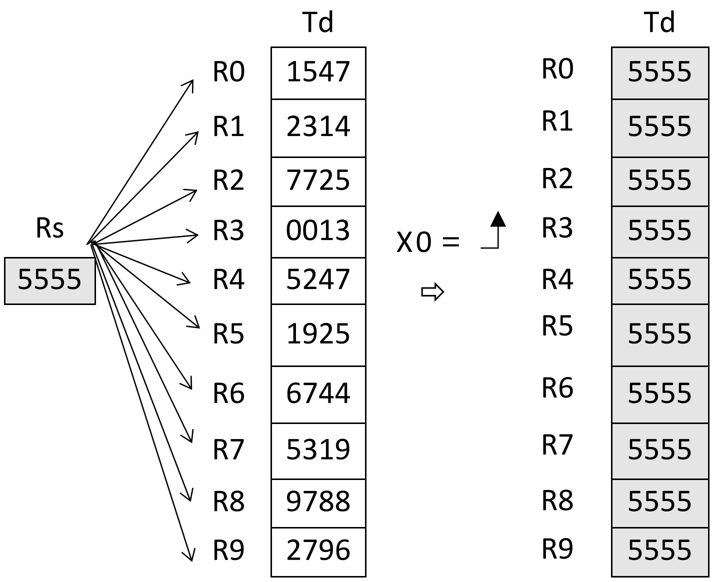
<table>
<tr><td>FUN107 D P
T_FIL</td><td>TABLE FILL</td><td>FUN107 D P
T_FIL</td></tr>
<tr><td>Example</td><td colspan="2"></td></tr>
</table>

<table>
<tr><th>Ladder diagram</th><th>ST</th></tr>
<tr><td></td><td>

```
IF X0 THEN
T_Fill( Rs:= 555, Td:= R0, L:= 10);
END_IF
```

</td></tr>
</table>

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

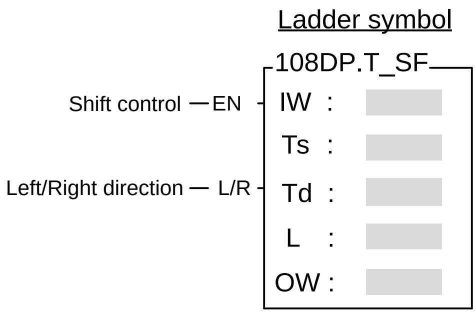


Before execution After execution

7-14-8 TABLE SHIFT

FUN108 DP T_SHF	TABLE SHIFT	FUN108 DP T_SHF

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

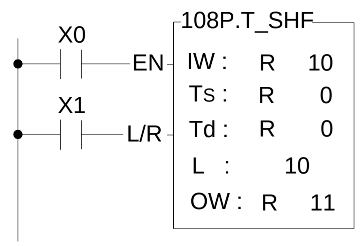
Ts, Td may combine with V, Z, P0~P9 to serve indirect address application

Operand- \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WM29584	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34895	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	16/32-bit +/- number	V、Z P0~P9
IW	○	○	○	○	○	○	○	○	○	○	○	○	○	
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
Td		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	2~256	
OW		○	○	○	○	○	○		○	○*	○*	○		

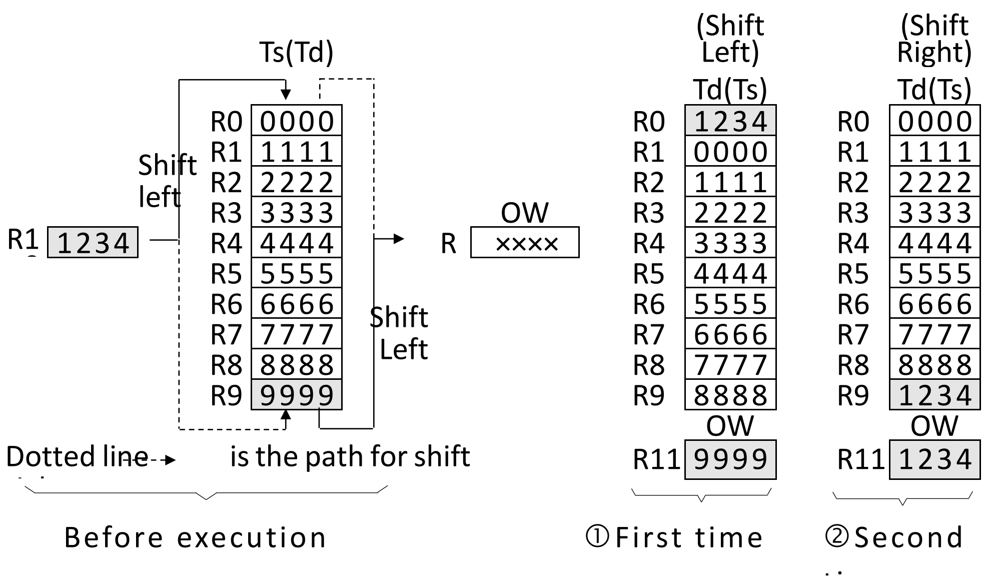
Description

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

FUN108 D P T_SHF	TABLE SHIFT	FUN108 D P T_SHF
Example		

Ladder diagram	ST
	

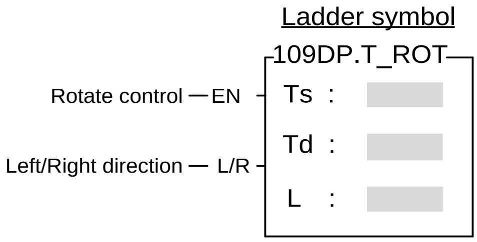
- In the program at left, Ts and Td is the same table. Therefore, the table shifts itself and then writes back to itself (the table must be writ able). It first perform a shift left operation (let X1 = 1, and X0 go from 0→1) then perform a shift to right operation (let X1 = 0, and makes X0 go from 0→1).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



7-14-9 TABLE ROTATE

FUN109 D P T_ROT	TABLE ROTATE	FUN109 D P T_ROT

Symbol



Ts : Source table for rotate

Td: Destination table storing results of rotation

L : Lengths of table

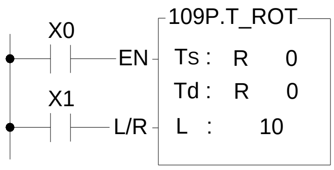
Ts, Td may combine with V, Z, P0~P9 to serve indirect address application

Operand \ Range	WX	WY	WM	WS	TM	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 │ WX1008	WY0 │ WY1008	WM0 │ WM29584	WS0 │ WS3088	T0 │ T1023	C0 │ C1279	R0 │ R34767	R34768 │ R34895	R35024 │ R35151	R35280 │ R43223	R43224 │ R47319	D0 │ D11999	2 │ 256	V、Z P0~P9
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
Td		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	○	

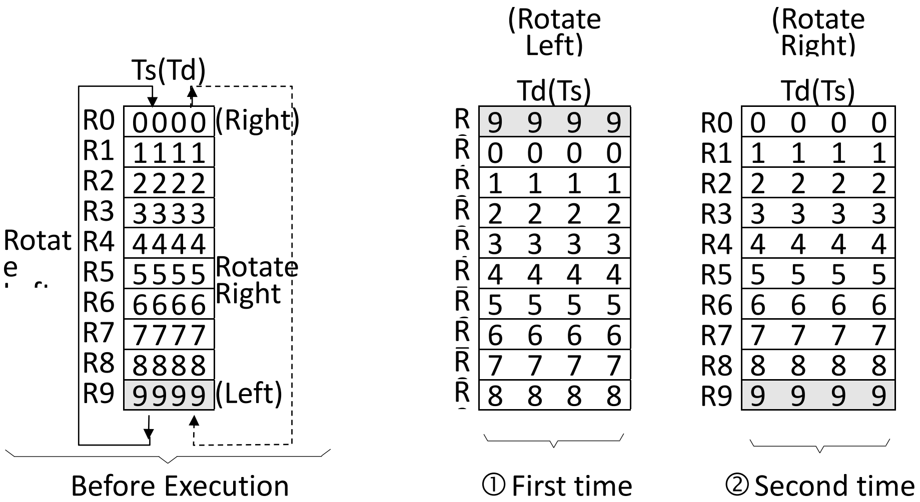
Description

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

FUN109 D P T_ROT	TABLE ROTATE	FUN109 D P T_ROT
Example		

Ladder diagram	ST
	

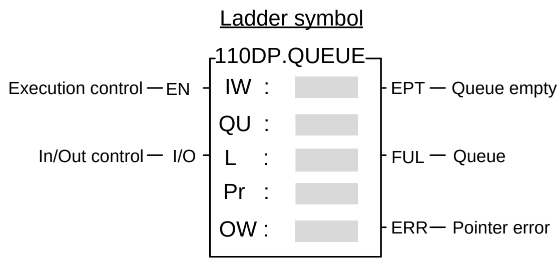
- In the program at left, Ts and Td is the same table. The table after rotation will write back to itself. It first perform one left rotation (let X1 = 1, and X0 go from 0→1), and then performs one right rotation (let X1 = 0, and X0 go from 0→1). The results are shown at right in the diagram below.



7-14-10 QUEUE

FUN110 D P QUEUE	QUEUE	FUN110 D P QUEUE

Symbol



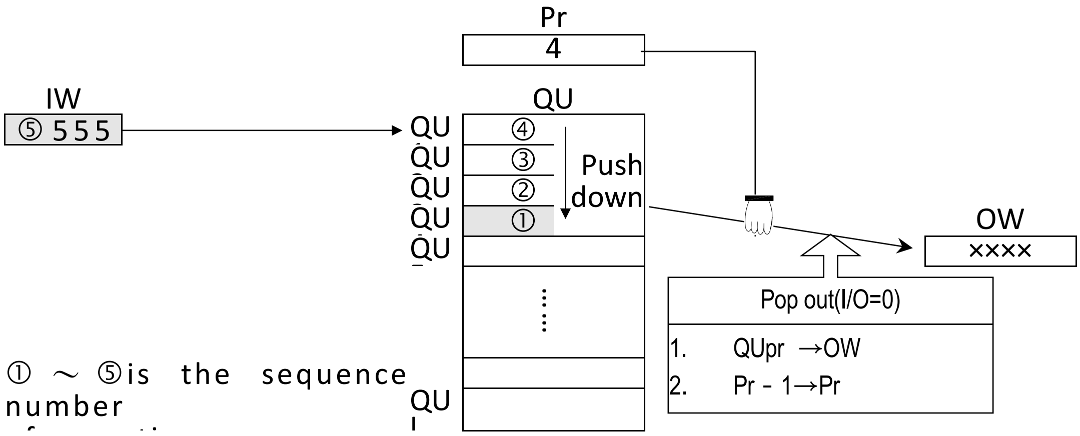
IW: THE DATA TO BE INSERTED INTO THE STORAGE COLUMN OR ITS REGISTER NUMBER
QU: THE STARTING REGISTER NUMBER OF THE STORAGE ROW
L: THE LENGTH OF THE STORAGE COLUMN
PR: INDEX REGISTER NUMBER
OW: REGISTER NUMBER RECEIVING DATA MOVED OUT FROM THE MEMORY
QU CAN BE COMBINED WITH V, Z, P0~P9 FOR INDIRECT ADDRESSING APPLICATION

Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WM29584	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34895	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	16/32-bit +/- number	V、Z P0~P9
IW	○	○	○	○	○	○	○	○	○	○	○	○	○	
QU		○	○	○	○	○	○		○	○	○*	○		○
L							○				○*	○	2~256	
Pr		○	○	○	○	○	○		○	○*	○*	○		
OW		○	○	○	○	○	○		○	○*	○*	○		

Description

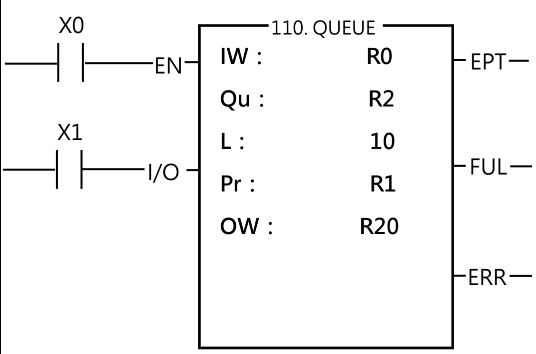
- Queue is also a kind of table. It is different from ordinary table in that its queue register numbers go from 1 to L and not from 0 to L-1. In other words, QU1~QUL respectively correspond to pointers Pr = 1 to L, and Pr = 0 is used to show that the queue is empty.
  Note: The system uses the serial number L+1 register for internal calculations, users must avoid this register
- 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.

FUN110 D P QUEUE	QUEUE	FUN110 D P QUEUE



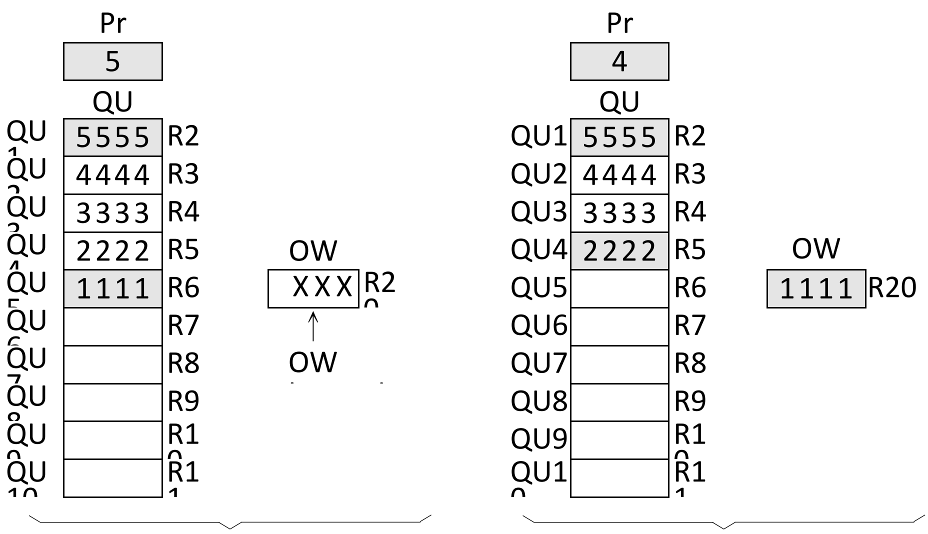
- When execution control "EN" = 1 or has a transition from 0 to 1 (P instruction), the status of in/out control "I/O" determines whether the IW data will be pushed into the queue (when "I/O" = 1) or be popped out and transferred to OW (when "I/O" = 0). As shown in the diagram above, the IW data will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

FUN110 D P QUEUE	QUEUE	FUN110 D P QUEUE
Example		

Ladder diagram	ST
	see code below


```
IF X0 THEN
QUEUE( InOut:= X1,
       IW:= R0,
       Qu:= R2,
       L:= 10,
       Pr:= R1,
       OW:= R20,
       EPT=> M0,
       FUL=> M1);
END_IF
```

- The program above 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.

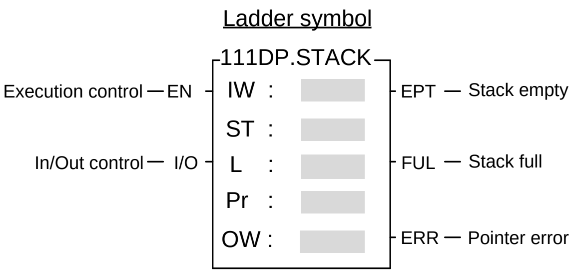


After push in (X1=1，X0 from 0→1)　　　After pop off(X1=0，X0 from 0→1)

7-14-11 STACK

FUN111 D P STACK	STACK	FUN111 D P STACK

Symbol



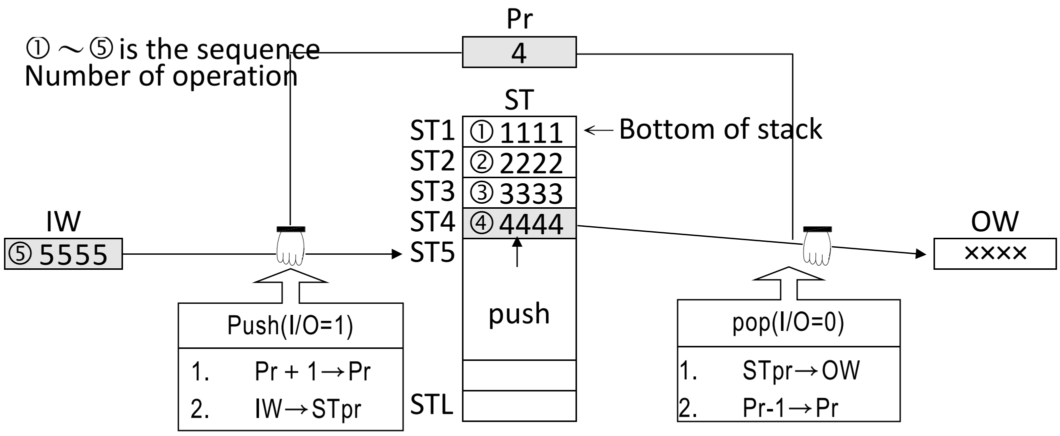
IW : Data pushed into stack, can be a constant or a register
ST : Starting register of stack
L : Size of stack
Pr : Pointer register
OW : Register accepting data popped out from stack
ST may combine with V, Z, P0~P9 to serve indirect address application

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 WX1008	WY0 WY1008	WM0 WM29584	WS0 WS3088	T0 T1023	C0 C1279	R0 R34767	R34768 R34895	R35024 R35151	R35280 R43223	R43224 R47319	D0 D11999	16/32-bit +/- number	V、Z P0~P9
IW	○	○	○	○	○	○	○	○	○	○	○	○	○	
ST		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	2~256	
Pr		○	○	○	○	○	○		○	○*	○*	○		
OW		○	○	○	○	○	○		○	○*	○*	○		

Description

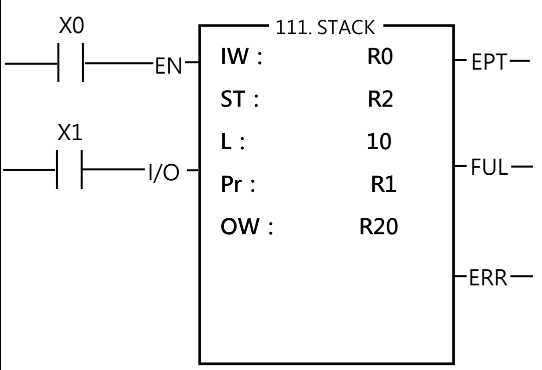
- Like queue, stack is also a kind of table. The nature of its pointer is exactly the same as with queue, i.e. Pr = 1 to L, which corresponds to ST1 to STL, and when Pr = 0 the stack is empty.
- Stack is the opposite of queue, being a last in first out (LIFO) device. This means that the data that was 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:

FUN111 D P STACK	STACK	FUN111 D P STACK

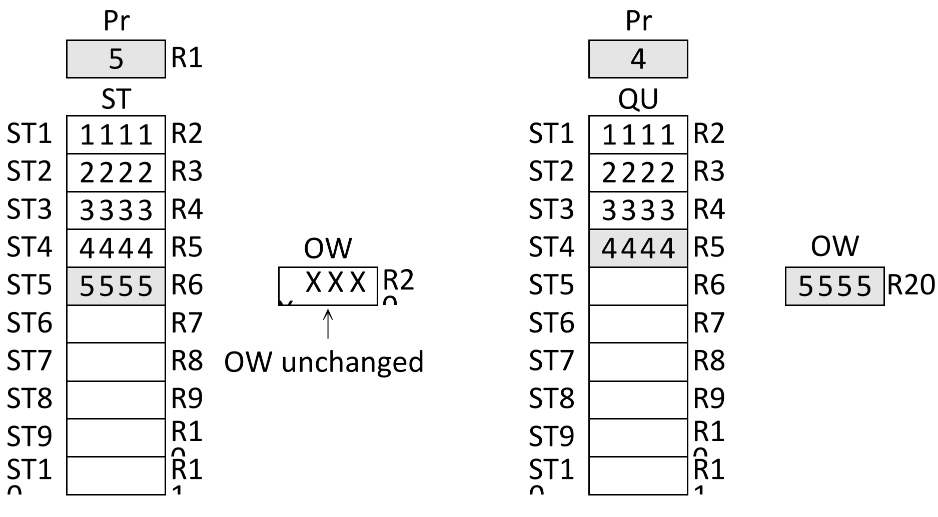


- When execution control "EN" = 1 or has a transition from 0 to 1(P instruction), the status of in/out control "I/O" determines whether the IW data will be pushed into the stack (when "I/O" = 1), or the data pointed by Pr within the stack (the data most recently pushed into the stack) will be moved out and transferred to OW (when "I/O" = 0). Note that the data pushed in is stacking, so before pushed in, Pr will increased by 1 to point to the top of the stack then the data will be pushed in. When it is popped out, the data pointed by pointer Pr (the most recently pushed in data) will be transferred to OW. After then Pr will decreased by 1. Under any circumstances, the pointer Pr will always point to the data that was pushed into the stack most recently.
- 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 STL 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 ST1 to STL). Beyond this range, the pointer error flag "ERR" will set to 1, and the instruction will not be carried out.

FUN111 D P STACK	STACK	FUN111 D P STACK
Example		

Ladder diagram	ST
	IF X0 THEN STACK(InOut:= X1, IW:= R0, ST:= R2, L:= 10, Pr:= R1, OW:= R20, EPT=> M0, FUL=> M1); END_IF

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

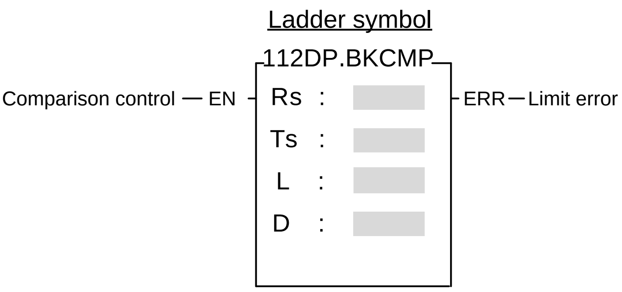


After push(X1=1，X0 from 0→1)　　After pop(X1=0，X0 from 0→1)

7-14-12 BLOCK COMPARE (DRUM)

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

Symbol



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

Operand \ Range	Y	M	S	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	Y0 – Y1023	M0 – M19583	S0 – S3103	WX0 – WX1008	WY0 – WY1008	WM0 – WM29584	WS0 – WS3088	T0 – T1023	C0 – C1279	R0 – R34767	R34768 – R34895	R35024 – R35151	R35280 – R43223	R43224 – R47319	D0 – D11999	16/32-bit +/-number
Rs				○	○	○	○	○	○	○	○	○	○	○	○	○
Ts				○	○	○	○	○	○	○	○	○	○	○	○	
L										○				○*	○	1~256
D	○	○	○													

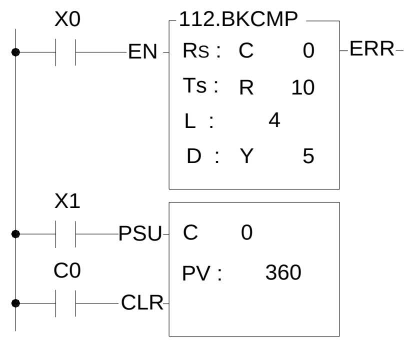
Description

- When comparison control "EN" = 1 or has a transition from 0 to 1(P instruction), comparisons will be perform one by one between the contents of Rs and the upper and lower limits form by L pairs of 16 or 32-bit (D modifier) registers starting from the Ts register (starting from T0 each adjoining 2 register units form a pair of upper and lower limits). If the value of Rs falls within the range of the pair, then the bit within the comparison results relay D which corresponds to that pair will be set to 1. Otherwise it will be set as 0 until comparison of all the L pairs of upper and lower limits is completed.
- When M9160=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 M9160=1, there is no restriction on the relation of upper limit and lower limit, this can apply for 360°rotary electronic drum switch application.

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

	Upper limit	Lower limit	Compared ← →	Compare value	Result ← →	Output
0	T_{S1}	T_{S0}				D0
1	T_{S3}	T_{S2}	←	R_S	←→	D1
≀	≀	≀	≀		≀	≀
L−1	T_{S2L-1}	T_{S2L-2}				D_{L-1}

- 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.

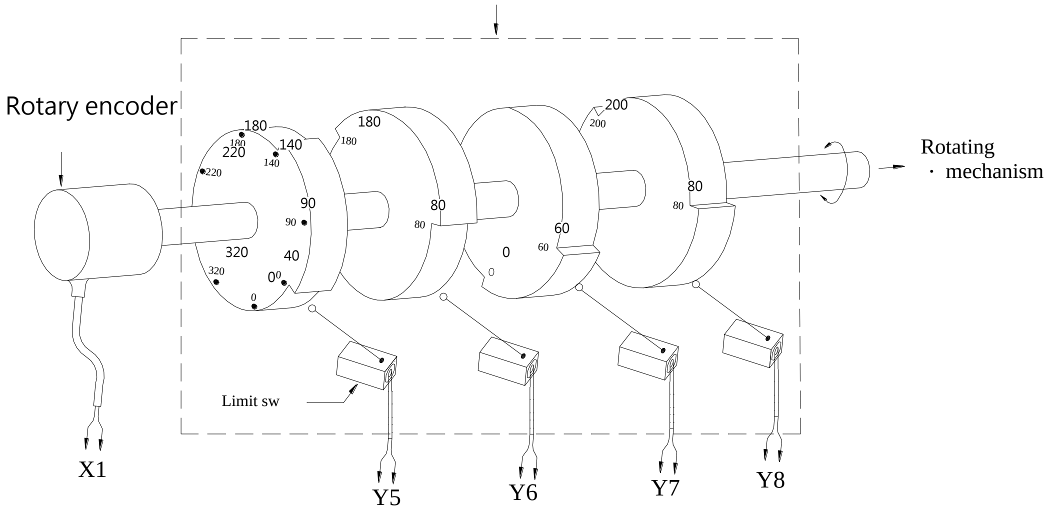
Ladder diagram	ST
	IF X0 THEN BKCMP(Rs:= C0, Ts:= R10, L:= 4, D:= Y5); END_IF Counter(Pulse:= X1, Clr:= C0, C:= 0, PV:=360, IsUp=> C0);

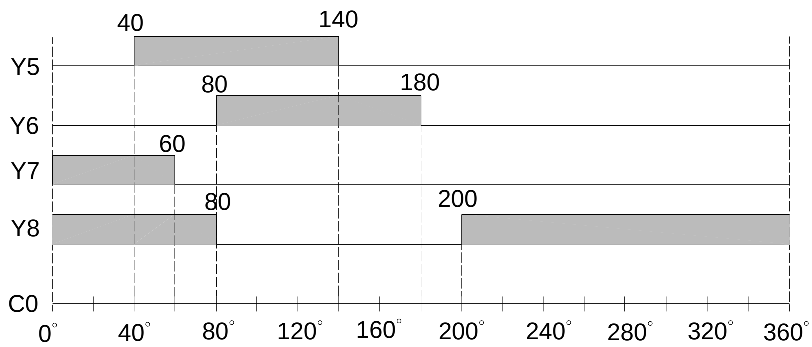
- In this program, C0 represents the rotation angle (Rs) of a drum shaft. The block compare instruction performs a comparison between Rs and the 4 pairs (L = 4) of upper and lower limits, R10,R11, R12,R13, R14,R15 and R16,R17. The comparison results can be obtained from the four drum output points Y5 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

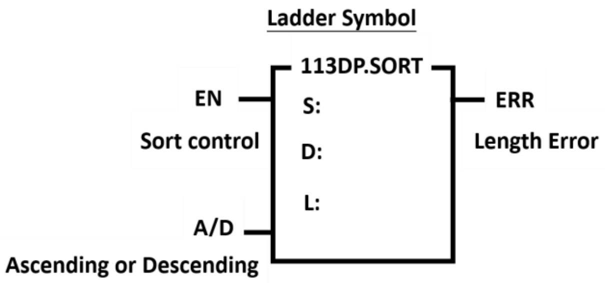




7-14-13 DATA SORTING (SORTING)

FUN113 D P SORT	SORTING	FUN113 D P SORT

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 / Ope-rand	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	2 | 256
S	○	○	○	○	○	○	○	○	
D			○				○*	○	
L			○				○	○	○

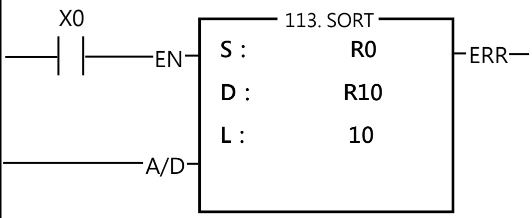
Description

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

Example

FUN113 D P SORT	DATA SORTING	FUN113 D P SORT

Ladder diagram



ST

```
IF X0 THEN
SORT( AD:= TRUE,
      S:= R0,
      D:= R10,
      L:= 10);
END_IF
```

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

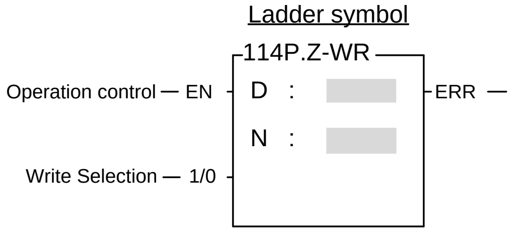
X0 = ↑ ⇨

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

7-14-14 ZONE WRITE

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

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	SM	S	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	K	XR
	Y0 │ Y1023	M0 │ M19583	M9120 │ M29599	S0 │ S3103	WY0 │ WY1008	WM0 │ WM29584	WS0 │ WS3088	T0 │ T1023	C0 │ C1279	R0 │ R34767	R35024 │ R35151	R35280 │ R43223	R43224 │ R47319	D0 │ D11999		V 、Z P0 ~ P9
D	○	○		○	○	○	○	○	○	○	○	○	○	○		○
N										○			○	○	1-511	○

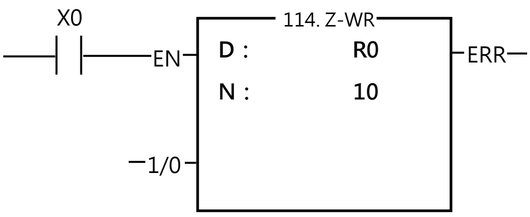
Description

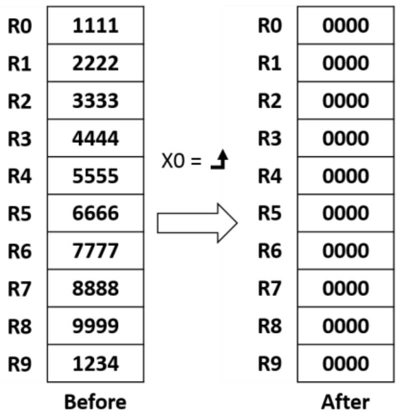
- When operation control "EN"=1 or changes from 0→1（ P instruction）, it will perform the write operation according to the input status of write selection, the specified area of registers or bits will all be reset to 0 ("1/0"=0) or set to 1("1/0"=1).
- The valid data length of sort operation is between 0 and 511, other length will set the “ERR” to 1 and the sort operation will not perform.

Example 1	Registers R0~R9 will be reset to 0 while X0=1

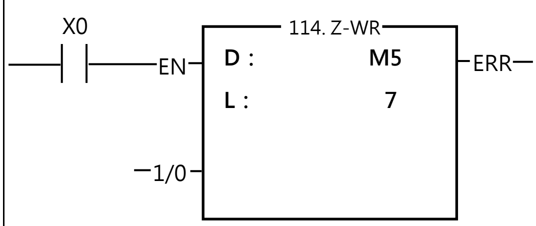
FUN114 D P Z-WR	ZONE WRITE	FUN114 D P Z-WR

Ladder diagram	ST

	IF X0 THEN
	ZoneWR(Val:= FALSE,
	       D:= R0,
	       N:= 10);
	END_IF



Example 2	(When X0 is “ON”, clear M5~M11 to 0)

<table>
<tr><th>Ladder diagram</th><th>ST</th></tr>
<tr><td>
</td><td>

```
IF X0 THEN
ZoneWR( Val:= FALSE,
       D:= M5,
       N:= 7);
END_IF
```

</td></tr>
</table>

M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Before

X0 = ↑

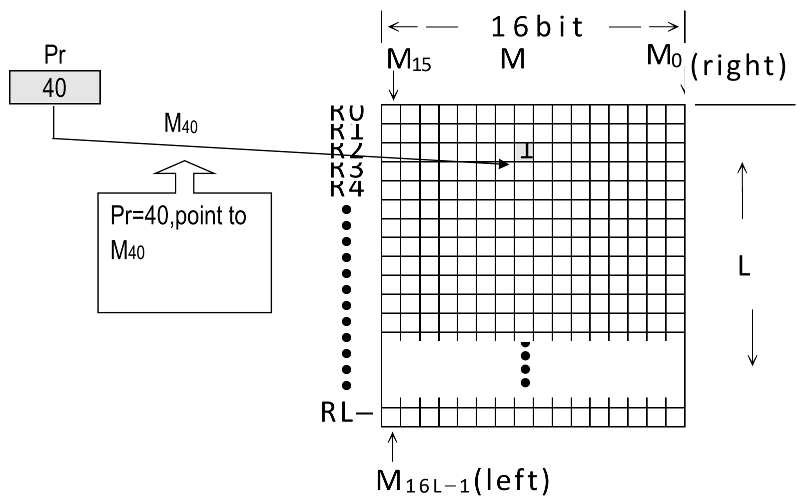
M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1

After

Matrix Instructions

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

- 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 M0 to M16L-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 M0 to M16L-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.



7-15 Matrix Instruction (FUN120~130)

7-15-1 MATRIX AND

FUN120 P MAND	MATRIX AND	FUN120 P MAND
Symbol		

Ladder symbol

```
                      120P.MAND
Operation control ─EN ┤ Ma :
                        Mb :
                        Md :
                        L  :
```

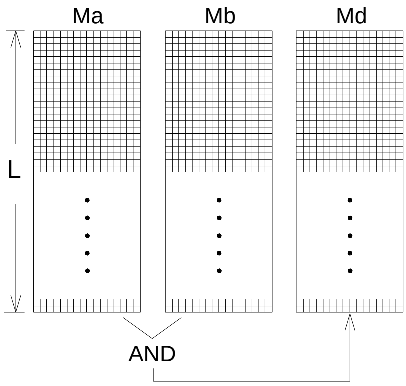
Ma : Starting register of source matrix a
Mb : Starting register of source matrix b
Md : Starting register of destination matrix
L : Length of matrix (Ma, Mb and Md)
Ma, Mb, Md may combine with V, Z, P0~P9 to serve indirect address application

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WM[illegible]	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34[illegible]	R35024 ∣ R35[illegible]	R35280 ∣ R43[illegible]	R43224 ∣ R47[illegible]	D0 ∣ D11999	2 ∣ 256	V、Z P0~P[illegible]
Ma	○	○	○	○	○	○	○	○	○	○	○	○		○
Mb	○	○	○	○	○	○	○	○	○	○	○	○		○
Md		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	○	

Description	

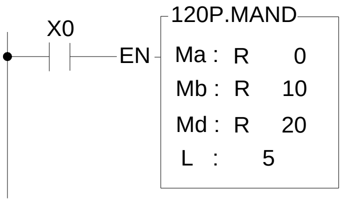
- When operation control "EN" = 1 or has a transition from 0 to 1 (P instruction), this instruction will perform a logic AND (only if 2 bits are 1 will the result be 1, otherwise it will be 0)operation between two source matrixes with a length of L, Ma and Mb. The result will then be stored in the destination matrix Md, which is also the same length (the AND operation is done by bits with the same bit numbers). For example, if Ma0 = 0, Mb0 = 1, then Md0 = 0; if Ma1 = 1, Mb1 = 1, then Md1 = 1; etc, right up until AND reaches Ma16L-1 and Mb16L-1.

FUN120 P MAND	MATRIX AND	FUN120 P MAND

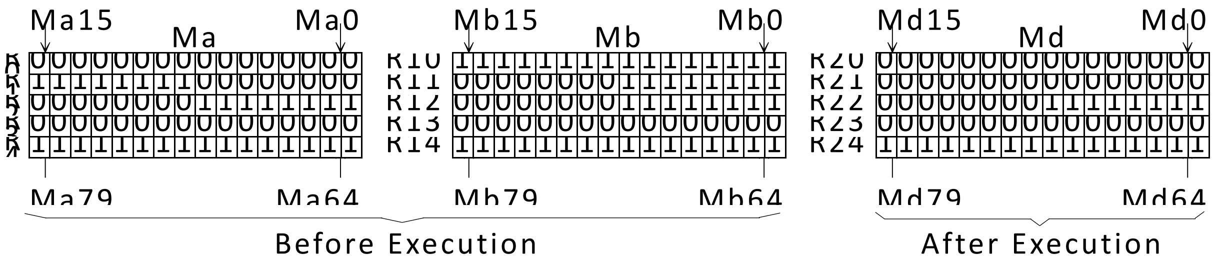


Example	

Ladder diagram	ST

	`IF X0 THEN`
	`MAND( Ma:= R0, Mb:= R10, Md:= R20,`
	`L:= 5);`
	`END_IF`

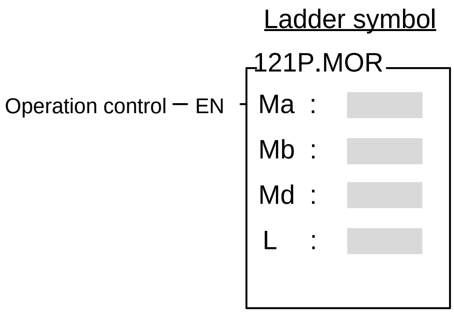
- In the program at left, when X0 goes from 0→1, then matrix Ma, comprised by R0 to R4, and matrix Mb, comprised by R10 to R14, will do an AND operation. The results will be stored back in matrix Md, comprised by R20 to R24. The result is shown at right in the diagram below.



7-15-2 MATRIX OR

FUN121 P MOR	MATRIX OR	FUN121 P MOR

Symbol



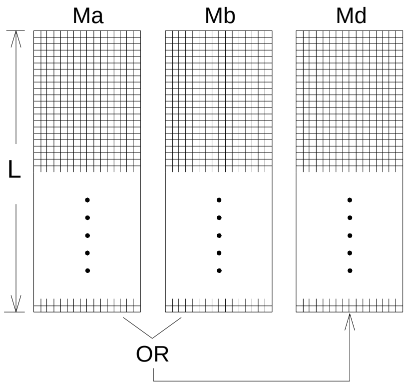
Ma : Starting register of source matrix a
Mb : Starting register of source matrix b
Md : Starting register of destination matrix
L : Length of matrix (Ma, Mb and Md)
Ma, Mb, Md may combine with V, Z, P0~P9 to serve indirect address application

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ~ WX1008	WY0 ~ WY1008	WM0 ~ WM29584	WS0 ~ WS3088	T0 ~ T1023	C0 ~ C1279	R0 ~ R34767	R34768 ~ R34895	R35024 ~ R35151	R35280 ~ R43223	R43224 ~ R47319	D0 ~ D11999	2 ~ 256	V、Z P0~P9
Ma	○	○	○	○	○	○	○	○	○	○	○	○		○
Mb	○	○	○	○	○	○	○	○	○	○	○	○		○
Md		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	○	

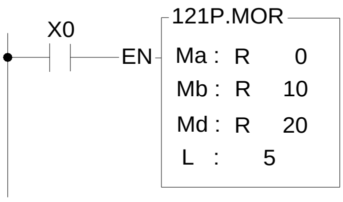
Description

- When operation control "EN" = 1 or has a transition from 0 to 1 (P instruction), this instruction will perform a logic OR(If any 2 of the bits are 1, then the result will be 1, and only if both are 0 will the result be 0) operation between 2 source matrixes with a length of L, Ma and Mb. The result will then be stored in the destination matrix Md, which is also the same length (the OR operation is done by bits with the same bit numbers). For example, if Ma0 = 0, Mb0 = 1, then Md0 = 1; if Ma1 = 0, Mb1 = 0, then Md1 = 0; etc, right up until OR reaches Ma16L-1 and Mb16L-1.

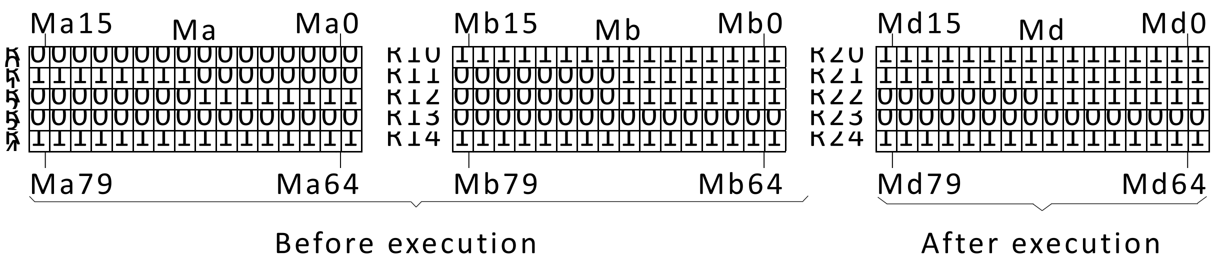
FUN121 P MOR	MATRIX OR	FUN121 P MOR



Example

Ladder diagram	ST
	IF X0 THEN MOR(Ma:= R0, Mb:= R10, Md:= R20, L:= 5); END_IF

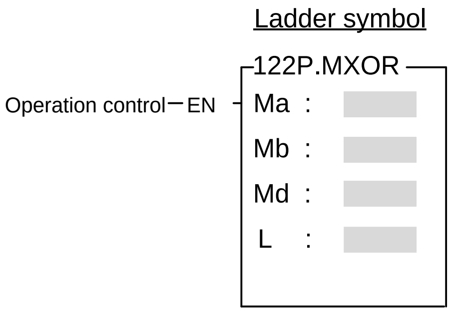
- In the program at left, when X0 goes from 0→1, then matrix Ma, comprised by R0 to R4, and matrix Mb, comprised by R10 to R14, will do an OR operation. The results will then be stored into the destination matrix Md, comprised by R10 to R14. In this example, Mb and Md is the same matrix, so after operation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



7-15-3 MATRIX EXCLUSIVE OR

FUN122 P MXOR	MATRIX EXCLUSIVE OR	FUN122 P MXOR

Symbol



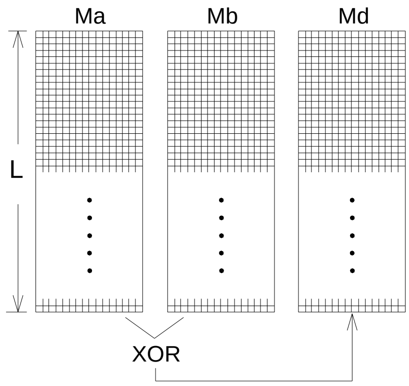
Ma : Starting register of source matrix a
Mb : Starting register of source matrix b
Md : Starting register of destination matrix
L : Length of matrix (Ma, Mb and Md)
Ma, Mb, Md may combine with V, Z, P0~P9 to serve indirect address application

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WM29584	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34895	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	2 ∣ 256	V、Z P0~P9
Ma	○	○	○	○	○	○	○	○	○	○	○	○		○
Mb	○	○	○	○	○	○	○	○	○	○	○	○		○
Md		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	○	

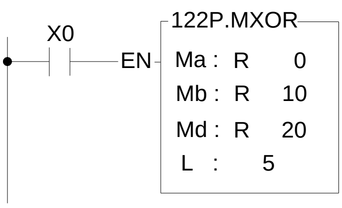
Description

- When operation control "EN" = 1 or has a transition from 0 to 1 (P instruction), this instruction will performs a logic XOR (if the 2 bits are different, then the result will be 1, otherwise it will be 0)between 2 source matrixes with a length of L, Ma and Mb. The result will then be stored back into the destination matrix Md, which also has a length of L. For example the XOR operation is done by bits with the same bit numbers - for example, if Ma0 = 0, Mb0 = 1, then Md0 = 1; if Ma1 = 1, Mb1 = 1, then Md1 = 0; etc, right up until XOR reaches Ma16L-1 and Mb16L-1.

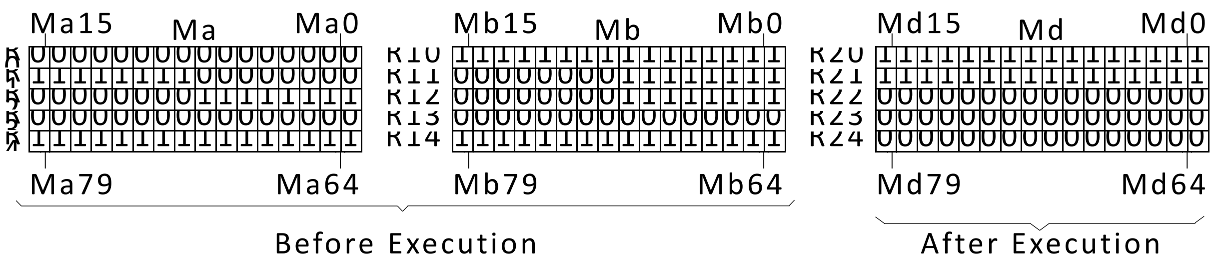
FUN122 P MXOR	MATRIX EXCLUSIVE OR	FUN122 P MXOR



Example	

Ladder diagram	ST
	IF X0 THEN MXOR(Ma:= R0, Mb:= R10, Md:= R20, L:= 5); END_IF

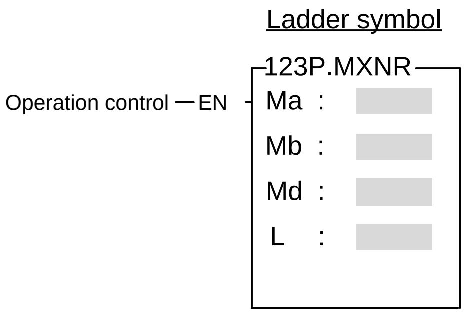
- In the program at left, when X0 goes from 0→1, will perform a XOR operation between matrix Ma, comprised by R0 to R4, and matrix Mb, comprised by R10 to R14. The results will then be stored in destination matrix Md, comprised by R20 to R24. The results are shown at right in the diagram below.



7-15-4 MATRIX ENCLUSIVE OR

FUN123 P MXNR	MATRIX EXCLUSIVE NOR	FUN123 P MXNR

Symbol



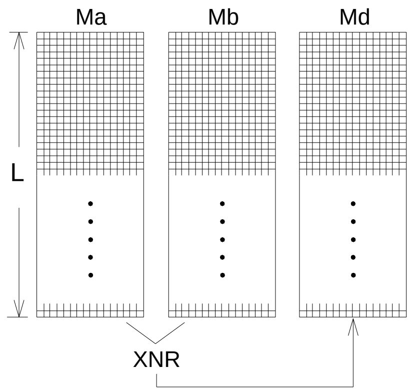
Ma : Starting register of source matrix a
Mb : Starting register of source matrix b
Md : Starting register of destination matrix
L : Length of matrix (Ma, Mb and Md)
Ma, Mb, Md may combine with V, Z,P0~P9 to serve indirect address application

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WM29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	2 | 256	V、Z P0~P9
Ma	○	○	○	○	○	○	○	○	○	○	○	○		○
Mb	○	○	○	○	○	○	○	○	○	○	○	○		○
Md		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	○	

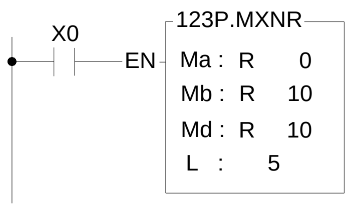
Description

- When operation control "EN" = 1 or has a transition from 0 to 1 (P instruction), will perform a logic XNR operation (if the 2 bits are the same, then the result will be 1, otherwise it will be 0) between 2 source matrixes with a length of L, Ma and Mb. The results will then be stored into the destination matrix Md, which also has the same length (the XNR operation is done by bits with the same bit numbers). For example, if Ma0 = 0, Mb0 = 1, then Md0 = 1; if Ma1 = 1, Mb1 = 1, then Md1 = 0; etc, right up until XNR reaches Ma16L-1 and Mb16L-1.

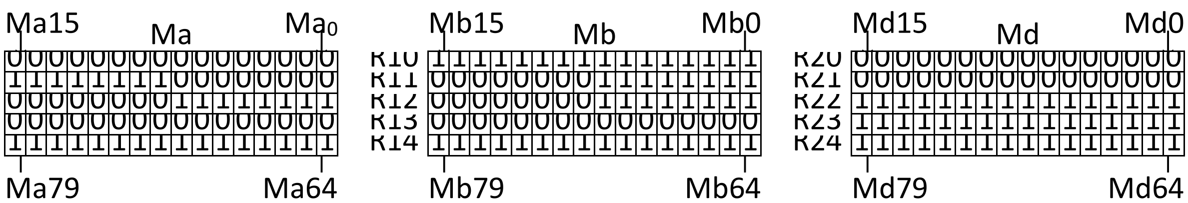
FUN123 P MXNR	MATRIX EXCLUSIVE NOR	FUN123 P MXNR



Example

Ladder diagram	ST
	IF X0 THEN MXNR(Ma:= R0, Mb:= R10, Md:= R10, L:= 5); END_IF

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

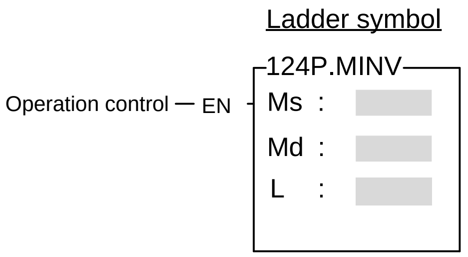


Before Execution

After Execution

7-15-5 MATRIX INVERSE

FUN124 P MINV	MATRIX INVERSE	FUN124 P MINV
Symbol		



Ms: Starting register of source matrix

Md : Starting register of destination

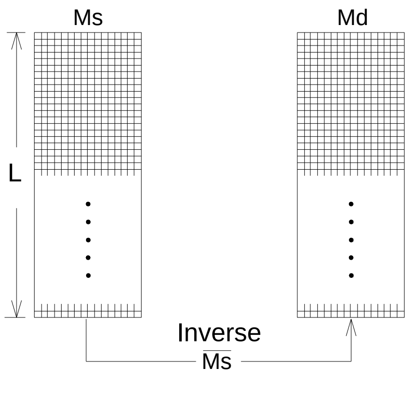
L : Length of matrix (Ms and Md)

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

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 │ WX1008	WY0 │ WY1008	WM0 │ WM29584	WS0 │ WS3088	T0 │ T1023	C0 │ C1279	R0 │ R34767	R34768 │ R34895	R35024 │ R35151	R35280 │ R43223	R43224 │ R47319	D0 │ D11999	2 │ 256	V 、Z P0~P9
Ms	○	○	○	○	○	○	○	○	○	○	○	○		○
Md		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	○	

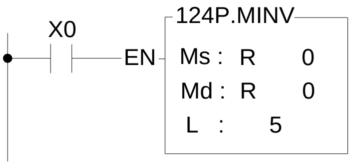
Description	

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

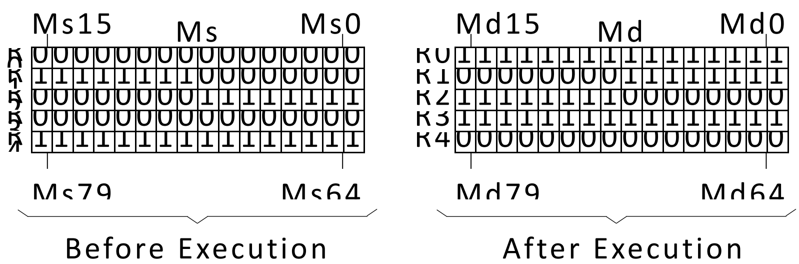


FUN124 P MINV	MATRIX INVERSE	FUN124 P MINV
Exaxmple		

Ladder diagram	ST

	`IF X0 THEN`
	`MINV( Ms:= R0, Md:= R0, L:= 5);`
	`END_IF`

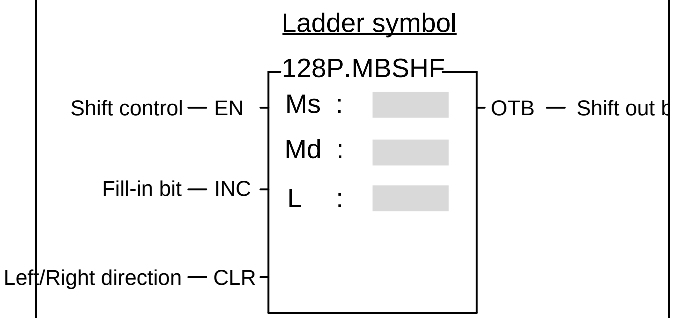
- In the program at left, when X0 goes from 0→1, the matrix comprised by R0 to R4 will be inverted, and then store back into itself (because in this example Ms and Md are the same matrix). The results obtained are shown at right in the diagram below.



7-15-6 MATRIX BIT SHIFT

FUN128 P MBSHF	MATRIX BIT SHIFT	FUN128 P MBSHF

Symbol



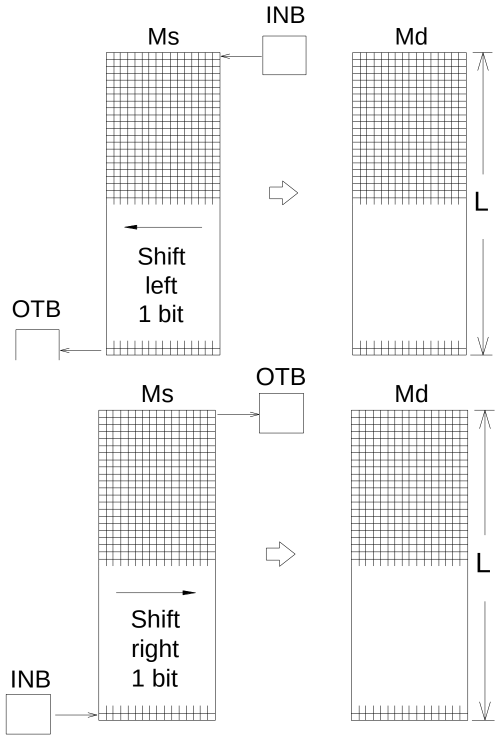
Ms : Starting register of source matrix
Md : Starting register of destination matrix
L : Length of matrix (Ms and Md)
Ms, Md may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WM29584	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34895	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	2 ∣ 256	V、Z P0~P9
Ms	○	○	○	○	○	○	○	○	○	○	○	○		○
Md		○	○	○	○	○	○		○	○*	○*	○		○
L											○*	○	○	

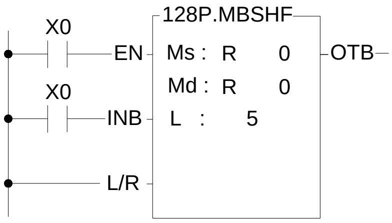
Description

- When shift control "EN" = 1 or has a transition from 0 to 1 (P instruction), source matrix Ms will be retrieved and completely shifted one position to the left (when L/R = 1) or one position to the right (when L/R = 0). The space caused by the shift (with a left shift it will be M0, and with a right shift it will be M16L-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 M16L-1, and with a right shift it will be M0) will appear at the output bit "OTB". Then the results of this shifted matrix will be filled into the destination matrix Md.

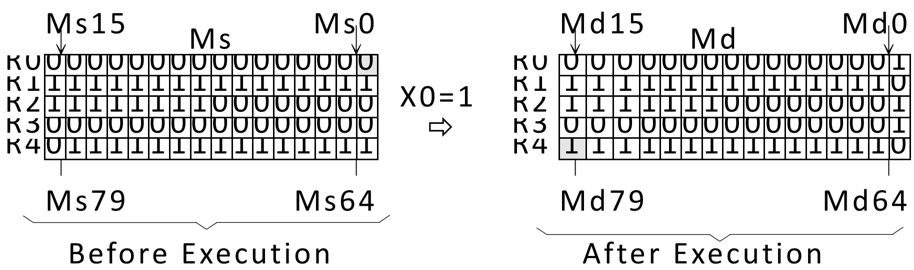
FUN128 P MBSHF	(MATRIX BIT SHIFT)	FUN128 P MBSHF



Example	

Ladder diagram	ST
	IF X0 THEN MBSHF(INB:= X0, L_R:= TRUE, Ms:= R0, Md:= R0, L:= 5, OTB=> M0); END_IF

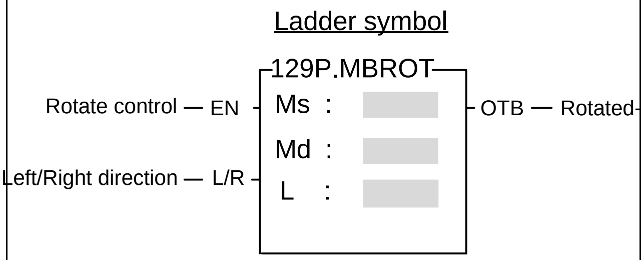
- The program at left is an example where Ms and Md are the same matrix. When X0 goes from 0→1, Ms will be completely retrieved and moved to the left (because L/R = 1) by 1 bit. It will then be stored back to Md, and the results are shown at right in the diagram below.



7-15-7 MATRIX BIT ROTATE

FUN129 P MBROT	MATRIX BIT ROTATE	FUN129 P MBROT

Symbol



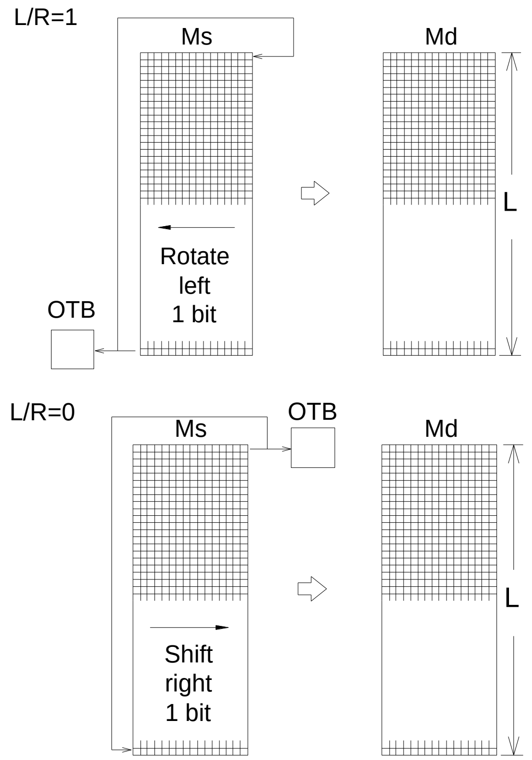
Ms : Starting register of source matrix
Md : Starting register of destination matrix
L : Length of matrix (Ms and Md)
Ms, Md may combine with V, Z, P0~P9 to serve indirect address application

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WM29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	2 | 256	V、Z P0~P9
Ms	○	○	○	○	○	○	○	○	○	○	○	○		○
Md		○	○	○	○	○	○		○	○*	○*	○		○
L											○*	○	○	

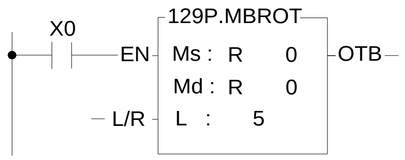
Description

- When rotate control "EN" = 1 or has a transition from 0 to 1 (P instruction), matrix Ms will be completely retrieved and rotated by one bit towards the left (when L/R = 1) or to the right (when L/R = 0). The space created by the rotation (with a left rotation it will be M0, and with a right rotation it will be M16L-1) will be replaced by the status of the rotated-out bit (with a left rotation it will be M16L-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".

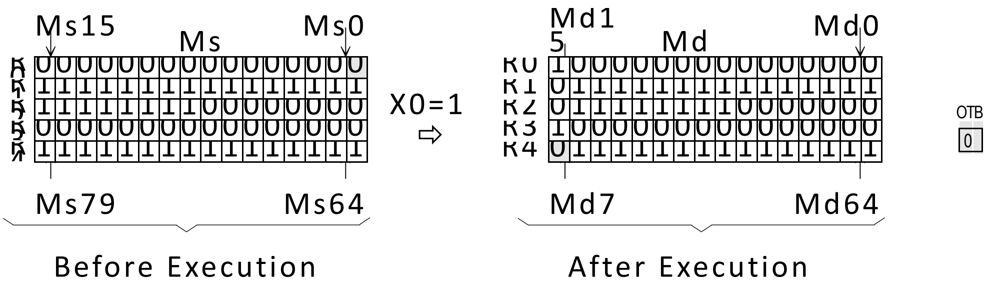
FUN129 P MBROT	MATRIX BIT ROTATE	FUN129 P MBROT



Example	

Ladder diagram	ST
	IF X0 THEN MBROT(L_R:= FALSE, Ms:= R0, Md:= R0, L:= 5, OTB=> M0); END_IF

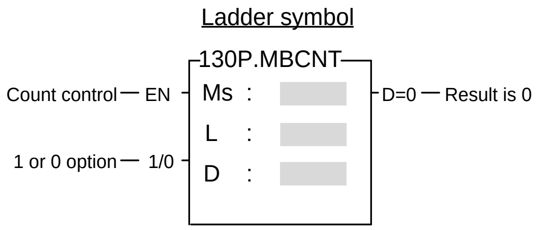
- In the program at left, Ms and Md are the same matrix. When X0 goes from 0→1, then the whole of Ms is retrieved and rotated right (because L/R = 0) by 1 bit. It is then stored back into Ms itself (because in this example Ms and Md are the same matrix). The results are shown at right in the diagram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



7-15-8 MATRIX BIT STATUS COUNT(MBCNT)

FUN130 P MBCNT	MATRIX BIT STATUS COUNT	FUN130 P MBCNT

Symbol



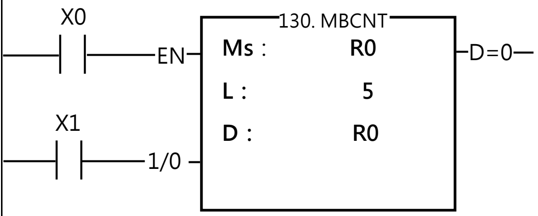
Ms : Starting register of matrix
L : Matrix length
D : Register storing count results
Ms may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	2 | 256	V,Z P0-P9
Ms	○	○	○	○	○	○	○	○	○	○	○	○		○
L							○				○*	○	○	
D		○	○	○	○	○	○		○	○	○*	○		

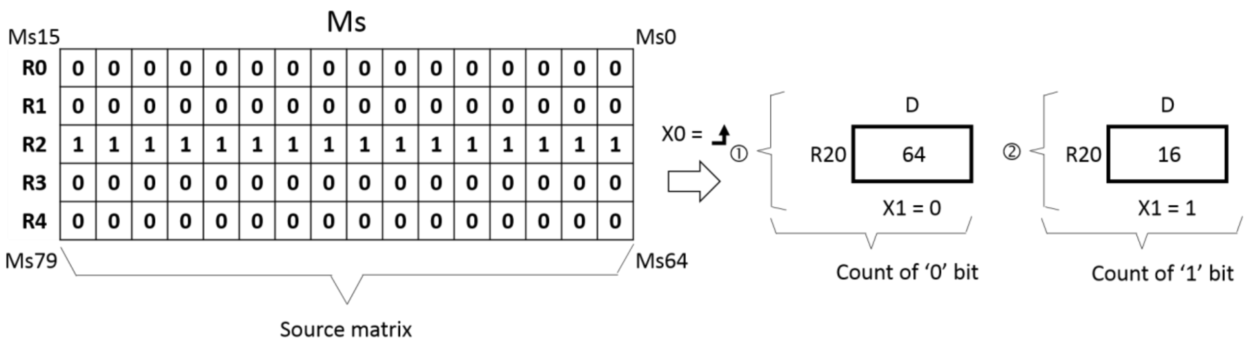
Description

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

FUN130 P MBCNT	MATRIX BIT STATUS COUNT	FUN130 P MBCNT
Example		

Ladder diagram	ST
	IF X0 THEN MBCNT(OnOff:= X1, Ms:= R0, L:= 5, D:= R0); END_IF

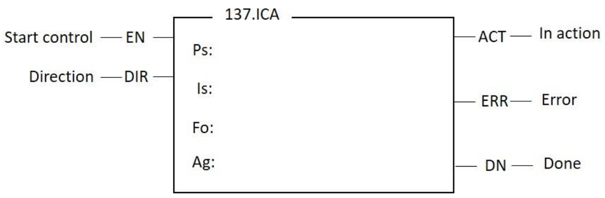
- This program sets X1 first as 0 (to count bits with status of 0) and then as 1 (to count bits with status of 1) and let the signal X0 has a transition from 0→1 for both case, the execution results are shown at right in the diagram below .



7-16 NC Positioning Instruction (FUN140~143)

7-16-1 ICA

FUN137 ICA	ICA	FUN137 ICA
Symbol		



Ps: Group of Pulse output (0~7)
0: Y0 & Y1
1: Y2 & Y3
2: Y4 & Y5
3: Y6 & Y7
4: Y8 & Y9
5: Y10 & Y11
6: Y12 & Y13
7: Y14 & Y15
Ls: External input X point index number (0~15)
Fo: Target Axis working speed
(1~100000 or 1~200000)
Ag: Fixed angle when interrupted (0~36000)

Operand \ Range	X	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	CPU's Xn	WX0 | WX1008	WY0 | WY1008	WM0 | WM19578	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	
Pw														0 ~ 7
Ls	○													
Fo								○				○	○	1 ~ 100000 or 1~ 200000
Ag								○				○	○	0 ~ 36000

FUN137 ICA	ICA	FUN137 ICA
Description		
1. The positioning axis can be controlled up to PSO7, but the actual maximum axis number that can be controlled varies with the host machine model. 2. The target working speed and the maximum frequency vary according to the host model, 100K and 200K. 3. In general-purpose and advanced sports hosts, external input points 8~15 of X will be reserved for the motion function and not supported by this command. 4. The external input point does not need additional special configuration in the interrupt setting in the I/O configuration. The relevant settings will be automatically made when the command is executed.		

7-16-2 ICF

FUN138 ICF	ICF	FUN138 ICF
Symbol		

Ps: Group of Pulse output (0~7)
0: Y0 & Y1
1: Y2 & Y3
2: Y4 & Y5
3: Y6 & Y7
4: Y8 & Y9
5: Y10 & Y11
6: Y12 & Y13
7: Y14 & Y15
Ls: External input X point index number (0~15)
Fo: Target Axis working speed
(1~100000 or 1~200000)
Fd: Output pulse movement amount after interrupt capture

Range / Operand	X	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	CPU's Xn	WX0 | WX1008	WY0 | WY1008	WM0 | WM19578	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	
Pw														0 ~ 7
Ls	○													
Fo								○				○	○	1 ~ 100000 or 1~ 200000
Fd								○				○	○	○

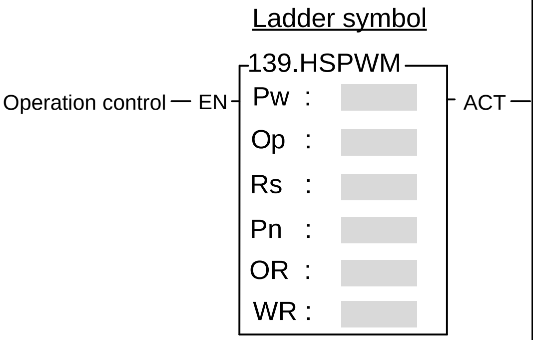
FUN138 ICF	ICF	FUN138 ICF
Description		

1. The positioning axis can be controlled up to PSO7, but the actual maximum axis number that can be controlled varies with the host machine model.
2. The target working speed and the maximum frequency vary according to the host model, 100K and 200K.
3. In general-purpose and advanced sports hosts, external input points 8~15 of X will be reserved for the motion function and not supported by this command.
4. The external input point does not need additional special configuration in the interrupt setting in the I/O configuration. The relevant settings will be automatically made when the command is executed.

7-16-3 HSPWM

FUN139 HSPWM	HIGH SPEED PULSE WIDTH MODULATION	FUN139 HSPWM

Symbol



Pw: High-speed pulse width modulation output point (0=Y0, 1=Y2, 2=Y4, 3=Y6, 4=Y8, 5=Y10, 6=Y12, 7=Y14)
Op: output polarity;
0=Output not inverting
1=output inverted
Rs: resolution;
0=1/100 (1%)
1=1/1000 (0.1%)
Pn: Output frequency parameter setting (0~255)
OR: PWM output width setting register 0~100 or 0~1000
WR: Instruction operation work register, other programs cannot be reused

Operand \ Range	Y	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	CPU's Yn	WX0 – WX1008	WY0 – WY1008	WM0 – WM29584	WS0 – WS3088	T0 – T1023	C0 – C1279	R0 – R34767	R34768 – R34895	R35024 – R35151	R35280 – R43223	R43224 – R47319	D0 – D11999	
Pw	○													0 ~ 3
Op	○													0 ~ 1
Rs	○													0 ~ 1
Pn		○	○	○	○	○	○	○	○	○	○	○	○	0 ~ 255
OR								○				○	○	0 ~ 1000
WR			○	○	○	○	○	○		○	○	○	○	

Description

FUN139 HSPWM	HIGH SPEED PULSE WIDTH MODULATION	FUN139 HSPWM

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

1. $f_{pwm} = \frac{184320}{(P_n + 1)}$ While Rs (Resolution)=1/100
2. $f_{pwm} = \frac{18432}{(P_n + 1)}$ While Rs (Resolution)=1/1000

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

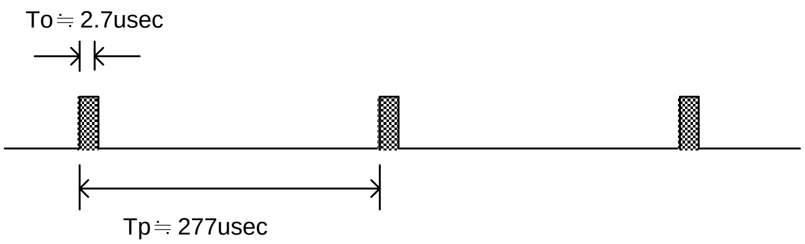
$f_{pwm} = \frac{184320}{(50+1)}$ = 3614.117…… ≒ 3.6KHz

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

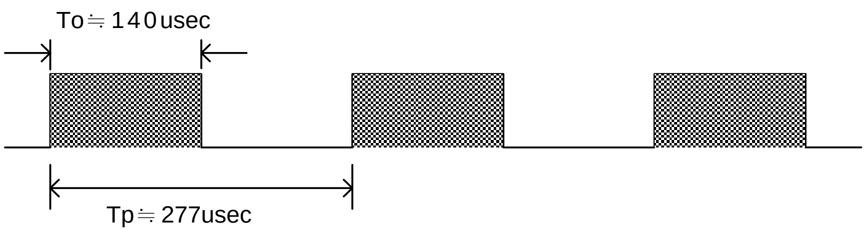
For Rs = 1/100, if OR(Setting of output pulse width) = 1, then T0 ≒ 2.7uS; if OR(Setting of output pulse width) = 50, then To ≒ 140uS.

.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 :



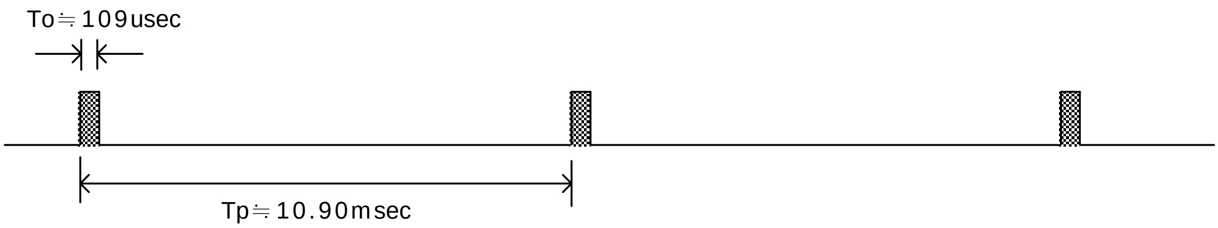
FUN139 HSPWM	HIGH SPEED PULSE WIDTH MODULATION	FUN139 HSPWM
Example 2	If Pn (Setting of output frequency) = 200, Rs = 1(1/1000), then	

$f_{pwm} = \dfrac{18432}{(200+1)}$ ≒ 91.7Hz ; T(Period) = $\dfrac{1}{f_{pwm}}$ ≒10.9mS

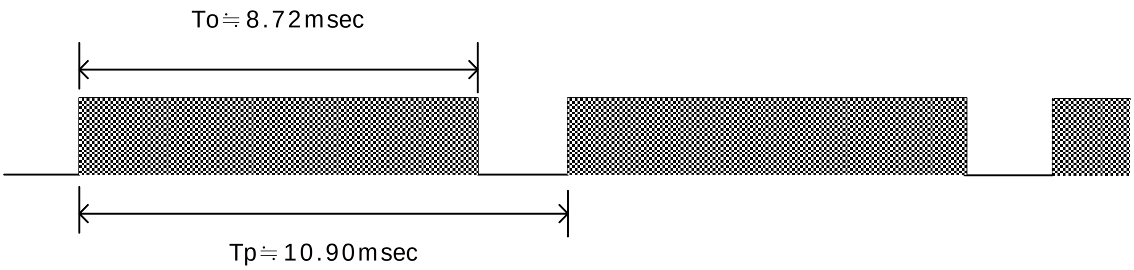
For Rs = 1/1000, if OR(Setting of output pulse width) = 10, then T0 ≒ 109uS; if OR(Setting of output pulse width) = 800, then To ≒ 8.72mS

.Output waveform :

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



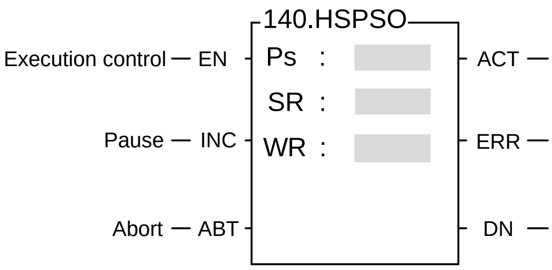
(2) Pn (Output frequency) = 200, Rs = 1 (1/1000), OR (Output pulse width) = 800 :



7-16-4 High Speed Pulse Output Instruction

FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO
Symbol		

Ladder symbol



Ps : The Pulse Output (0 ~ 3) selection

0：Y0 & Y1

1：Y2 & Y3

2：Y4 & Y5

3：Y6 & Y7

4：Y8 & Y9

5：Y10 & Y11

6：Y12 & Y13

7：Y14 & Y15

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 / Operand	HR	ROR	DR	K
	R0 | R34767	R43224 | R47319	D0 | D11999	2 | 256
Ps				0-3
SR	○	○	○	
WR	○	○*	○	

FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO
Description		

1. The positioning axis can be controlled up to PSO7, but the actual maximum axis number that can be controlled varies with the host machine model.
2. The NC positioning program of the FUN140 (HSPSO) command is edited in the form of a text program; each positioning point is called one step (including output frequency, action stroke, and transfer conditions), and one FUN140 can program up to 250 positioning points. Each positioning point needs to occupy 9 registers.
3. The biggest advantage of storing the positioning program in the temporary register is that if the man-machine is combined with the machine control setting, the positioning program can be stored in the man-machine. When changing the mold, the man-machine can directly access the Locator of the sub-mold.
4. When the execution control input "EN"=1, if Ps0~Ps7 are not occupied by other FUN140 instructions (Ps0=M9183, Ps1=M9184, Ps2=M9185, Ps3=M9186, Ps4=M9191, Ps5=M9192, Ps6= M9193, Ps7=M9194 state is ON, otherwise it is OFF), then start to execute from the next positioning point (if it has reached the last step, then start to execute from step 1 again); if Ps0~7 are occupied by other FUN140 instructions , the FUN140 to be occupied releases the control right, and this instruction obtains the pulse output right of positioning control.
5. When the execution control "EN"=0, stop the pulse output immediately.
6. When the pause output "PAU"=1, and the execution control "EN" was previously 1, the pulse output is paused. When the pause output "PAU"=0 and the execution control "EN" is still 1, it will continue to output the unfinished pulse number.
7. When the output "ABT" = 1, stop the pulse output immediately. (The next time when the execution control input "EN"=1, it will be executed again from the first step positioning point)
8. When pulse output is in progress, the output indicator "ACT" is ON.
9. When the command is executed incorrectly, the output indication "ERR" is ON. (The error code is stored in the error code register)
10. When each step of positioning is completed, the output indication "DN" is ON.

FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO

*** Be sure to set the working mode of the Pulse output (if not set, Y0~Y15 is regarded as a general output) to one of the three modes of U/D, P/R or A/B, the Pulse output can output normally.

U/D Mode: Y0 (Y2, Y4, Y6, Y8, Y10, Y12, Y14), as up pulse.
Y1 (Y3, Y5, Y7, Y9, Y11, Y13, Y15), as down pulse.
P/R Mode: Y0 (Y2, Y4, Y6, Y8, Y10, Y12, Y14), as the pulse out.
Y1 (Y3, Y5, Y7, Y9, Y11, Y13, Y15), as the direction.
A/B Mode: Y0 (Y2, Y4, Y6, Y8, Y10, Y12, Y14), as A phase pulse.
Y1 (Y3, Y5, Y7, Y9, Y11, Y13, Y15), as B phase pulse.

The output polarity for Pulse Output can select to be Normally ON or Normally OFF.

※FUN140 does not support pulse mode (U), if you need to use it, please use it with FUN139

[Interface Processing Signal]

M9183	ON: Ps0 ready
	OFF: Ps0 in action
M9184	ON: Ps1 ready
	OFF: Ps1 in action
M9185	ON: Ps2 ready
	OFF: Ps2 in action
M9186	ON: Ps3 ready
	OFF: Ps3 in action
M9187	ON: Ps0 complete the last step
M9188	ON: Ps1 complete the last step
M9189	ON: Ps2 complete the last step
M9190	ON: Ps3 complete the last step

FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO

M9191	ON: Ps4 ready
	OFF: Ps4 in action
M9192	ON: Ps5 ready
	OFF: Ps5 in action
M9193	ON: Ps6 ready
	OFF: Ps6 in action
M9194	ON: Ps7 ready
	OFF: Ps7 in action
M9195	ON: Ps4 complete the last step
M9196	ON: Ps5 complete the last step
M9197	ON: Ps6 complete the last step
M9198	ON: Ps7 complete the last step

Ps No.	Current Output Frequency	Current PS Position	Remaining number of PS to be output	Error Code
Ps0	DR35328	DR35336	DR35344	R35320
Ps1	DR35330	DR35338	DR35346	R35321
Ps2	DR35332	DR35340	DR35348	R35322
Ps3	DR35334	DR35342	DR35350	R35323
Ps4	DR35655	DR35663	DR35671	R35647
Ps5	DR35657	DR35665	DR35673	R35648
Ps6	DR35659	DR35667	DR35675	R35649
Ps7	DR35661	DR35669	DR35677	R35650

FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO

R35324：Ps0 the step number at the end of each step
R35325：Ps1 the step number at the end of each step
R35326：Ps2 the step number at the end of each step
R35327：Ps3 the step number at the end of each step
R35651：Ps4 the step number at the end of each step
R35652：Ps5 the step number at the end of each step
R35653：Ps6 the step number at the end of each step
R35654：the step number at the end of each step

- Positioning Progrm Format:

SR：The initial register of the positioning program, the description is as follows:

SR	A55AH	;Valid positioning program, the initial register flag must be A55AH
SR+1	Total Steps	;1~250
SR+2		
SR+3		
SR+4		
SR+5		
SR+6		
SR+7		The first step of point positioning program (each step occupies 9 registers)
SR+8		
SR+9		
SR+10		
	. . .	
		Step N of point positioning program
SR+N×9+2		

FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO

Instruction Operation Working Register Description:

WR as Stating Register

WR+0	Steps currently working or reserved
WR+1	Work flag
WR+2	System use
WR+3	System use
WR+4	System use
WR+5	System use
WR+6	System use

WR+0: If the command is being executed, the content value of the temporary register is the number of steps being executed (1~N).

If the instruction is not being executed, the content value of the register represents the number of steps currently reserved.

WR+1: B0 ~ B7, Total steps

B8 = Reserved

B9 = Reserved

B10= Reserved

B11= Reserved

B12=ON, Pulse output (output indication "ACT").

B13=ON, Command execution error (output indication "ERR").

B14=ON, One-step positioning is done (output indicates "DN").

***After each positioning point is completed, the output indication "DN" will remain ON; if you do not want the output indication to remain ON, then after each positioning point is completed, use the upper edge contact command controlled by the output indication coil to set WR + 1 clear the content of the register to 0, and it can be achieved.

FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO

Error Code			
0	: No errors	53: Homing clears CLR output point errors	
1	: Parameter 0 Error	54: I/O configuration error (Ex: used in unsupported mode, Such as FUN140 with pulse wave mode; It should be changed to FUN139 with pulse wave mode)	
2	: Parameter 1 Error	60: Illegal tween-driven command	
3	: Parameter 2 Error		
4	: Parameter 3 Error		
5	: Parameter 4 Error		
6	: Parameter 5 Error		
7	: Parameter 6 Error		Possible error codes when executing FUN141
8	: Parameter 7 Error		
9	: Parameter 8 Error		
10	: Parameter 9 Error		
13	: Parameter 12 Error		
14	: Parameter 13 Error		
15	: Parameter 14 Error		
30	: Speed setting variable number error		
31	: Speed setpoint error		
32	: Stroke setting variable number error		
33	: Stroke setting value error		
34	: Illegal positioning program		
35	: Step length error		
36	: Exceeded the maximum number of steps		
37	: Max. frequency error		
38	: Start/stop frequency error		
39	: Movement correction value is too large		Possible error codes when executing FUN140 and FUN147
40	: Movement out of range		
41	: ABS addressing is not allowed within DRVC		
42		: DRVC cannot connect to DRVZ command	
43		: Driver command code error	
50	: DRVZ working mode error		
51	: Near point DOG input point error		
52		: Zero signal PG0 input point error	

Note: The content of the error indication register will keep the latest error code. If you need to confirm that there is no more error, you can clear the error indication register to 0. As long as the content remains 0, it means there is no error occur.

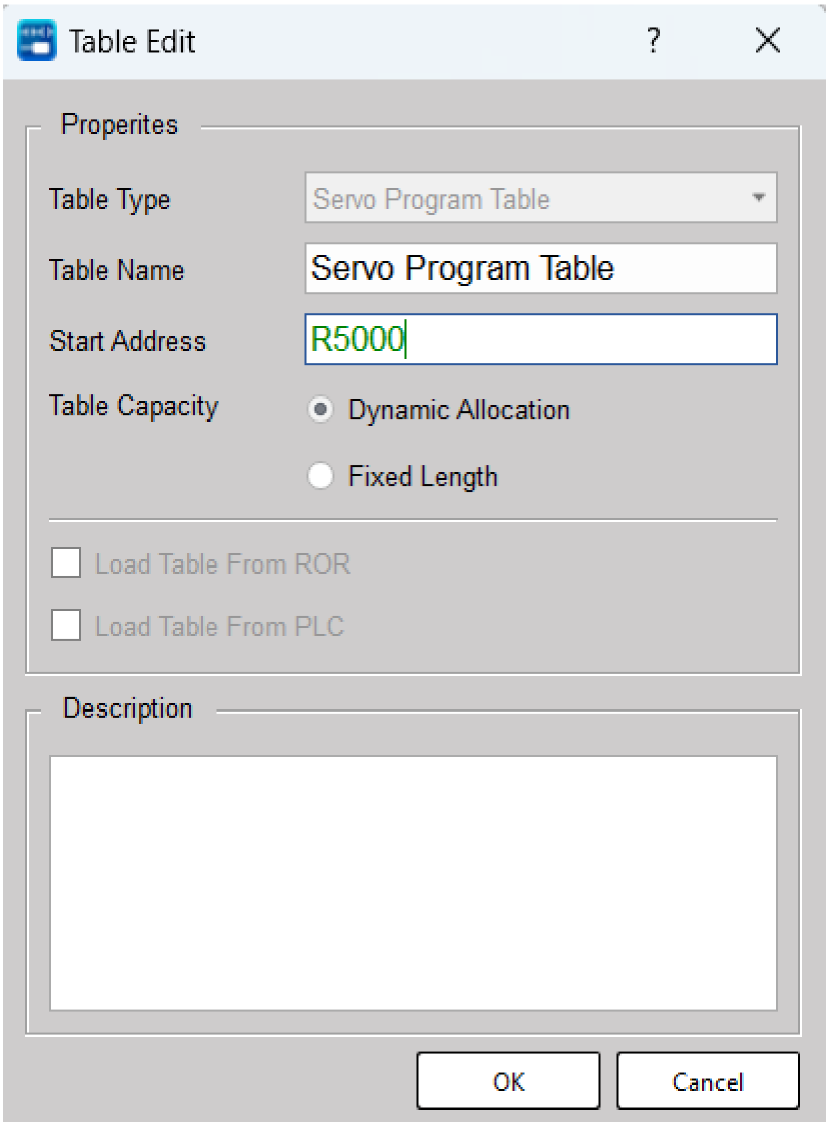
FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSP

Edit Servo Command Table Using UperLogic

Click on the Servo Command Form in the project window: Project Name

Edir Table

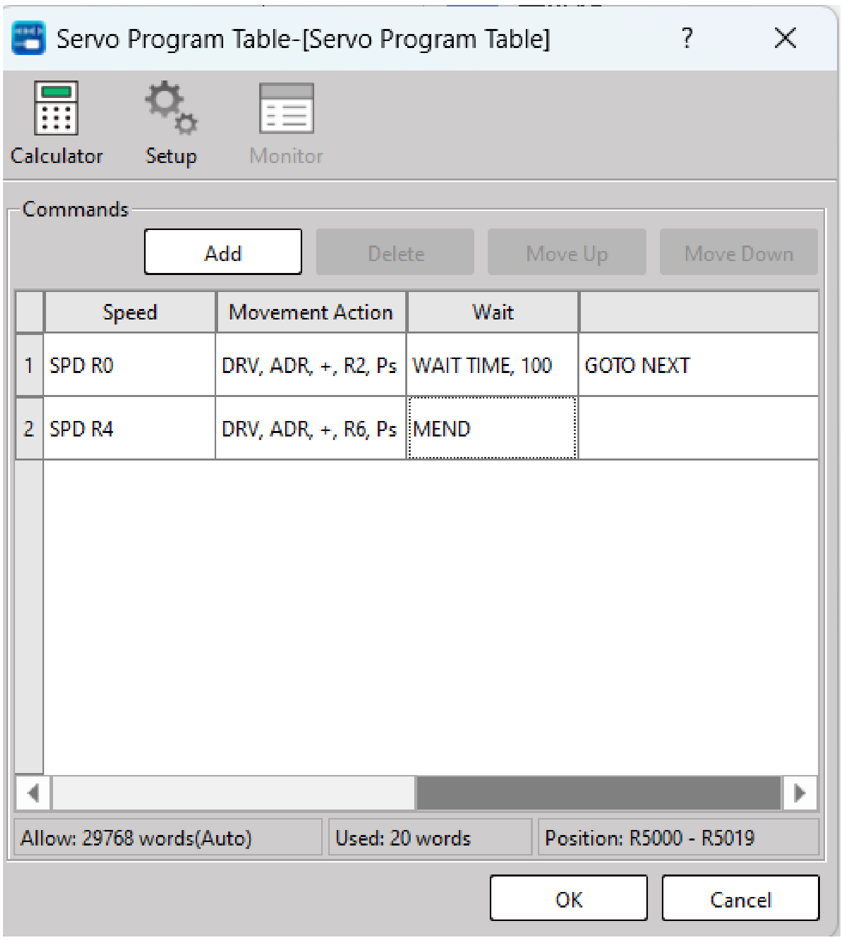
Servo Command Table → After right-clicking, click "Add Servo Command Form"



FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO

- Table type: Fixed as "servo command form".
- Table name: You can enter an easily identifiable name for the servo command form, which is convenient for future modification or debugging. 。
- Table start position: The start position of the data table start register SR used by the servo command instruction (FUN140).

※ For the establishment of the servo command form, please refer to Chapter 7 (Form Input and Editing) of the UperLogic Interface Manual, or click the command and press Z (shortcut key) to create it.



FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO

- In order to make the positioning program easy to edit, read, and maintain, we have derived the following related commands under the FUN140 command. Users can directly edit and modify the positioning program under Uperlogic.
- The list of positioning derivative commands is as follows:

Command	Operand	Description
SPD	XXXXXX or Rxxxxx or Dxxxxx	● Frequency or speed of pulse wave output (FUN141 parameter 0=0 is speed; parameter 0=1 or 2 is frequency, the system defaults to frequency); operands can directly input constants or variables (Rxxxx, Dxxxx); When the element is a variable, a total of two temporary registers are required, such as D10, which means that D10 (Low Word) and D11 (High Word) are frequency or speed setting values. ● When the speed setting is selected, the system will automatically convert the speed setting value into frequency output. ● Frequency output range:1≦frequency output≦100000 or 200000Hz *** When the frequency setting value = 0, this instruction waits until the setting value is not equal to 0 before executing the positioning pulse output.

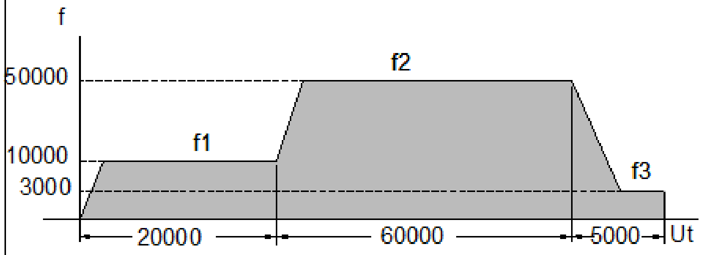
FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO

<table>
<tr>
<td>DRV</td>
<td>ADR，+，XXXXXXXX，Ut
ADR，+，XXXXXXXX，Ps
ADR，–，XXXXXXXX，Ut
ADR，–，XXXXXXXX，Ps
ADR， ，XXXXXXXX，Ut
ADR， ，–XXXXXXXX，Ut
ADR， ，XXXXXXXX，Ps
ADR， ，–XXXXXXXX，Ps
ADR，+，Rxxxx，Ut
ADR，+，Rxxxx，Ps
ADR，–，Rxxxx，Ut
ADR，–，Rxxxx，Ps
ADR， ，Rxxxx，Ut
ADR， ，Rxxxx，Ps
ADR，+，Dxxxx，Ut
ADR，+，Dxxxx，Ps
ADR，–，Dxxxx，Ut
ADR，–，Dxxxx，Ps
ADR， ，Dxxxx，Ut
ADR， ，Dxxxx，Ps
ABS， ，XXXXXXXX，Ut
ABS， ，–XXXXXXXX，Ut
ABS， ，XXXXXXXX，Ps
ABS， ，–XXXXXXXX，Ps
ABS， ，Rxxxx，Ut
ABS， ，Rxxxx，Ps
ABS， ，Dxxxx，Ut
ABS， ，Dxxxx，Ps</td>
<td>Pulse output
(When FUN141 parameter 0=1, the unit is Ps; when parameter 0=0 or 2, the unit is mm, Deg, Inch; the system defaults to Ps)
When the pulse wave output unit is not Ps, the system will convert it to Ps number output according to the settings of parameters 1, 2, and 3 of FUN141.
There are four operands in the DRV instruction, which are described as follows:
The first operand: positioning coordinate selection
ADR or ABS: ADR, relative value coordinate positioning.
ABS, absolute value coordinate positioning.
The second operand: selection of running direction (relative value coordinates are valid)
' + ' or ' - ': ' + ', run forward or count up.
' - ', reverse or count down.
Or ' ': ' ', the running direction is determined by the stroke setting value
(Positive value: Forward rotation; Negative value: Reverse rotation)
The third operand: stroke setting value (pulse wave output).
XXXXXXXX or –XXXXXXXX or Rxxxx or Dxxxx: Constants or variables (Rxxxx, Dxxxx) can be directly input; when using variables, two temporary registers are required, such as R0, which means that R0 (Low Word) and R1 (High Word) are stroke setting values.
*** When the stroke setting value = 0, no matter the coordinate is ADR or ABS, it means continuous operation.
Stroke setting range:–99999999 ≤Stroke setting value ≤ 99999999
The fourth operand: stroke setting value resolution
Ut or Ps: when Ut, the resolution is one unit; (determined by FUN141 parameter 0, 3) when Ps, the forced resolution is one Ps</td>
</tr>
</table>

FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO

Command	Operand	Description
DRVC	ADR, or ABS, +, or -, xxxxx, or Rxxxx, or Dxxxx, Ut or Ps	The use and operation element description of DRVC is the same as that of DRV instruction. ***DRVC is used for continuous multi-stage speed change control (up to 8 stages) *** For the continuous multi-stage speed change control formed by DRVC, only the first DRVC instruction can use absolute value coordinate positioning. ***The running direction of DRVC can only be determined by ' + ' or ' -' ***The direction of continuous multi-stage speed control (forward and reverse) can only be determined by the direction of the first stage, and the direction operator of the subsequent command is invalid; that is, the multi-stage speed change control can only be in the same direction. Example: Continuous three-stage speed control 001 ；Pulse frequency=10KHz SPD 10000 ；Forward rotation 20000 DRVC ADR，+，20000，Ut units GOTO NEXT 002 ；Pulse frequency =50 KHz SPD 50000 ；Forward rotation 60000 DRVC ADR，+，60000，Ut units GOTO NEXT 003 ；Pulse frequency =3 KHz SPD 3000 ；Forward rotation 50000 DRV ADR，+，5000，Ut units WAIT X0 ；Wait for X0 ON, and GOTO 1 execute the first step again *** Note: The number of DRVC instructions must be one less than the number of consecutive segments, that is, the last segment must use the DRV instruction.

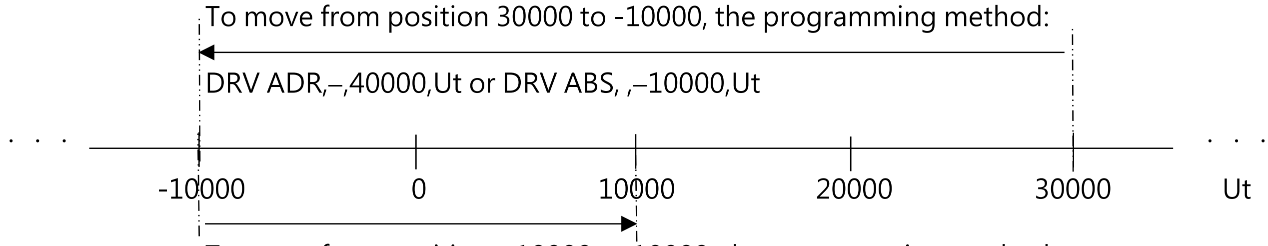
FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO

Command	Operand	Description
		The above example is three consecutive speed control, DRVC instruction uses two, and the third section must use DRV instruction. The above example shows: 
DRVZ	MD1	DRVZ is a convenient return-to-origin command that supports MD1 return-to-origin. ***For details about MD1 of DRVZ, please refer to Section 9.6 (Mechanical Return to Origin).

Note: Comparison between relative coordinate positioning (ADR) and absolute value coordinate positioning (ABS)

To move from position 30000 to -10000, the programming method:

DRV ADR,–,40000,Ut or DRV ABS, ,–10000,Ut



To move from position 10000 to 10000, the programming method is: DRV ADR,+,20000,Ut or DRV ABS, ,10000,Ut DRV ABS, ,10000,Ut

FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO

Command	Operand	Description
WAIT	TIME, XXXXX or Rxxxxx or Dxxxxx or X0～X1023 or Y0～Y1023 or M0～M29599 or S0～S3103	● When the pulse output is completed, it is necessary to execute the next waiting command; There are five types of operands, which are described as follows: Time: Waiting time (unit is 0.01 second), you can directly input constant or variable (Rxxxx or Dxxxx); when the timer is up, execute the GOTO instruction the number of steps. X0~X1023: Wait for the input contact signal to be ON, and execute the number of steps indicated by GOTO. Y0~Y1023: Wait for the output contact signal to be ON, and execute the number of steps indicated by GOTO. M0~M29599: Wait for the internal relay to be ON, and execute the number of steps indicated by GOTO. S0~S3103: Wait for the step relay to be ON, and execute the number of steps indicated by GOTO.
ACT	TIME, XXXXX or Rxxxxx or Dxxxxx	● After the pulse wave outputs the action time described by ACT, immediately execute the steps indicated by GOTO; that is, after the pulse wave output for a period of time, immediately execute the next step. The action time (unit: 0.01 second) can be directly input as a constant or variable (Rxxxxx or Dxxxxx); when the action time is up, the number of steps indicated by GOTO will be executed.

FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO

Command	Operand	Description
EXT	X0～X1023 or Y0～Y1023 or M0～M29599 or S0～S999	○External trigger command, when the pulse wave output is in progress (the number of pulse waves has not been sent), if the external trigger signal is activated (ON), the number of steps indicated by GOTO will be executed immediately; if the pulse wave output has been completed, the external trigger signal has not yet Action is the same as the WAIT instruction, the number of steps indicated by GOTO will be executed only when the signal (ON).
GOTO	NEXT or 1～N or Rxxxxx or Dxxxxx	When the conditions of WAIT, ACT, EXT and other instructions are met, use the GOTO instruction to describe the number of steps to be executed. NEXT: Represents the next step 1~N: Execute the first few steps Rxxxxx: The number of steps to be executed is stored in the temporary register Rxxxxx Dxxxxx: The number of steps to be executed is stored in the temporary register Dxxxxx
MEND		Positioning program ends

FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO

- Writing of positioning program:

Before editing the positioning program, you must first complete the FUN140 command, and specify the initial register number to store the positioning program in the FUN140 command; when editing the positioning program, the newly edited positioning program will be stored in the specified register In a block, each locating point (called 1 step) will occupy 9 registers. If there are N locating points (N steps), a total of N 9 + 2 registers will be occupied.

*** Note: The register for storing the positioning program cannot be reused!

Program Format and Examples:

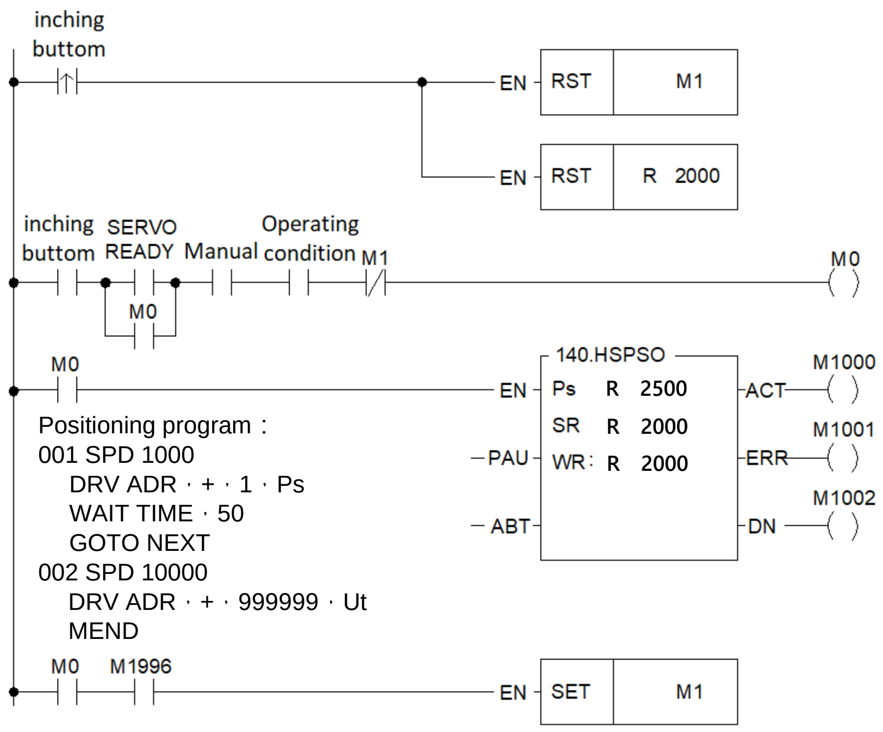
```
001 SPD              5000    ; Pulse frequency=5KHz
    DRV  ADR,+,10000,Ut      ; Forward rotation 10000 units
    WAIT TIME,100     ; Wait 1 second
    GOTO             NEXT    ; Execute nest step
002 SPD              R1000   ; The pulse frequency is stored in DR1000 (R1001 and R1000)
    DRV  ADR,+,D100,Ut  ; Transfer strokes are stored in DD100 (D101 and D100)
    WAIT TIME,R500    ; Waiting time is stored in R500
    GOTO             NEXT    ; Execute nest step
003 SPD              R1002   ; The pulse frequency is stored in DR1002 (R1003 and R1002)
    DRV  ADR,–,D102,Ut  ; The reverse stroke is stored in DD102 (D103 and D102)
    EXT  X0           ; When the external trigger X0 (deceleration point) is ON, execute the next step
immediately
    GOTO             NEXT
004 SPD              2000    ; Pulse frequency=2K HZ
    DRV  ADR,–,R4072,Ps      ; Continue to execute the unfinished PS number of step 3 (stored in
DR4072)
    WAIT X1           ; While waiting for X1 ON
    GOTO             1    ; Execute the first step
```

FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO

Program Example: Jog Forward

When the button is pressed for less than 0.5 seconds (variable), only one (variable) pulse is output; When the inching button is pressed for more than 0.5 seconds (variable), the pulse wave will be output continuously (frequency is 10KHz, variable), and the output will not stop until the inching button is released; or it can be designed to only output N pulses at most.

Ladder diagram



- Clear end signal
- Executed from the first step each time
- After the last step is executed, set end signal

ST

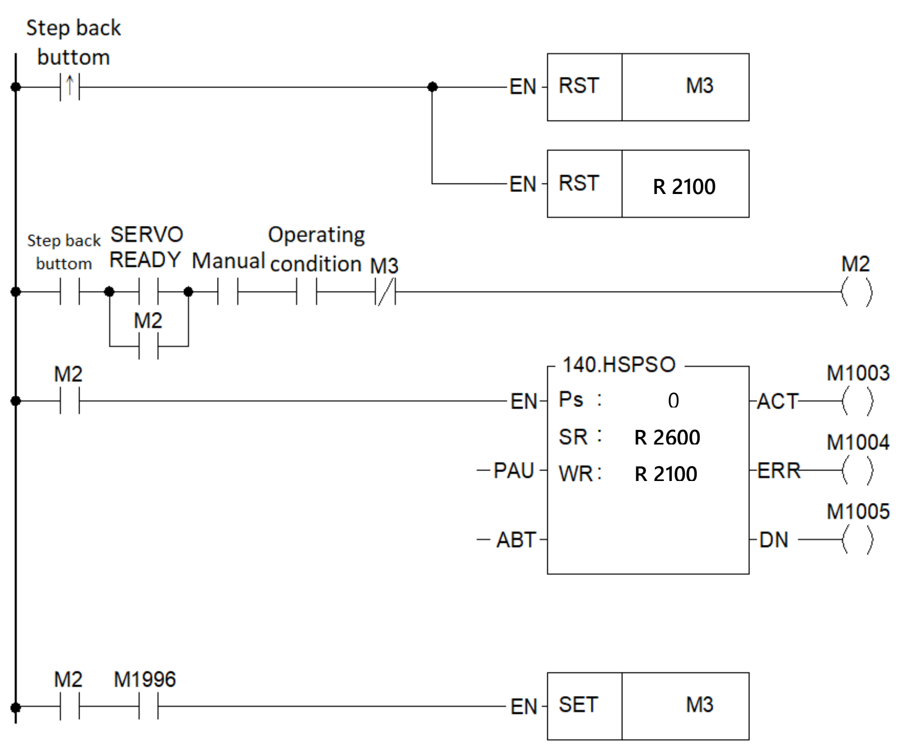
```
IF R_TRIG( S:= inching buttom ) THEN
M1 := FALSE;
R2000 := 0;
END_IF
IF  inching buttom  AND (SERVO READY OR M0) AND manual AND operating
condition AND M1 = FALSE THEN
M0 := TRUE;
ELSE
M0 := FALSE;
END_IF
HSPSO( EN:= M0, PAU:= FALSE, ABT:= FALSE, Ps:= 0, SR:= R2500, WR:= R2000,
ACT=> M1000, ERR=> M1001, DN=> M1002);
IF M0 AND M1996 Then
M1 := FALSE;
END_IF
```

FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION	FUN140 HSPSO

Program Example: Jog Backward

When the step back button is pressed for less than 0.5 seconds (variable), only one (variable) pulse is output.When the button is pressed for more than 0.5 seconds (variable), the pulse wave will be output continuously (frequency is 10KHz, variable), and the output will not stop until the button is released; or it can be designed to only output N pulses at most wave number.

Ladder diagram



- Clear end signal
- Executed from the first step each time

- After the last step is executed, set end signal

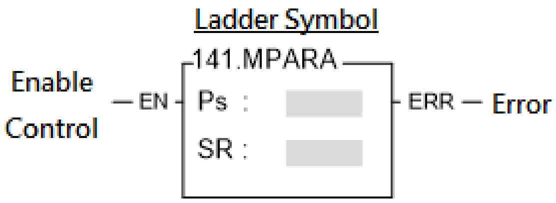
ST

```
IF R_TRIG( S:=  step back buttom ) THEN
M3  := FALSE;
R2100  := 0;
END_IF
IF step back buttom AND (SERVO READY OR M2) AND manual ANDoperating
condition AND M3 = FALSE THEN
M2  := TRUE;
ELSE
M2  := FALSE;
END_IF
HSPSO( EN:= M2, PAU:= FALSE, ABT:= FALSE, Ps:= 0, SR:= R2500, WR:= R2000,
ACT=> M1003, ERR=> M1004, DN=> M1005);
IF M2 AND M1996 Then
M3  := FALSE;
END_IF
```

7-16-5 POSITIONING PROGRAM PARAMETER SETTING COMMAND (MPARA)

FUN141 MPARA	MPARA	FUN141 MPARA

Symbol



Ps: Group of Pulse output (0~7)

SR: Parameter table starting register, 18 parameters in total, occupying 24 registers

Operand \ Range	HR	DR	ROR	K
	R0 | R34767	D0 | D11999	R43224 | R47319	2 | 256
Ps				0 ~ 7
SR	○	○	○	

Description

- The positioning axis can be controlled up to PSO7, but the actual maximum axis number that can be controlled varies with the host machine model.
- It is not necessary to use this instruction (But in the first-time setting is necessary). 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.)

FUN141 MPARA	MPARA	FUN141 MPARA

R2000	0 ~ 2		Parameter 0	System default =1
R2001	1 ~ 65535 Ps/Rev		Parameter 1	System default =2000
DR2002	1 ~ 999999 μM/Rev 1 ~ 999999 mDeg/Rev 1 ~ 99999930.1 mInch/Rev		Parameter 2	System default =2000
R2004	0 ~ 3		Parameter 3	System default =2
DR2005	1 ~ 921600 Ps/sec 1 ~ 153000		Parameter 4	System default =460000
DR2007	0~921600 Ps/sec 1~153000		Parameter 5	System default =141
R2009	1~65535 Ps/sec		Parameter 6	System default =1000
R2010	0~32767		Parameter 7	System default =0
R2011	0~30000		Parameter 8	System default =5000
R2012	0~1	0~1	Parameter 9	System default =0100H
R2013	-32768~32767		Parameter 10	System default =0
R2014	-32768~32767		Parameter 11	System default =0
R2015	0~30000		Parameter 12	System default =0
R2016	0~30000		Parameter 13	System default =500
DR2017	0~1999999		Parameter 14	System default =0
DR2019	00H~FFH 00H~FFH	00H~FFH 00H~FFH	Parameter 15	System default =FFFFFFFFH
DR2021	-999999~999999		Parameter 16	System default =0
R2023	0~255		Parameter 17	System default =1

FUN141 MPARA	MPARA	FUN141 MPARA

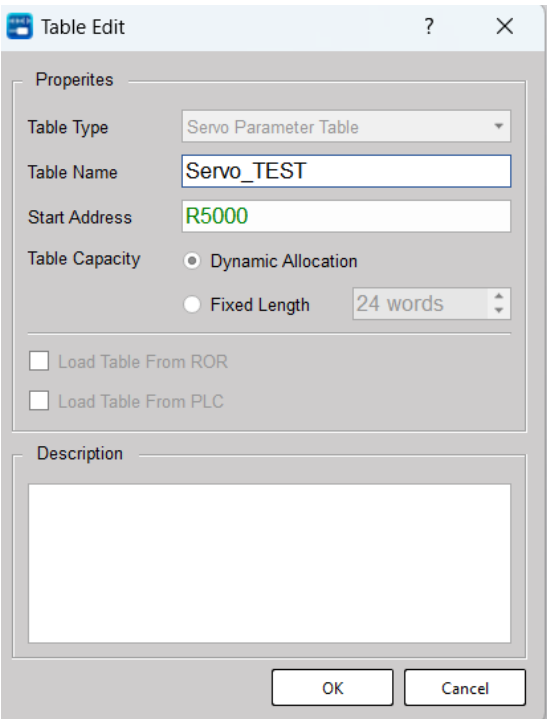
Use UperLogic to edit the servo parameter table

In the project window, click the servo parameter table:

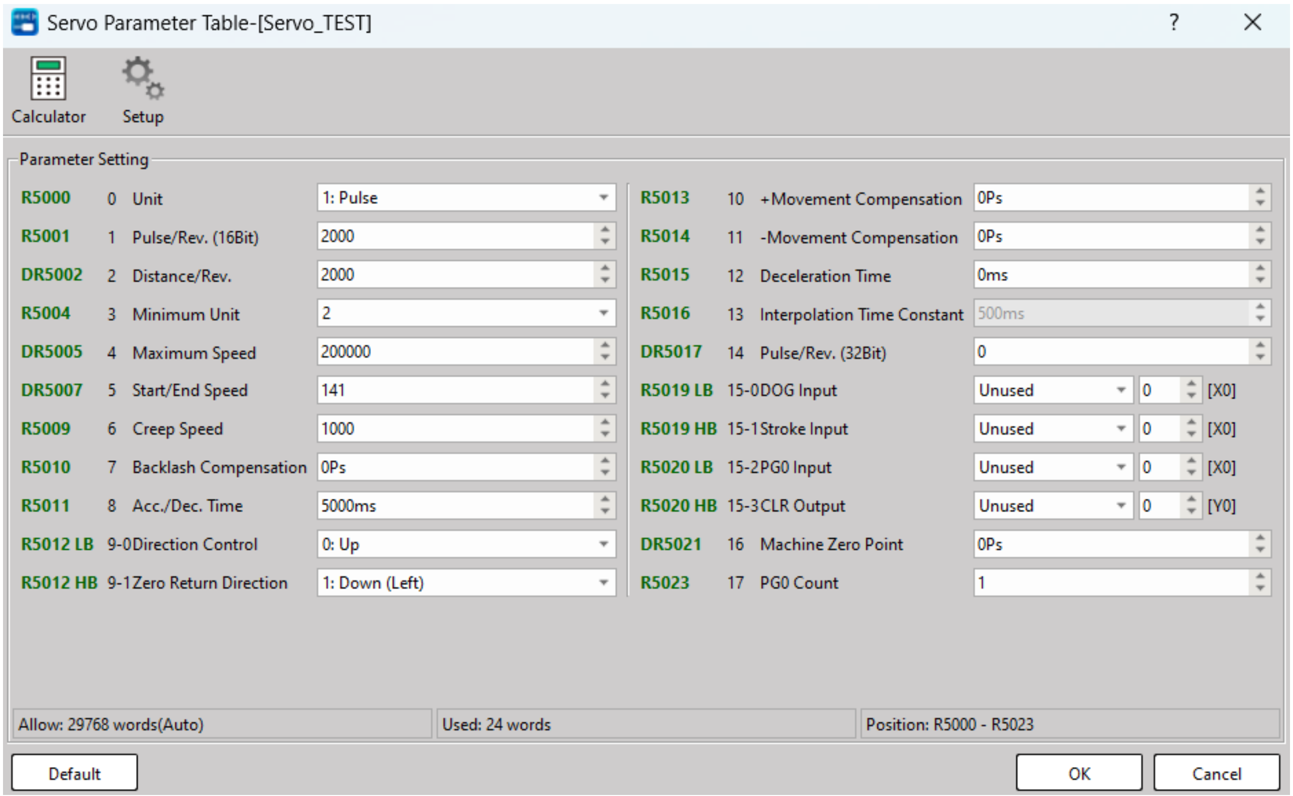
Project Name

Table Edit

Servo Parameter Table →After right-clicking, click "Add Servo Parameter Table".



- Table type: fixed as "servo parameter form".
- able name: You can enter an easily identifiable name for the servo parameter table, which is convenient for future modification or debugging.
- Table start position: The start position of the data table start register SR used by the servo parameter command (FUN141).



Parameter Description

- Parameter 0: unit setting, the default value is 1
  - ○When the setting value is 0, the travel and speed setting values used in the program are specified in mm, Deg, Inch as the unit, which is called the mechanical unit.
  - ○When the setting value is 1, the travel and speed setting values used in the program are all specified in the unit of Pulse, which is called the motor unit.
  - ○When the setting value is 2, the stroke setting values used in the program are all specified in mm, Deg, Inch as the unit, and the speed setting is all specified in Pulse as the unit, which is called compound unit.

Parameter 0, Unit	“0” Mechanical unit	“1” Motor unit	“2” Compound unit
Parameter 1, 2	no need to set	No need to set	Must be set
Parameter 3, 7, 10, 11	Mm, Deg, Inch	Ps	Mm, Deg, Inch
Parameter 4, 5, 6, 15, 16	Cm/Min, Deg/Min, Inch/Min	Ps/Sec	Ps/Sec

FUN141 MPARA	MPARA	FUN141 MPARA

- Parameter 1: pulse number/1 revolution, the default value is 2000, that is, 2000 Ps/Rev
  The number of pulses required for one revolution of the motor (A)
  A=1 ~ 65535 (when it is above 32767, set it as a decimal positive number) Ps/Rev
  When parameter 14 = 0, take parameter 1 as pulse number/1 revolution.
  When parameter 14 ≠ 0, take parameter 14 as pulse number/1 revolution.
- Parameter 2: movement amount/1 revolution, the default value is 2000, that is, 2000 Ps/Rev
- The distance driven by one revolution of the motor (B)

B=1 ~ 999999 μM/Rev

1 ~ 999999 mDeg/Rev

1 ~ 999999x0.1 mInch/Rev

- Parameter 3: The minimum setting unit, the default value is 2, equivalent to two decimal places

Parameter 0 / Parameter3	Set Value =0, Mechanical unit; Set Value =2, Compound unit；			Set Value 1 Motor unit Ps
	mm	Deg	Inch	
Set Value=0	x1	x1	x0.1	x1000
Set Value =1	x0.1	x0.1	x0.01	x100
Set Value =2	x0.01	x0.01	x0.001	x10
Set Value =3	x0.001	x0.001	x0.0001	x1

FUN141 MPARA	MPARA	FUN141 MPARA

- Parameter 4: Maximum speed setting, the default value is 460000, that is, 460000 Ps/Sec
  - Motor and compound unit: 1 ~ 921600 Ps/Sec
  - Mechanical unit: 1 ~ 153000 (cm/Min, x10 Deg/Min, Inch/Min)

  But the highest frequency can not be greater than 921600 Ps/Sec

  $f\_max = (V\_max \times 1000 \times A) / (6 \times B) \leq 921600$ Ps/Sec

  $f\_min \geq 1$ Ps/Sec

Note: A=parameter 1, B=parameter 2

- Parameter 5: start/end speed, default value=141
  - Motor and compound unit: 1~921600 Ps/Sec
  - Mechanical unit: 1 ~ 15300 (cm/Min · ×10 Deg/Min · Inch/Min)

But the highest frequency cannot be greater than 921600 Ps/Sec。

- Parameter 6: homing deceleration speed, the default value is 1000

  Motor and compound unit: 1 ~ 65535 Ps/Sec

  Mechanical unit: 1 ~ 15300 (Cm/Min, x10 Deg/Min, Inch/Min)

- Parameter 7: Gear backlash correction value, default value=0

  Note: Multi-axis linear interpolation command is invalid

Setting range: 0 ~ 32767 Ps。

When walking in reverse, the walking distance will automatically add this value.

FUN141 MPARA	MPARA	FUN141 MPARA

- Parameter 8: Acceleration and deceleration time setting, default value=5000, unit is mS.
  Note: Multi-axis linear interpolation command is invalid
  Setting range: 0~30000 mS.
  This time represents the time required to accelerate from rest to maximum speed (parameter 4), or decelerate from maximum speed to rest.
  The acceleration and deceleration of this system is equal slope control.
  When parameter 12=0, this parameter is used as the deceleration time.
  The acceleration and deceleration control of this system will automatically move in a triangle wave or trapezoid wave according to the actual action stroke.
- Parameter 9: Setting of homing direction and running direction, the default value is 0100H
  Note: Multi-axis linear interpolation command is invalid

	b15 　　b8	b7 　　b0
SR+12	Parameter 9-1	Parameter 9-0

- Parameter 9-0: Running direction setting, the default value is 0
  When the set value = 0, the forward rotation pulse output, the current Ps value will increase
  Reverse the pulse output, the current Ps value will decrease
  When the set value = 1, the forward rotation pulse output, the current Ps value will decrease
  Reverse the pulse output, and increase the current Ps value
- Parameter 9-1：Homing return direction setting, the default value is 1
  When the set value is 0, the homing direction is the current Ps value plus the upward direction (the origin is on the right)
  When the set value = 1, the direction of homing is the direction of decreasing the current Ps value (the origin is on the left)
- Parameter 10: Forward rotation movement correction value, default value=0
  Note: Multi-axis linear interpolation command is invalid
  Setting range: 32768 ~ 32767 Ps
  - When outputting forward rotation pulse wave, this value will be automatically added as the moving distance.
- Parameter 11: Reverse movement compensation value, default value=0 Note: Multi-axis linear interpolation command is invalid
  Setting range:-32768 ~ 32767 Ps
  - When the pulse output is reversed, this value will be automatically added as the moving distance.

- Parameter 12: Deceleration time setting, the default value = 0, the unit is mS
  Note: The multi-axis linear interpolation command is invalid
  - Setting range：0 ~ 30000 mS。
  - When parameter 12 = 0, use parameter 8 as the deceleration time.
  - When parameter 12 ≠ 0, use parameter 12 as the deceleration time.
- Parameter 13: Interpolation acceleration and deceleration time (fixed number) setting, the default value is 500
  Note: Multi-axis line tweening command is dedicated
  - Setting range: 0 ~ 30000 mS
  - It is used to set the time required to accelerate from stillness (speed=0) to the working frequency during linear interpolation motion; this time is also used for deceleration and stop control
- Parameter 14: pulse number/1 revolution, the default value is 0
  - Setting range: 0 ~ 1999999。
  - When parameter 14 = 0, take parameter 1 as pulse number/1 revolution.
  - When parameter 14 ≠ 0, take parameter 14 as pulse number/1 revolution.
- Parameter 15: Control interface I/O setting, the default value isFFFFFFFFH

	b15 b8	b7 b0
SR+19	Parameter 15-1	Parameter 15-0
SR+20	Parameter 15-3	Parameter 15-2

  - Parameter 15-0: Proximity DOG input contact setting; must be the input point of the host (SR+19)

    b6 ~ b0 : Proximity DOG input contact number (0 ~ 15, namely X0 ~ X15)

    b7 = 0: Near-point DOG input is a normally open contact (A or NO contact)
    = 1: The near-point DOG input is a normally closed contact (B or NC contact)

    b7 ~ b0=FFH, no near-point DOG input
  - Parameter 15-1: Travel limit input contact setting (SR+19)

    b14 ~ b8 : Travel limit input contact number (0 ~ 125, namely X0 ~ X125)

    b15 = 0: Travel limit input is a normally open contact (A or NO contact)
    = 1: Travel limit input is a normally closed contact (B or NC contact)

    b15 ~ b8=FFH : No stroke limit input

FUN141 MPARA	MPARA	FUN141 MPARA

- Parameter 15-2: Zero signal PG0 input contact setting; must be the input point of the host (SR+20)

  b6 ~ b0 : Zero signal PG0 input contact number (0 ~ 15, namely X0 ~ X15)

  b7 = 0 : The leading edge of near point DOG starts to count the zero point signal

  = 1 : The trailing edge of the near point DOG starts to count the zero signal

  b7 ~ b0 = FFH : No zero signal PG0 input

- Parameter 15-3: Zero reset signal CLR output contact setting; must be the output point of the host (SR+20)

  b15 ~ b8 : Output contact number of zero reset signal CLR (0 ~ 23, that is, Y0 ~ Y23)

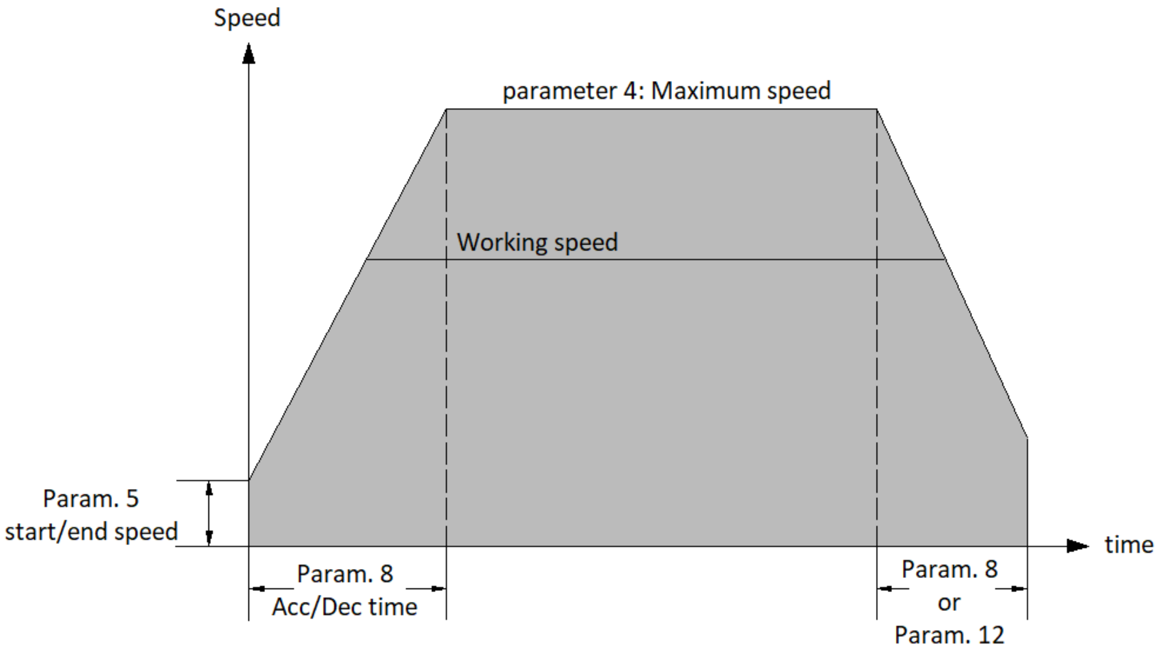
  b15 ~ b8=FFH : CLR output without reset signal

- Parameter 16: Mechanical origin position value, the default value is 0

  -999999 ~ 999999 Ps

- Parameter 17: Zero point signal number, the default value is1

  0 ~ 255 Count



7-16-6 STOP THE HSPSO PULSE OUTPUT (PSOFF)

FUN142 P PSOFF	STOP THE HSPSO PULSE OUTPUT	FUN142 P PSOFF

Symbol

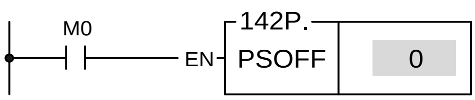
Ladder symbol

N: 0~7
Enforce the Pulse Output PSOn (n= Ps) to stop

Description

- The positioning axis can be controlled up to PSO7, but the actual maximum axis number that can be controlled varies with the host machine model.
- When execution control "EN" =1 or changes from 0→1 (P instruction), this instruction will enforce the assigned number set of HSPSO (High Speed Pulse Output) to stop pulse output.
- While in the application for mechanical original point reset, as soon as reach the original point can use this instruction to stop the pulse output immediately, so as to make the original point stop at the same position every time when performing mechanical original point resetting.

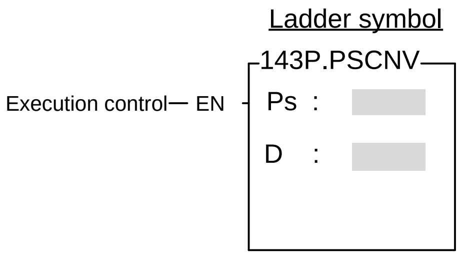
Example

Ladder diagram	ST
	`PSOFF( EN:= M0, Ps:= 0);`
When M0 changes from 0→1, force Ps0 to stop pulse output.	

7-16-7 Convert The Current Pulse Value to Display Value(PSCNV)

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

Symbol



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

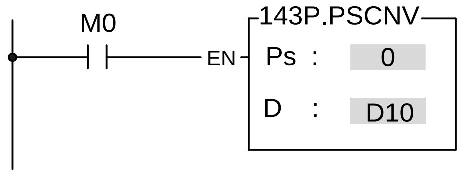
D : Register that stores the current position after conversion. It uses 2 registers, e.g. if D = D10, which means D10 is Low Word and D11 is High Word.

Range / Operand	HR	DR	ROR	K
	R0 ∣ R34767	D0 ∣ D11999	R43224 ∣ R47319	2 ∣ 256
Ps				0～7
D	○	○	○*	

Description

- The positioning axis can be controlled up to PSO7, but the actual maximum axis number that can be controlled varies according to the host machine model.
- When execution control “En” =1 or changes from 0→1 (P instruction), this instruction will convert the assigned current pulse position (PS) to be the mm (or Deg, Inch, or PS) that has same unit as the set value, so as to make current position displaying.
- Only when the FUN140 instruction is executed, then it can get the correct conversion value by executing this instruction.

FUN143 P PSCNV	CONVERT THE CURRENT PULSE VALUE TO DISPLAY VALUE (mm, Deg, Inch, PS)	FUN143 P PSCNV
Example		

Ladder diagram	ST
	PSCNV(EN:= M0, Ps:= 0, D:= D10);

When M0 changes from 0 to 1, convert the current pulse wave position of Ps0 (DR4088) into mm (or Deg or Inch or PS) with the same unit as the set value, and store it in DD10 as the current position display.

FUN144 HSPWM2	HIGH SPEED PULSE WIDTH MODULATION 2	FUN144 HSPWM2

Symbol

PW: Pulse width modulation output point (0=Y0, 1=Y2, 2=Y4, 3=Y6, 4=Y8, 5=Y10, 6=Y12, 7=Y14)

Op: output polarity (0=positive phase, 1=inverted phase)

Hz: output frequency (1~100000000 or 1~200000000, unit 0.001Hz)

OR: Pulse output width (0~100, unit %)

Range / Operand	Y	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	CPU's Yn	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WM19578	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34895	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	
Pw								○				○	○	
Op														0 ~ 1
Hz								○				○	○	
OR								○				○	○	0 ~ 100

Description

1. Compared with FUN139, FUN144 provides more direct and convenient high-speed PWM output control without calculating parameters through built-in formulas.
2. The maximum output frequency may be 100K or 200K depending on the model. If the maximum output frequency exceeds the maximum output frequency, it will not be executed.

7-17 Enable/Disable (FUN145~146)

7-17-1 ENABLE CONTROL OF THE INTERRUPT AND PERIPHERAL

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

Ladder symbol	
Enable control — EN — [145P. EN | LBL]	LBL : External input or peripheral label name that to be enabled.

Description

- When enable control "EN"=1 or changes from 0→1 (P instruction), it allows the external input or peripheral interrupt action which is assigned by LBL.
- The enabled interrupt label name is as follows:(Please refer the section 5.3 for details).

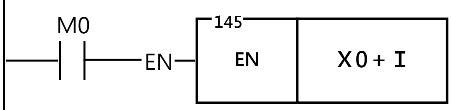
FUN145 P EN	ENABLE CONTROL OF THE INTERRUPT AND PERIPHERAL	FUN145 P EN
Description		

LBL name	Description	LBL name	Description	LBL name	Description
HSC0I	HSC0 High speed counter interrupt	X4−I	X4 negative edge interrupt	LTM2I	10 ms timer LTM2 interrupt
HSC1I	HSC1 High speed counter interrupt	X5+I	X5 positive edge interrupt	LTM3I	10 ms timer LTM3 interrupt
HSC2I	HSC2 High speed counter interrupt	X5−I	X5 negative edge interrupt	HST0I	HST0 High speed counter interrupt
HSC3I	HSC3 High speed counter interrupt	X6+I	X6 positive edge interrupt	HST1I	HST1 High speed counter interrupt
X0+I	X0 positive edge interrupt	X6−I	X6 negative edge interrupt	HST2I	HST2 High speed counter interrupt
X0−I	X0 negative edge interrupt	X7+I	X7 positive edge interrupt	HST3I	HST3 High speed counter interrupt
X1+I	X1 positive edge interrupt	X7−I	X7 negative edge interrupt		
X1−I	X1 negative edge interrupt	STM0I	1 ms timer STM0 interrupt		
X2+I	X2 positive edge interrupt	STM1I	1 ms timer STM1 interrupt		
X2−I	X2 negative edge interrupt	STM2I	1 ms timer STM2 interrupt		
X3+I	X3 positive edge interrupt	STM3I	1 ms timer STM3 interrupt		
X3−I	X3 negative edge interrupt	LTM0I	10 ms timer LTM0 interrupt		
X4+I	X4 positive edge interrupt	LTM1I	10 ms timer LTM1 interrupt		

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

- 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.

Example

<table>
<tr><th>Ladder diagram</th><th>ST</th></tr>
<tr><td>
</td><td><pre>IF M0 THEN
INTEnable(LB:= X0 + I);
END_IF</pre></td></tr>
</table>

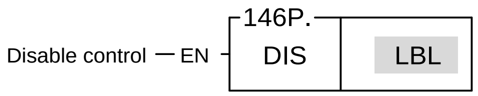
- When M0 changes from 0→1, it allows X0 to send interrupt when X0 changes from 0→1. CPU can rapidly process the interrupt service program of X0+I.

7-17-2 DISABLE CONTROL OF THE INTERRUPT AND PERIPHERAL

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

Symbol

Ladder symbol



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

Description

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

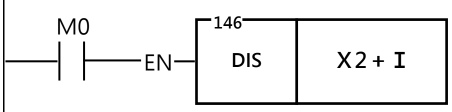
FUN146 P DIS	DISABLE CONTROL OF THE INTERRUPT AND PERIPHERAL	FUN146 P DIS

LBL name	Description	LBL name	Description	LBL name	Description
HSC0I	HSC0 High speed counter interrupt	X4−I	X4 negative edge interrupt	LTM2I	10 ms timer LTM2 interrupt
HSC1I	HSC1 High speed counter interrupt	X5+I	X5 positive edge interrupt	LTM3I	10 ms timer LTM3 interrupt
HSC2I	HSC2 High speed counter interrupt	X5−I	X5 negative edge interrupt	HST0I	HST0 High speed counter interrupt
HSC3I	HSC3 High speed counter interrupt	X6+I	X6 positive edge interrupt	HST1I	HST1 High speed counter interrupt
X0+I	X0 positive edge interrupt	X6−I	X6 negative edge interrupt	HST2I	HST2 High speed counter interrupt
X0−I	X0 negative edge interrupt	X7+I	X7 positive edge interrupt	HST3I	HST3 High speed counter interrupt
X1+I	X1 positive edge interrupt	X7−I	X7 negative edge interrupt		
X1−I	X1 negative edge interrupt	STM0I	1 ms timer STM0 interrupt		
X2+I	X2 positive edge interrupt	STM1I	1 ms timer STM1 interrupt		
X2−I	X2 negative edge interrupt	STM2I	1 ms timer STM2 interrupt		
X3+I	X3 positive edge interrupt	STM3I	1 ms timer STM3 interrupt		
X3−I	X3 negative edge interrupt	LTM0I	10 ms timer LTM0 interrupt		
X4+I	X4 positive edge interrupt	LTM1I	10 ms timer LTM1 interrupt		

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

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

Example

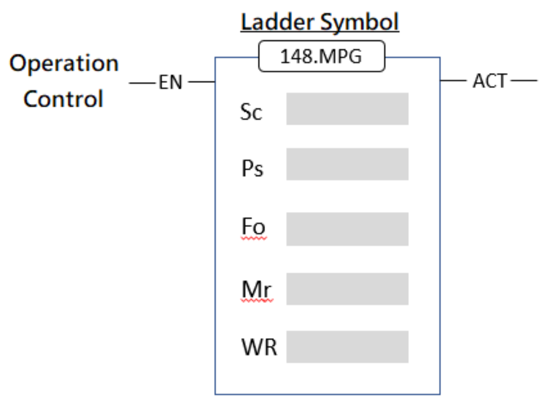
Ladder diagram	ST
	IF M0 THEN INTDisable(LB:= X2+I); END_IF

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

7-18 NC Positioning Instructions II (FUN148)

7-18-1 MANUAL PULSE GENERATOR FOR POSITIONING

FUN148 MPG	MANUAL PULSE GENERATOR FOR POSITIONING	FUN148 MPG
Symbol		

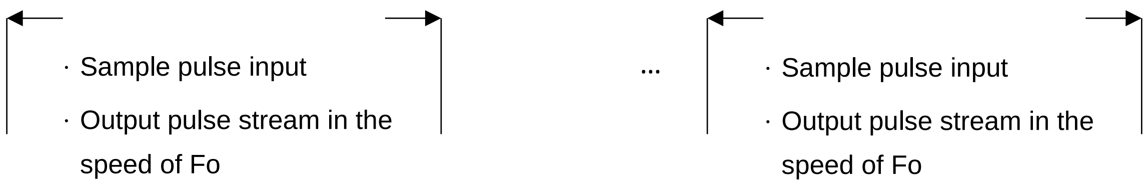


Sc: Source of high-speed counter; 0~7
Ps: Axis of pulse output; 0~3
Fo: Setting of output speed (2 registers)
Mr: Setting of multipliers (2 registers)
Mr+0: Multiplicand (Fa)
Mr+1: Dividend (Fb)
WR: Starting address of working registers, it needs 4 registers

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

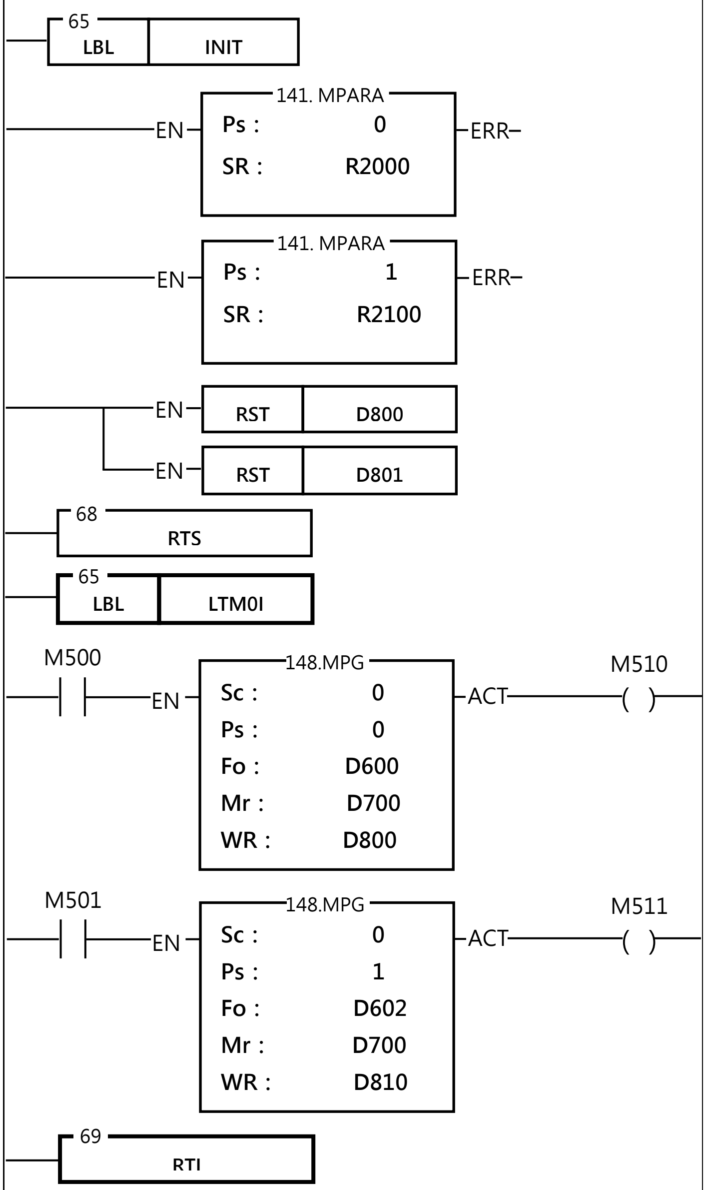
FUN148 MPG	MANUAL PULSE GENERATOR FOR POSITIONING	FUN148 MPG
Description		

- Let this instruction be executed in 10mS fixed time interrupt service routine (PV value set 5， unit times is 10ms,total 50ms，LTM1I)、or by using the 0.1mS high speed timer to generate 10mS fixed time interrupt service to have accurate repeat time to sample the pulse input from manual pulse generator. If it comes the input pulses, it will calculate the number of pulses needing to output according to the setting of multiplier (Mr+0 and Mr+1), and then outputs the pulse stream in the speed of setting (Fo) during this time interval.
- The setting of output speed (Fo) must be fast enough, and the acceleration / deceleration rate (Parameter 4 and parameter 8 of FUN141 instruction) must be sharp to guarantee it can complete the sending of pulse stream during the time interval if it is under high multiplier (100 or 200 times) situation.
- When execution 〝EN〞 =1, this instruction will sample the pulse input from manual pulse generator by reading the current value of assigned high speed counter every time interval; it doesn't have any output if it doesn't have any input pulse; but If it senses the input pulses, it will calculate the number of pulses needing to output according to the setting of multiplier (Mr+0 and Mr+1), and then outputs the pulse stream in the speed of setting (Fo) during this time interval.
- Number of output pulses = (Number of input pulses × Fa) / Fb
- This instruction also under the control of hardware resource management; it wouldn't be executed if the hardware is occupied.
- The output indicator ACT=1 if it outputs the pulses; otherwise ACT=0.
- This instruction will use 4 Registers(WR), other instructions can't share with.
- Please refer to Chapter 13 "The NC Positioning Control of M Serial PLC"of Advanced Application user manual for further details.

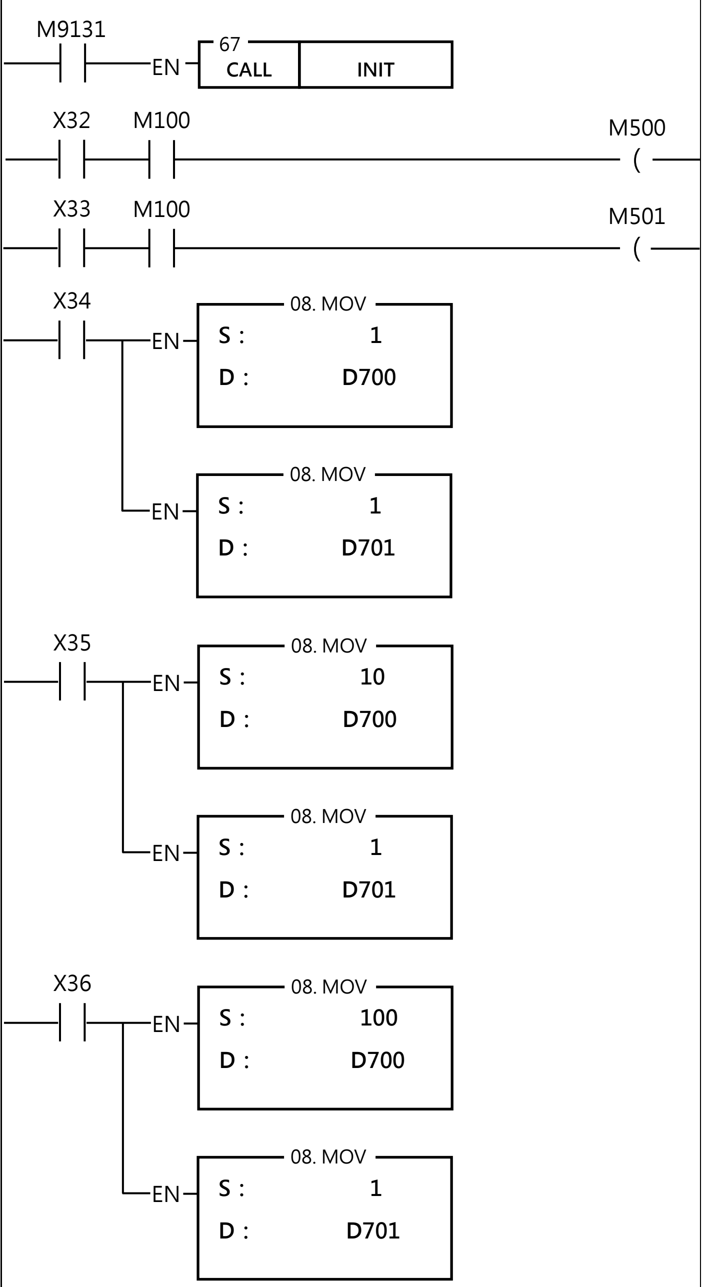


FUN148 MPG	MANUAL PULSE GENERATOR FOR POSITIONING	FUN148 MPG

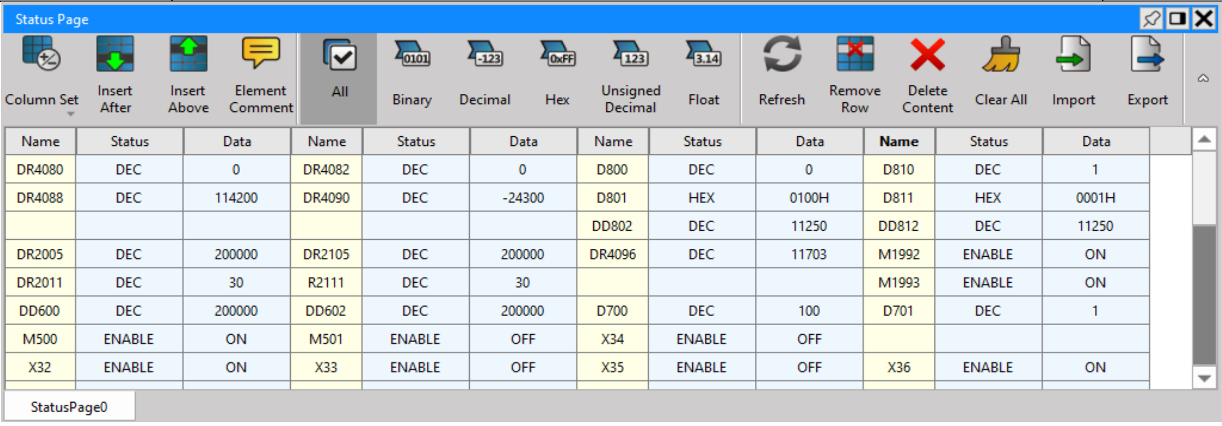
Example 1

Ladder diagram (sub)	ST
	see below


```
LBL (INIT)
MPARA( EN:=TRUE , Ps:=0 ,
SR:=R2000 );
MPARA( EN:=TRUE , Ps:=1 ,
SR:=R2100 );
D800 := 0;
D801 := 0;
RETURN;
LBL (LTM0I)
MPG( EN:= M500, Sc:= 0, Ps:= 0,
Fo:= D600, Mr:= D700, WR:= D800,
ACT=> M510);
MPG( EN:= M501, Sc:= 0, Ps:= 1,
Fo:= D602, Mr:= D700, WR:= D810,
ACT=> M511);
```

<table>
<tr><th>Ladder diagram (main)</th><th>ST</th></tr>
<tr><td>
</td><td>IF M9131 Then
CALL (INIT)
END_IF
IF X32 AND M100 Then
M500 := TRUE;
ELSE
M500 := FALSE;
END_IF
IF X33 AND M100 Then
M501 := TRUE;
ELSE
M501 := FALSE;
END_IF
IF X34 Then
D700 := 1;
D701 := 1;
END_IF
IF X35 Then
D700 := 10;
D701 := 1;
END_IF
IF X36 Then
D700 := 100;
D701 := 1;
END_IF</td></tr>
</table>

FUN148 MPG	MANUAL PULSE GENERATOR FOR POSITIONING	FUN148 MPG

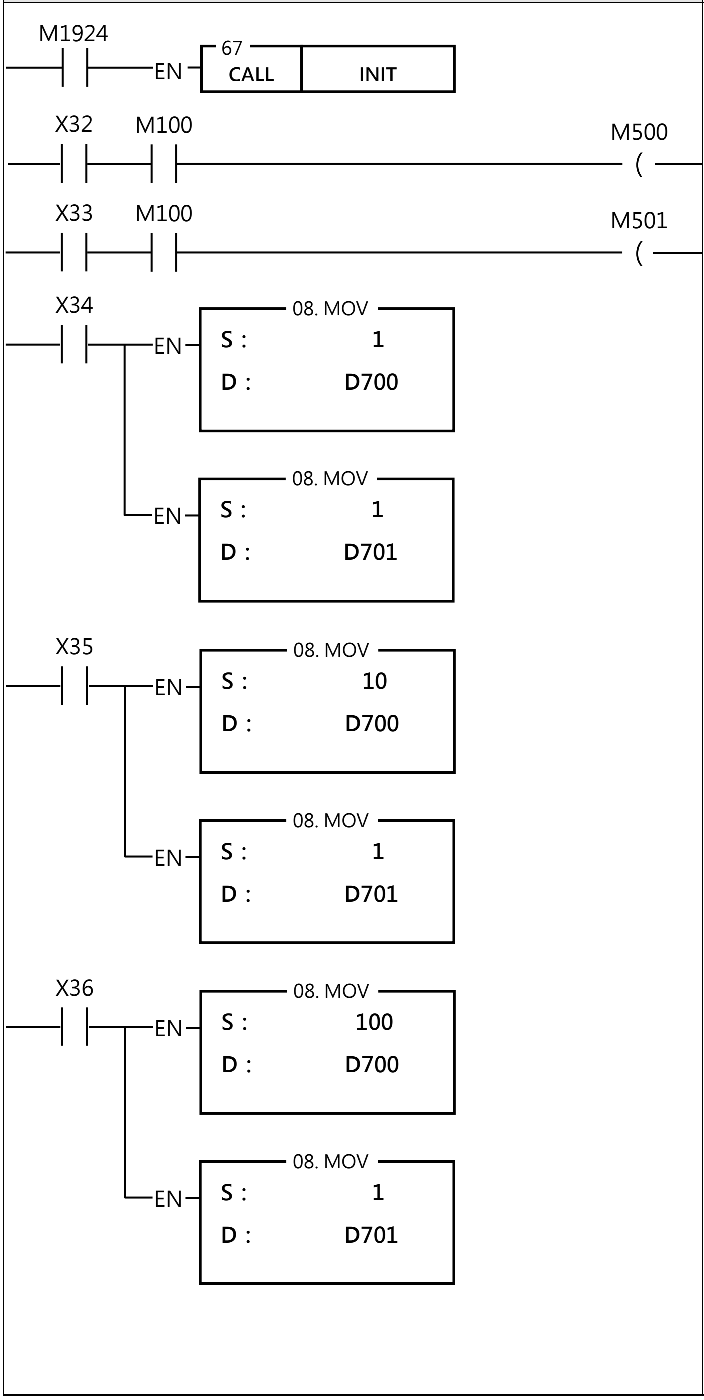


Name	Status	Data	Name	Status	Data	Name	Status	Data	Name	Status	Data
DR4080	DEC	0	DR4082	DEC	0	D800	DEC	0	D810	DEC	1
DR4088	DEC	114200	DR4090	DEC	-24300	D801	HEX	0100H	D811	HEX	0001H
						DD802	DEC	11250	DD812	DEC	11250
DR2005	DEC	200000	DR2105	DEC	200000	DR4096	DEC	11703	M1992	ENABLE	ON
DR2011	DEC	30	R2111	DEC	30				M1993	ENABLE	ON
DD600	DEC	200000	DD602	DEC	200000	D700	DEC	100	D701	DEC	1
M500	ENABLE	ON	M501	ENABLE	OFF	X34	ENABLE	OFF			
X32	ENABLE	ON	X33	ENABLE	OFF	X35	ENABLE	OFF	X36	ENABLE	ON

X32: Select the 0st axis (Ps0)
X33: Select the 1st axis (Ps1)
X34: output magnification is 1
X35: output magnification is 10
X36: output magnification is 100
M100: Manual wheel action selection
DR2005: Maximum output frequency of axis 0 (parameter 4 of FUN141 command); 200K Hz
R2011: Acceleration and deceleration time of the 0th axis (parameter 8 of the FUN141 instruction); 30mS
DD600: 0th axis manual wheel actuation output frequency; 200K Hz
DR2105: The maximum output frequency of the first axis (parameter 4 of the FUN141 command); 200K Hz
R2111: Acceleration and deceleration time of the first axis (parameter 8 of FUN141 instruction); 30mS
DD602: 1st axis manual wheel actuation output frequency; 200K Hz
Example description: Put the manual wheel positioning processing instructions of Ps0 and Ps1 in the 50MSI timing interrupt processing program.
When X32=1 and M100=1, start Ps0 hand wheel positioning processing; each interval (50mS) will sample the hand wheel input pulse (from HSC0); if no pulse input is sampled, FUN148 The command will not output; if there is a sampled pulse wave input, the output pulse number will be calculated according to the multiplier setting (D700 and D701), and then the calculated output pulse number will be output at the output frequency set by DD600.
Output pulse number = (HSC0 input pulse number in interval time×D700)/D701

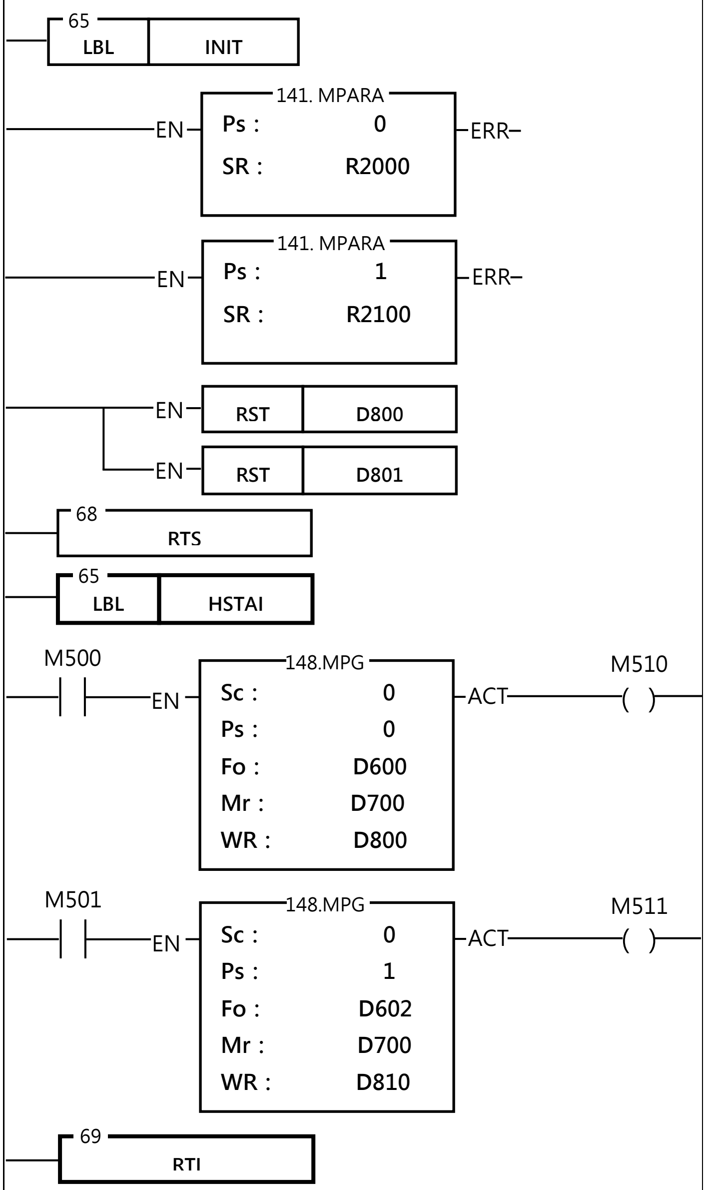
FUN148 MPG	MANUAL PULSE GENERATOR FOR POSITIONING	FUN148 MPG
Example 2		

Ladder diagram (main)	ST




```
IF M1924 Then
CALL (INIT)
END_IF
IF X32 AND M100 Then
M500 := TRUE;
ELSE
M500 := FALSE;
END_IF
IF X33 AND M100 Then
M501 := TRUE;
ELSE
M501 := FALSE;
END_IF
IF X34 Then
D700 := 1;
D701 := 1;
END_IF
IF X35 Then
D700 := 10;
D701 := 1;
END_IF
IF X36 Then
D700 := 100;
D701 := 1;
END_IF
```

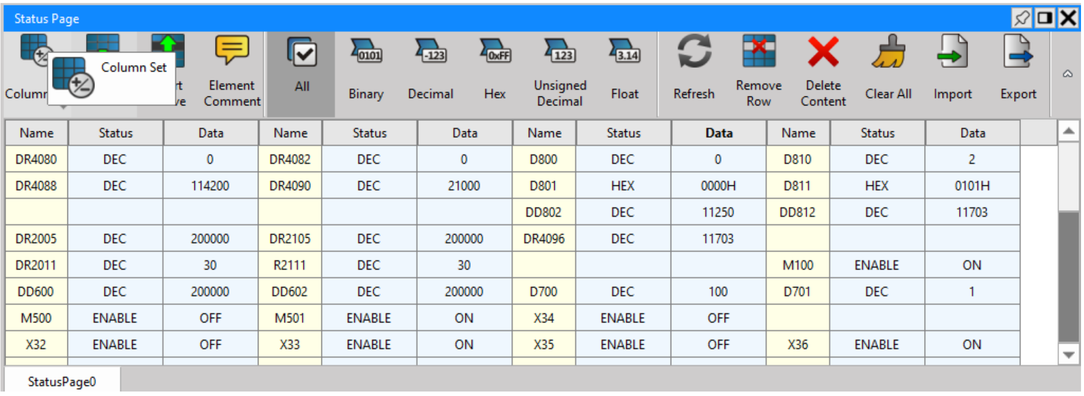
Ladder diagram (sub)



ST

```
LBL (INIT)
MPARA( EN:=TRUE , Ps:=0 ,
SR:=R2000 );
MPARA( EN:=TRUE , Ps:=1 ,
SR:=R2100 );
D800 := 0;
D801 := 0;
RETURN;
LBL (HSTAI)
MPG( EN:= M500, Sc:= 0, Ps:= 0,
Fo:= D600, Mr:= D700, WR:= D800,
ACT=> M510);
MPG( EN:= M501, Sc:= 0, Ps:= 1,
Fo:= D602, Mr:= D700, WR:= D810,
ACT=> M511);
```

FUN148 MPG	MANUAL PULSE GENERATOR FOR POSITIONING	FUN148 MPG



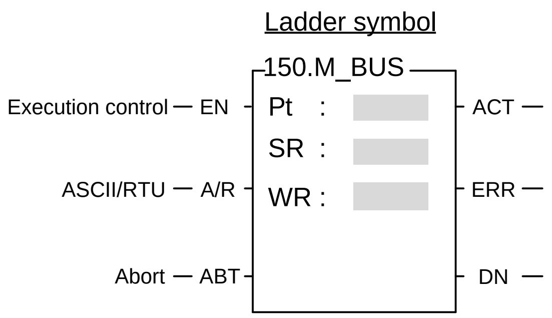
Name	Status	Data	Name	Status	Data	Name	Status	Data	Name	Status	Data
DR4080	DEC	0	DR4082	DEC	0	D800	DEC	0	D810	DEC	2
DR4088	DEC	114200	DR4090	DEC	21000	D801	HEX	0000H	D811	HEX	0101H
						DD802	DEC	11250	DD812	DEC	11703
DR2005	DEC	200000	DR2105	DEC	200000	DR4096	DEC	11703			
DR2011	DEC	30	R2111	DEC	30				M100	ENABLE	ON
DD600	DEC	200000	DD602	DEC	200000	D700	DEC	100	D701	DEC	1
M500	ENABLE	OFF	M501	ENABLE	ON	X34	ENABLE	OFF			
X32	ENABLE	OFF	X33	ENABLE	ON	X35	ENABLE	OFF	X36	ENABLE	ON

X32: Select the 0th axis (Ps0)
X33: Select the 1st axis (Ps1)
X34: output magnification is 1
X35: output magnification is 10
X36: output magnification is 100
M100: Manual wheel action selection
DR2005: Maximum output frequency of axis 0 (parameter 4 of FUN141 command); 200K Hz
R2011: Acceleration and deceleration time of the 0th axis (parameter 8 of the FUN141 instruction); 30mS
DD600: 0th axis manual wheel actuation output frequency; 200K Hz
DR2105: The maximum output frequency of the first axis (parameter 4 of the FUN141 command); 200K Hz
R2111: Acceleration and deceleration time of the first axis (parameter 8 of FUN141 instruction); 30mS
DD602: 1st axis manual wheel actuation output frequency; 200K Hz
Example description: Set the 0.1mS high-speed timer (HSTA) as a 50mS timer interrupt, and put the manual wheel positioning processing instructions of Ps0 and Ps1 in the HSTAI interrupt processing program.
When X33=1 and M100=1, start Ps1 hand wheel positioning processing; each interval (50mS) will sample the hand wheel input pulse (from HSC0); if no pulse input is sampled, FUN148 There will be no output for the command; if there is a sampled pulse wave input, the output pulse number will be calculated according to the multiplier setting (D700 and D701), and then the calculated output pulse number will be output at the output frequency set by DD602.
Output pulse number = (HSC0 input pulse number in interval time×D700)/D701

7-19 Communication Instruction (FUN150~156)

7-19-1 MODBUS MASTER INSTRUCTION(M-BUS)

FUN150 M-BUS	MODBUS MASTER INSTRUCTION （WHICH MAKES PLC AS THE MODBUS MASTER THROUGH PORT 1~2）	FUN150 M-BUS
Symbol		



Pt ：1~2, specify the communication port being acted as the Modbus master

SR：Starting register of communication program

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

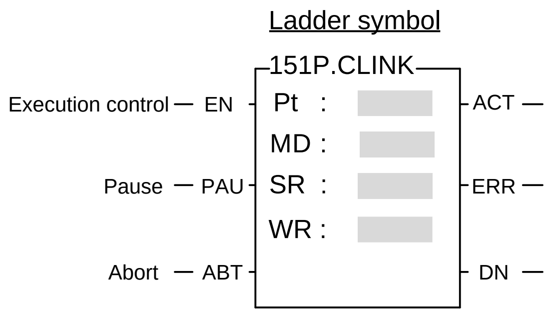
Range / Operand	HR R0 ∣ R34767	ROR R43224 ∣ R47319	DR D0 ∣ D11999	K
Pt				1-2
SR	○	○	○	
WR	○	○	○	

FUN150 M-BUS	MODBUS MASTER INSTRUCTION （WHICH MAKES PLC AS THE MODBUS MASTER THROUGH PORT 1~2）	FUN150 M-BUS
Description		

- FUN150 (M-BUS) instruction makes PLC act as Modbus master through Port 1 ~ 2, thus it is very easy to communicate with the intelligent peripheral with Modbus RTU/ASCII protocol.
- The master PLC may connect with 247 slave stations through the RS-485 interface.
- Only the master PLC needs to use Modbus RTU/ASCII instruction.
- 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 7 registries to make definition; every 7 registers define one packet of data transaction.
- When execution control 〝EN〞 changes from 0→1 and Abort“ABT”is 0, and if Port 1/2 hasn't been controlled by other communication instructions [i.e. M9135(Port1) / M9138(Port2)], this instruction will control the Port 1/2 immediately and set the M9135/M9138 to be 0 (which means it is being occupied), then going on a packet of data transaction immediately. If Port 1/2 has been controlled (M9135/M9138 = 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 (M9135/M9138 =1), and then this instruction will become enactive, set M9135/M9138 to be 0, and going on the data transaction immediately. 。
- While in transaction processing, if operation control “ABT” becomes 1, this instruction will abort this transaction immediately and release the control right (M9135/M9138 = 1). Next time, when this instruction takes over the transmission right again, it will restart from the first packet of data transaction. 。
- While 〝A/R〞 =0，Modbus RTU protocol； 〝A/R〞 =1，Modbus ASCII protocol.
- While it is in the data transaction, the output indication “ACT” will be ON.
- If there is error occurred when it finishes a packet of data transaction, the output indication “DN” & “ERR” will be ON.
- If there is no error occurred when it finishes a packet of data transaction, the output indication “DN” will be ON.
- For detailed application examples, please refer to Chapter 11 "Ethernet Function and Ethernet Communication" of the Advanced Software User Manual.

7-19-2 COMMUNICATION LINK INSTRUCTION (CLINK)

FUN151 CLINK	COMMUNICATION LINK INSTRUCTION (WHICH MAKES PLC ACT AS THE MASTER STATION IN CPU LINK NETWORK THROUGH PORT 1~2)	FUN151 CLINK
Symbol		



Pt: Assign the port, 1~2

MD: Communication mode, MD0~MD1

SR: Starting register of communication table

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

Range / Operand	HR R0 | R34767	ROR R43224 | R47319	DR D0 | D11999	K
Pt				1-4
MD				0-3
SR	○	○	○	
WR	○	○	○	

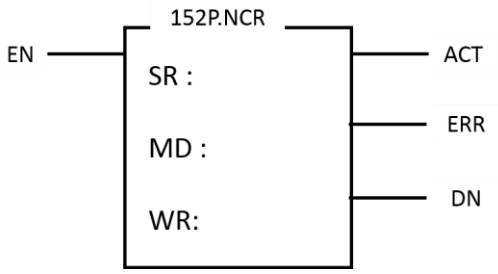
Description	

- This instruction provides MD0 ~ MD1. The following are the function description of respective modes.
- FUN151 (CLINK) : MD 0, it makes PLC act as the master of FATEK CPU Link Network through Port 1~2
- The master PLC may connect with 254 slave stations through the RS485 interface.
- Only the master PLC needs to use FUN151 instruction, the slave doesn't need.
- 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 7 registries to make definition; every 7 registers define one packet of data transaction.
- When execution control 〝EN〞 changes from 0→1 and both inputs“PAU”and “ABT”are 0, and if Port 1/2 hasn't been controlled by other communication instructions [i.e. M9135 (Port1) / M9138 (Port2) = 1], this instruction will control the Port 1/2 immediately and set the M9135/M9138 to be 0 (which means it is being occupied), then going on a packet of data transaction immediately. If Port 1/2 has been controlled (M9135/M9138= 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 (M9135/M9138 =1), and then this instruction will become enactive, set M9135/M9138 to be 0, and going on the data transaction immediately.
- While in transaction processing, if operation control“PAU”becomes 1, this instruction will release the control right (M9135/M9138 = 1) after this transaction. Next time, when this instruction takes over the transmission right again, it will restart from the next packet of data transaction.
- While in transaction processing, if operation control“ABT”becomes 1, this instruction will abort this transaction immediately and release the control right (M9135/M9138 = 1). Next time, when this instruction takes over the transmission right again, it will restart from the first packet of data transaction.
- While it is in the data transaction, the output indication“ACT”will be ON.
- If there is error occurred when it finishes a packet of data transaction, the output indication“DN”&“ERR”will be ON.
- If there is no error occurred when it finishes a packet of data transaction, the output indicatio “DN”will be ON.
- Please refer to Chapter 10.4 “The Applications for M-Series PLC Communication Link”

7-19-3 Network Active Communication (NCR)

FUN152 NCR	Network Active Communication	FUN152 NCR

Symbol



SR: Table starting register address
MD: Modbus TCP active communication (=1)
WR: Working register

Range / Ope-rand	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	K
	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	2 | 256
SR	○	○	○	○	○	○	○	○*	○*	○	
MD	○	○	○	○	○	○	○	○*	○*	○	1
WR	○	○	○	○	○	○	○	○*	○*	○	

Description

1. The FUN152 (NCR) command is connected to the smart peripheral (slave station) with the Modbus communication protocol through the Ethernet port.
2. This command is mainly based on the specified form, such as using the Modbus Master TCP form, read or write according to the specified form, and actively carry out network communication. The communication form must be set before use; only six registers are defined, and every 6 registers define a transfer transaction.
3. When EN is ON for this command, the communication will continue.
4. When the data transaction is being transmitted, the output indication "ACT" is ON.

FUN152 NCR	Network Active Communication	FUN152 NCR
Description		

SR occupies successive register

SR	Word Size	Purpose	Description
SR + 0	1	Identifying word: 0x544D	For identifying effective table: ‘M’,‘T’
SR + 1	1	Total lots of data transaction	Each individual communication is expressed by 6 units of registers.
SR + 2	2	Remote IP	
SR + 4	1	Remote port	
SR + 5	1	Maintain TCP online	= 0. Creating one lot of online for each individual communication. = 1. Maintain one lot of TCP online in the table.
SR + 6	1	Overtime setting	Unit : 10 ms
SR + 7	1	Re-test count	
SR + 8	1	Command code (Lot#1)	= 1. Read = 2. Write =3. Write in individual lot
SR + 9	1	Data length	Register: 1~125 Contact: 1~255
SR + 10	1	Type of Master PLC data	Please refer to 1~3 and 12~13 indicated in the description of Data Type Table provided below.
SR + 11	1	Starting number of Master PLC data.	For effective scope, please refer to the details described in the Data Type Table provided below.

FUN152 NCR	Network active communication	FUN152 NCR
Description		

SR	Word Size	Purpose	Description
SR + 12	1	Type of Slave PLC data	For 0, 1, 3 and 4, please refer to Modbus Data Type Table
SR + 13	1	Starting number of Master PLC data	For effective scope, please refer to the details described in the Modbus Data Type Table.
SR + 14	1	Slave PLC data type	Please refer to the Modbus data type table
SR + 15	1	Starting number of Slave PLC data	Please refer to the Modbus data type table
SR + 16	1	Command Code (Lot#2)	
…	…	…	

Data Type Table

Data Code	Data Type	Scope
1	Y (output relay)	0~1023
2	M (internal relay)	0~29599
3	S (step relay)	0~3103
12	R (data register)	0~34767
13	D (data register)	0~11999

Data Type Table

Data Code	Data Type	Scope
0	Output or internal relay	1~65535
4	Data register	1~65535
1	Contact input	1~65535
3	Input register	1~65535

Note: The type of master and slave data must be consistent. In other words, if the Master Station is set as Y/M/S, then the Slave Station must be set as 0/1. Likewise, if the Master Station is set as R/D, then the Slave Station must be set as 3/4; and vice versa.

FUN152 NCR	Network active communication	FUN152 NCR
Example		

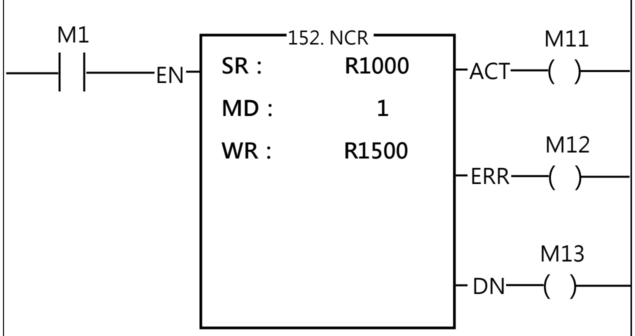
FUN152 Instruction Operand WR Description

	High Byte Low Byte	
WR+0	Current communication index	Current communication index: which transaction is in operation (counting from 0)
WR+1	Result code	The result code stores the operation result, 0=normal; other values, abnormal
WR+2	Function code	Function code, please refer to the following description
WR+3	Internal TCP connection index	
WR+4	Connection status	Connection status, WR+4 =0, communication is successful. =2, waiting for reply. =3, communication timeout. =4, in connection. =5, communication error
WR+5	Retries	

Function code: Low Byte is valid;
=0, read slave PLC system status;
=1, read data from slave PLC;
=2, write data to slave PLC

Result code: 0, communication transaction is successful.
2, The data length error
(The value is 0, or the transaction volume is greater than the upper limit).
3, The command code is wrong (the value is 0 or greater than 3).
4, The data type error (refer to the data type code)
5, The data number is wrong (refer to the starting number of the data).
6, The master and slave data types are different (for example, the master station is Y/M/S, while the slave station is 4).
7, Communication port error (only Port 1, 2, 3 or 4).
8, Illegal communication forms.
A, The slave station does not respond (Time-out exception).
B. Communication is abnormal (wrong data is received or slave station responds with error message).
C, Connection error

FUN152 NCR	Network active communication	FUN152 NCR
Example	Slave Station (IP: 192.168.0.151) 400101~ 400105 -> Master Station (IP: 192.168.0.150) R100~R104	

Ladder diagram	ST
	`NCR( EN:= M1, SR:= R1000, MD:= 1, WR:= R1500, ACT=> M11, ERR=> M12, DN=> M13);`

Description

When the input control "EN" changes from 0 to 1, based on the settings in the Modbus TCP table, the remote IP slave station reads the register data and stores it in the PLC master station, and continuously completes the data transaction.

The setting steps are as follows.
First add the Modbus Master form in the data form.
In the Modbus Master form, define the remote IP and Port, and the address to be read and written, including the data of the master station and the data of the slave station.
Edit the Fun152 NCR instruction on the Ladder of the master station.

FUN152 NCR	Network active communication	FUN152 NCR
Example		

Editing Communication Forms with UperLogic

Click in the project window

Communication Command Table: Project Name

Data Table

Modbus Master Table →

After right clicking, click "Add Modbus" Master Table" with a form type of "Modbus TCP Table",

Or on the "Project" tab, click "Data Form", drop down to select "Modbus Master Table", select "Add Modbus Master Table", table Type "Modbus TCP table" is also acceptable.

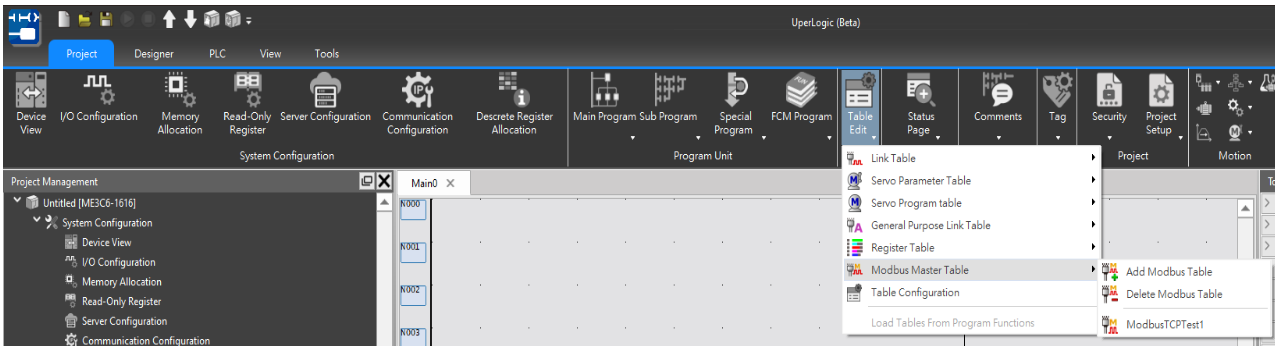


Fig. 86: Add Modbus Master Table

FUN152 NCR	Network active communication	FUN152 NCR
Example		

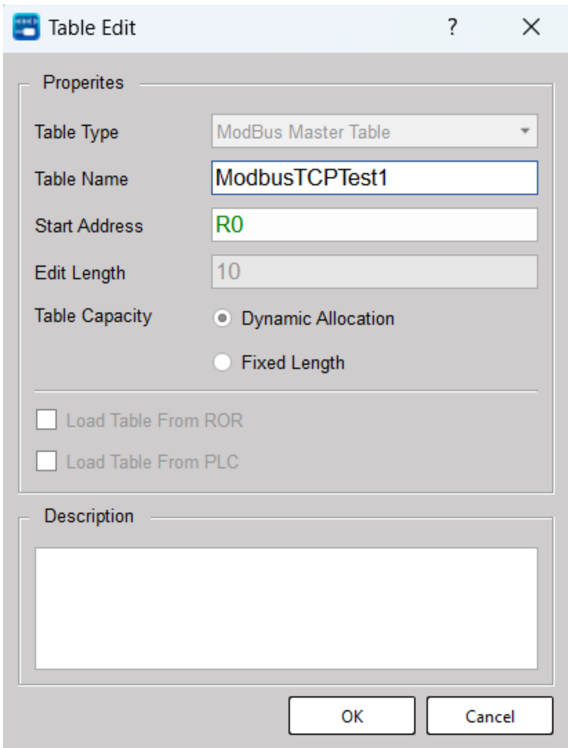


Fig. 87: Edit Modbus Master Table

- Table Type: Select "Modbus TCP Table".
- Table name: You can enter an easily identifiable name for the connection form, which is convenient for future modification or debugging.
- Table start position: Input the start position of the start register SR of the communication program (data transmission form) used by the communication command (FUN152).

FUN152 NCR	Network active communication	FUN152 NCR
Example		

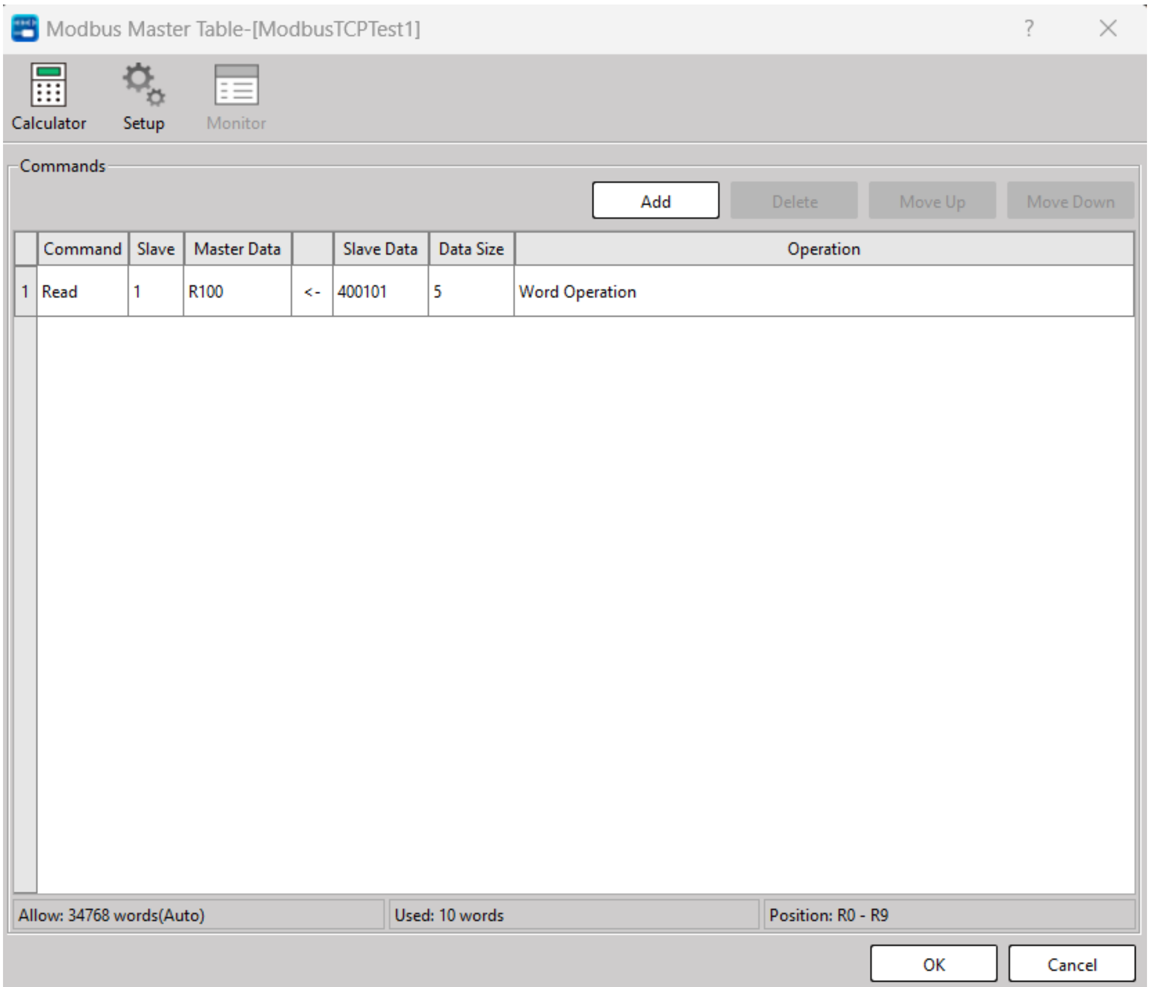
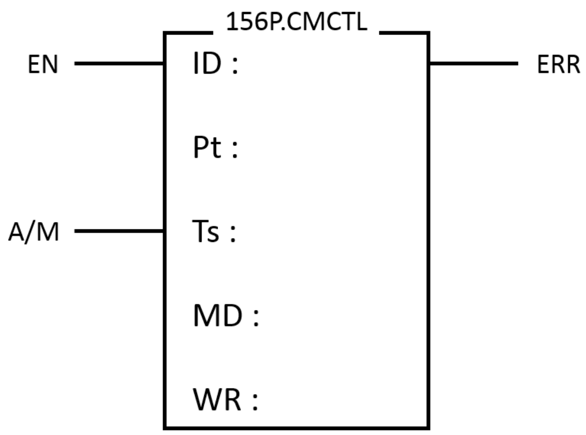


Fig. 88: Modbus Master Table

- Remote IP: The IP address of the remote device.
- Remote port number: The port number of the remote device.
- Command: The master station reads the data from the Modbus slave station, or writes data to the Modbus slave station.
- Master station data: In the read operation, it is the location where the data is read from the slave station and stored, and in the write operation, it is the location from the master station to write the data to the slave station.

- Slave station information: The slave station wants to send back the position of the master station during the read operation, and the position of writing data from the master station to the slave station during the input operation. ◦
- Length: The length to be transmitted, the read length is 125, and the write length is 123.
- Connection maintenance: When starting, it will only initiate a TCP connection establishment request for the remote IP, and subsequent communications will exchange data on this connection; otherwise, it will re-establish a TCP connection for each communication.

7-19-4 CMCTL

FUN156P CMCTL	CMCTL	FUN156P CMCTL
Symbol		
	ID : Used module number Pt : Appointed COMA/COMB (A= 0;B=1) Ts : Communication table mask Bit 1 : Table 1 Bit 2 : Table 2 | Bit 15 : Table 15 Bit 16~Bit 31 : Reserved. Do not use. MD : Set mode 0 : RUN ONCE 1 : RUN CYCLING 2 : STOP WR : Saving operations state Bit0~Bit1 : Table 0 state Bit2~Bit3 : Table 1 state Bit30~Bit31 : Table 15 state = 0 : RUN_ONCE = 1 : RUN_CYCLING = 2 : STOP,	

FUN156P CMCTL	CMCTL	FUN156P CMCTL

Range / Operand	HR	OR	SR	ROR	DR	K
	R0 | R34767	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	
ID						0-127
Pt						0-23
Ts	○	○	○	○	○	0-63
MD	○	○	○	○	○	0-63
WR	○			○	○	5word

Description

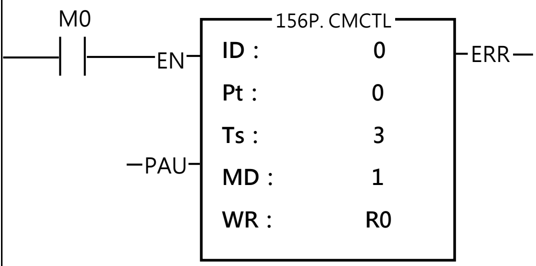
Such command should be used with the CM25 and CM55 communication modules. Before each use, it is required to set up the communication module data.

- EN OFF->ON will carry out communication control, ON->OFF will stop
- PAU is not yet supported
- The communication status code of each table will be updated in the allocated status register, and the address can be confirmed by using the device view

FUN156：WR Description

Bits	Status	Values
Bit0 Bit1	Table0 Status	Table0 = 0(00): RUN_ONCE = 1(01): RUN_CYCLING = 2(10): STOP
Bit2 Bit3	Table1 Status	Table1 = 0(00): RUN_ONCE = 1(01): RUN_CYCLING = 2(10): STOP
Bit4 Bit5	Table2 Status	Table2 = 0(00): RUN_ONCE = 1(01): RUN_CYCLING = 2(10): STOP
Bit6 Bit7	Table3 Status	Table3 = 0(00): RUN_ONCE = 1(01): RUN_CYCLING = 2(10): STOP
Bit30 Bit31	Table15 Status	Table15 = 0(00): RUN_ONCE = 1(01): RUN_CYCLING = 2(10): STOP

Reserved after Bit32

Ladder diagram	ST
 	`CMCTL( EN:= M0, PAU:= M1, ID:= 0, Pt:= 0, Ts:= 3, MD:= 1, WR:= R0, ERR=> M2);`

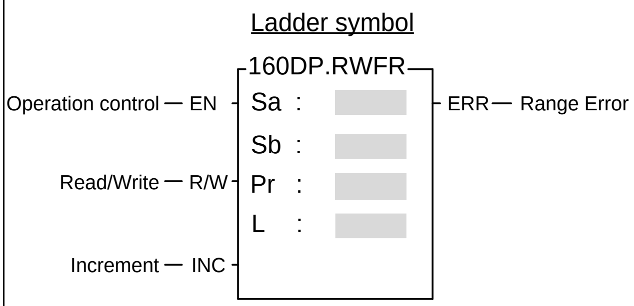
As indicated in the figure above, when M0 becomes 1, the command will open Port 0 of the No. #0 module and then start the communication according to Table 1 and Table 2 (0001b+0010b=0011b and then 3(10) is obtained). Next, select RUN CYCLING Mode and then RO for use as the working register.

7-20 Data Movement Instructions (FUN160~162)

7-20-1 Read/Write File Register

FUN160 D P RWFR	Read/Write File Register	FUN160 D P RWFR

Symbol



Sa : Starting address of data register
Sb : Starting address of file register
Pr : Record pointer register
L : Quantity of register to form a record, 1~511

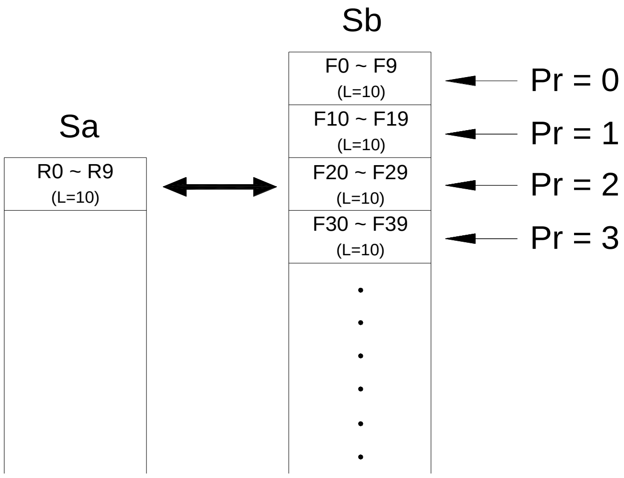
Sa operand can combine V、Z、P0~P9 for index addressing.

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	FR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999		V,Z P0-P9	D0 | D11999
Sa	○	○	○	○	○	○	○	○	○	○	○	○		○	
Sb															○
Pr		○	○	○	○	○	○		○	○*	○*	○			
L							○				○*	○	1-511		

Description

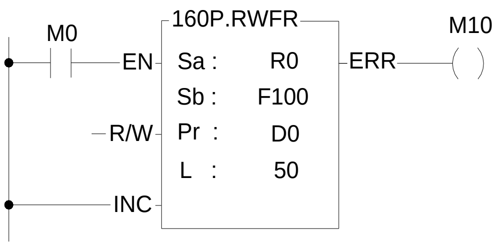
- When operation control "EN"=1 or changes from 0→1(P instruction), it will perform the read ("R/W"=1) or write ("R/W"=0) file register operation. While reading, the content of data registers starting from Sa will be overwritten by the content of file registers addressed by the base file register Sb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

FUN160DP RWFR	Read/Write File Register	FUN160DP 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).

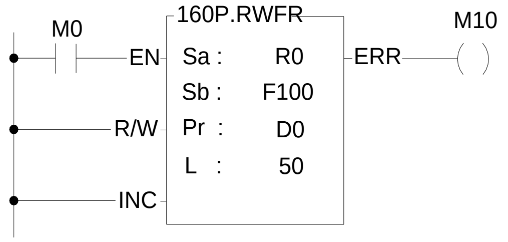
Example 1

<table>
<tr><th>Ladder diagram</th><th>ST</th></tr>
<tr><td>
</td><td><pre>ReadWriteFileReg(EN:=M0,
                  R_W:= FALSE,
                  INC:= TRUE,
                  Sa:= R0,
                  Sb:= F100,
                  Pr:= D0,
                  L:= 50);</pre></td></tr>
</table>

- 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.

FUN160DP RWFR	Read/Write File Register	FUN160DP RWFR
Example 2		

Ladder diagram	ST

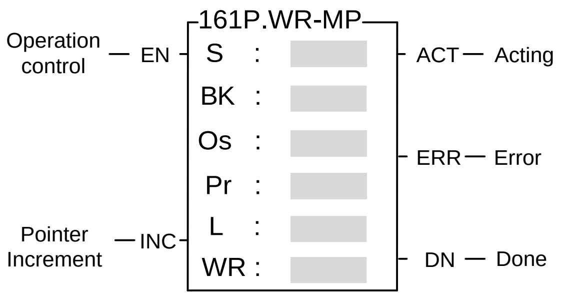
	`ReadWriteFileReg( EN:=M0,`
	`R_W:= TRUE,`
	`INC:= TRUE,`
	`Sa:= R0,`
	`Sb:= F100,`
	`Pr:= D0,`
	`L:= 50);`

- 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.

7-20-2 Write SD Card (WR-MP)

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

Ladder symbol



S：Starting address of the source data

BK：Block number of the MEMORY_PACK，0 ~ 1

Os：Offset of the block

Pr：Address of the pointer

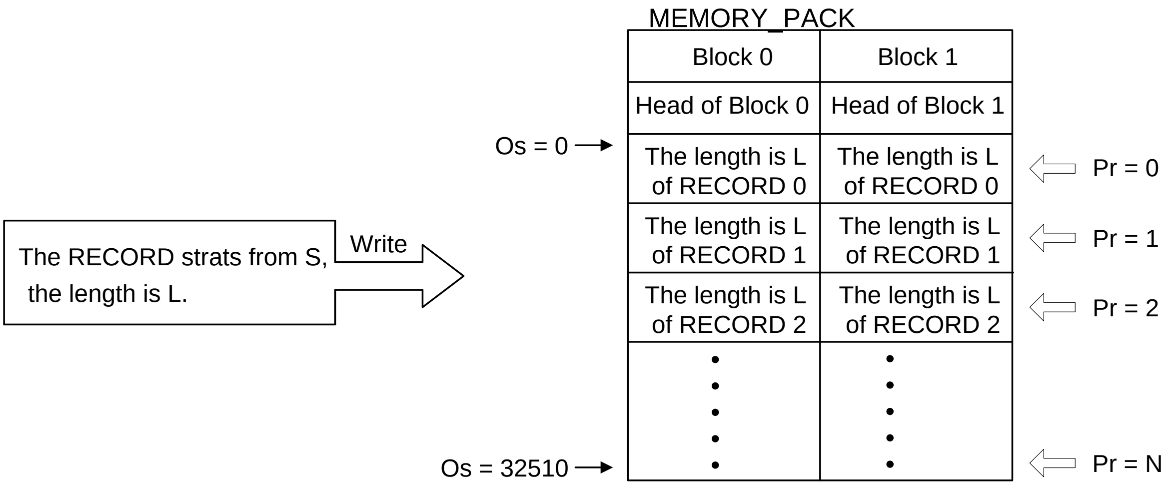
L：Quantity of writing，1 ~ 128

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

Range / Operand	HR	ROR	DR	K	XR
	R0 | R34767	R43224 | R47319	D0 | D11999		V,Z P0-P9
S	○	○	○		○
Bk				0-1	
Os	○	○	○	0-32510	
Pr	○	○*	○		
L	○	○*	○	1-128	
WR	○	○*	○		

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

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

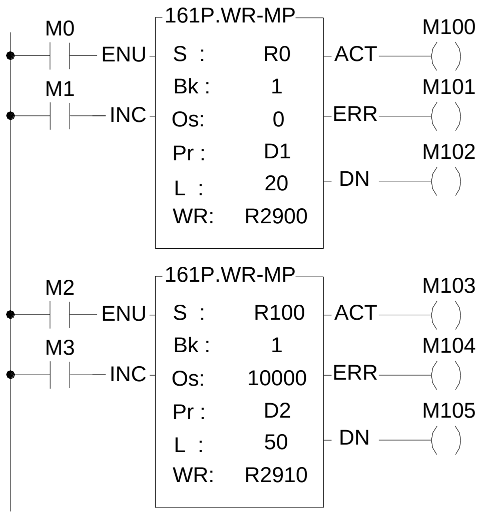


- When input "INC" = 1, the content of the pointer will be increased by one after the execution of writing, it points to next record.
- If the value of L is equal to 0 or greater than 128, or the pointed data area over the range, the output "ERR" will be 1, it will not perform the writing operation.

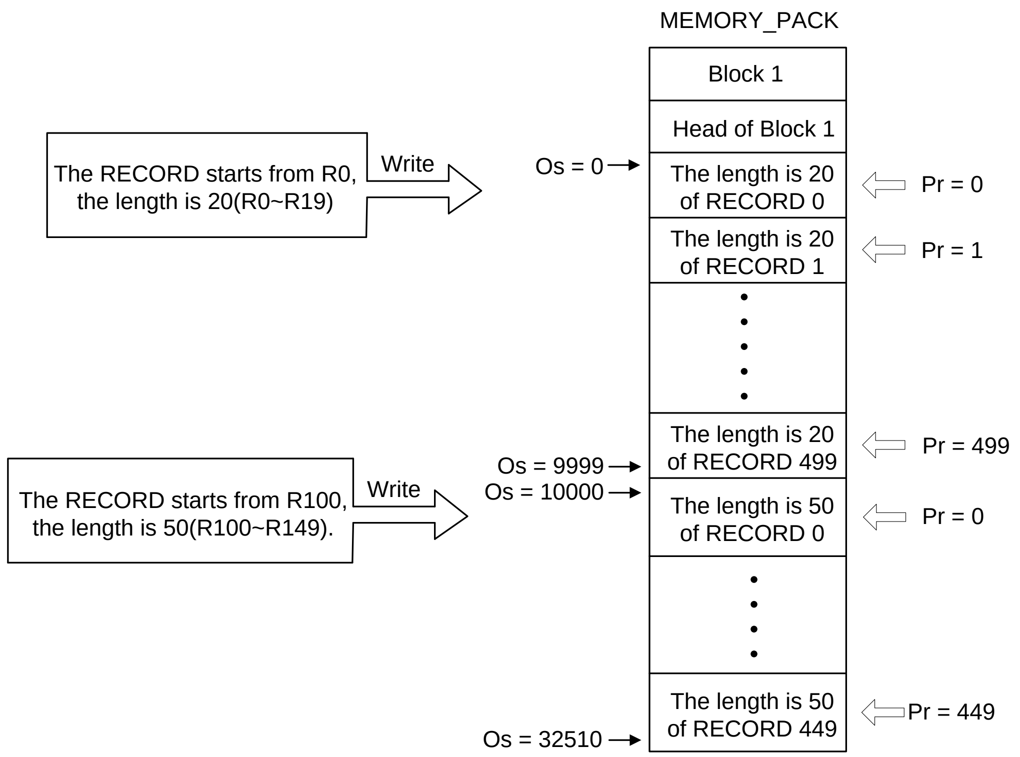
FUN161P WR-MP	Write Data Record into the MEMORY_PACK (Write memory pack)	FUN161P WR-MP

- It needs couple of PLC solving scans for data writing and verification; during the execution, the output "ACT" will be 1; when completing the execution and verification without the error, the output "DN" will be 1; when completing the execution and verification with the error, the output "ERR" will be 1.
- M-Series PLC MEMORY_PACK can be configured to store the user's ladder program or machine's working parameters, or both. The ladder program can be stored into the block 0 only, but the machine's working parameters can be stored into block 0 or 1; the memory capacity of each block has 32K Word in total.

Example	Writing the record into block 1 with the different length

Ladder diagram	ST
	WriteSDMem(EN:= M0, INC:= M1, S:= R0, Bk:= 1, Os:= 0, Pr:= D1, L:= 20, WR:= R2900, ACT=> M100, ERR=> M101, DN=> M102); WriteSDMem(EN:= M2, INC:= M3, S:= R100, Bk:= 1, Os:= 10000, Pr:= D2, L:= 50, WR:= R2910, ACT=> M103, ERR=> M104, DN=> M105);

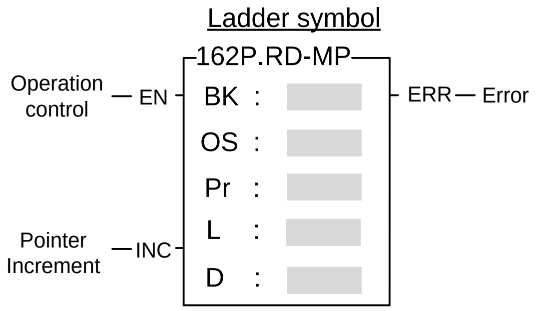
FUN161P WR-MP	Write Data Record into the MEMORY_PACK (Write memory pack)	FUN161P WR-MP



7-20-3 Read SD Card (RD-MP)

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

Symbol



BK：Block number of the MEMORY_PACK，0 ~ 1

Os：Offset of the block

Pr：Address of the pointer

L：Quantity of reading，1 ~ 128

D：Starting address to store the reading record

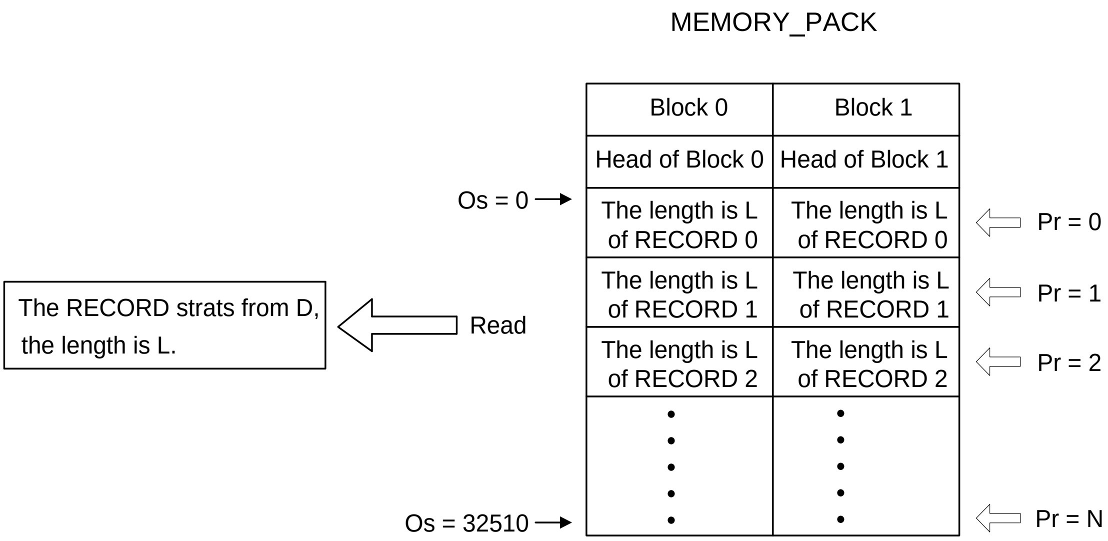
Range / Operand	HR	ROR	DR	K
	R0 | R34767	R43224 | R47319	D0 | D11999	
Bk				
Os	○	○	○	0-1
Pr	○	○*	○	0-32510
L	○	○*	○	1-128
D	○	○*	○	

Description

- If the MEMORY_PACK of the M-Series PLC has stored the data record written by the FUN161 instruction, they can be read out for machine's working through this instruction, it will reduce the tuning time for machine operation.
- When execution control "EN" = 1 or from 0→1(P instruction), it will perform the data reading, where BK is the block number of the MEMORY_PACK storing the record, Os is the offset of specified block, Pr is the pointer to point to corresponding data area, L is the quantity of this record, and D is the starting address to stor this reading of record. The access of MEMORY_PACK manipulation adopts the concept of RECORD data structure to implement with.

  The working diagram as shown below :

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

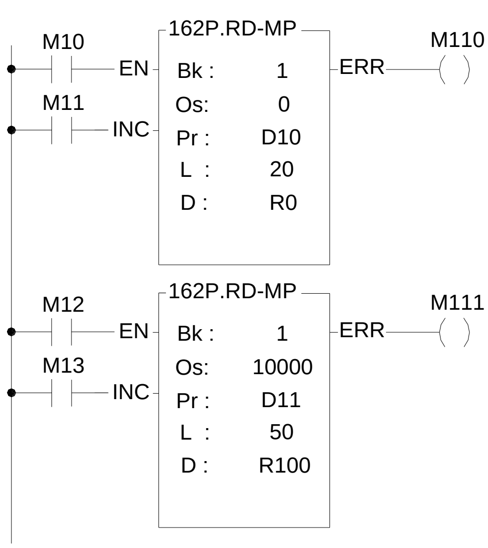


- When input "INC"=1, the content of the pointer will be increased by one after the execution of reading, it points to next record.
- If the value of L is equal to 0 or greater than 128, or the pointed data area over the range, the output "ERR" will be 1, it will not perform the reading operation.

FUN162 P RD-MP	Read Data Record from the MEMORY_PACK （Read memory pack）	FUN162 P RD-MP
Example	Reading the record from block 1 with the different length	

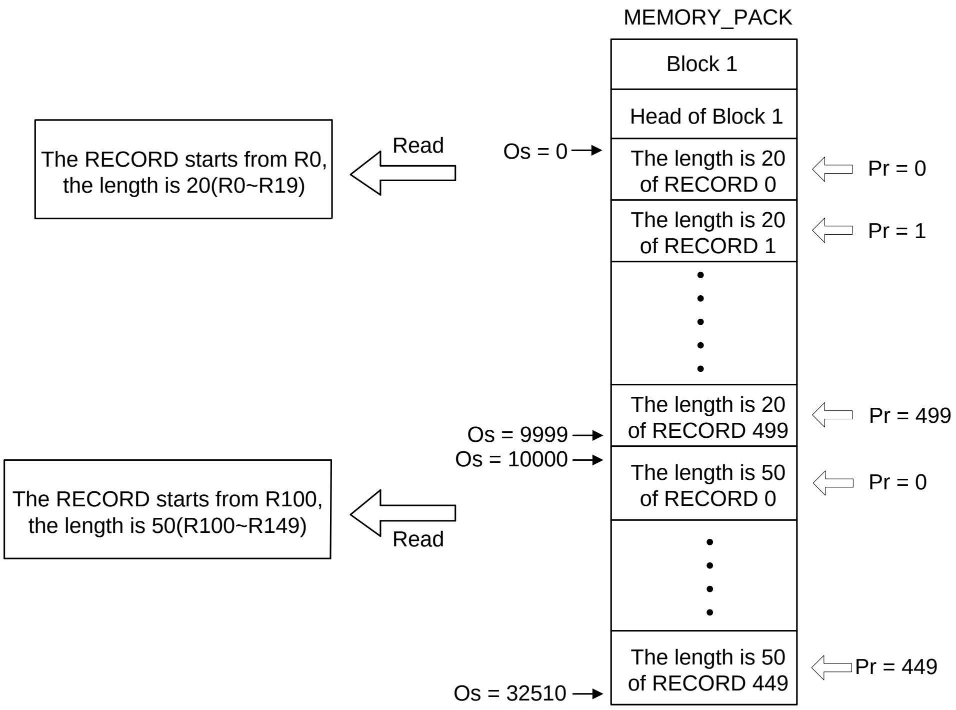
※ It is necessary that correct data in MEMORY_PACK or this example can’t execute correctly.

Ladder diagram



ST

```
ReadSDMem( EN:=M10,
           INC:= M11,
           Bk:= 1,
           Os:=0,
           Pr:= D10,
           L:= 20,
           D:= R0,
           ERR=> M110);
ReadSDMem( EN:=M12,
           INC:= M13,
           Bk:= 1,
           Os:= 10000,
           Pr:= D11,
           L:= 50,
           D:= R100,
           ERR=> M111);
```

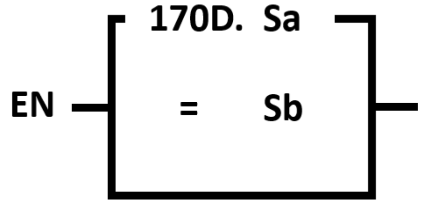


7-21 In Line Comparison Instruction (FUN170~175)

7-21-1 Equal To Compare

FUN170 D =	EQUAL TO COMPARE （Compare whether Sa is equal to Sb）	FUN170 D =

Symbol



Sa：Operand A or the starting address of Sa

Sb：Operand B or the starting address of Sb

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

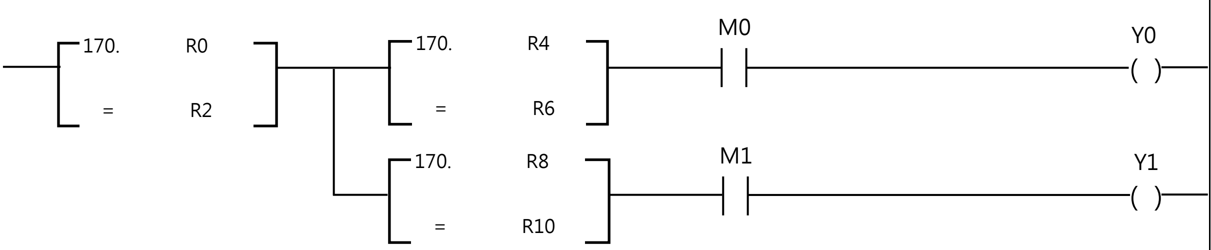
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +- numbers	V,Z P0-P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Description

- When execution input 〝EN〞 =1, this instruction will be executed in signed number to compare Sa with Sb. If Sa=Sb, the output is 1; otherwise the output is 0.

FUN170 D =	EQUAL TO COMPARE (Compare whether Sa is equal to Sb)	FUN170 D =
Example 1		

Ladder diagram



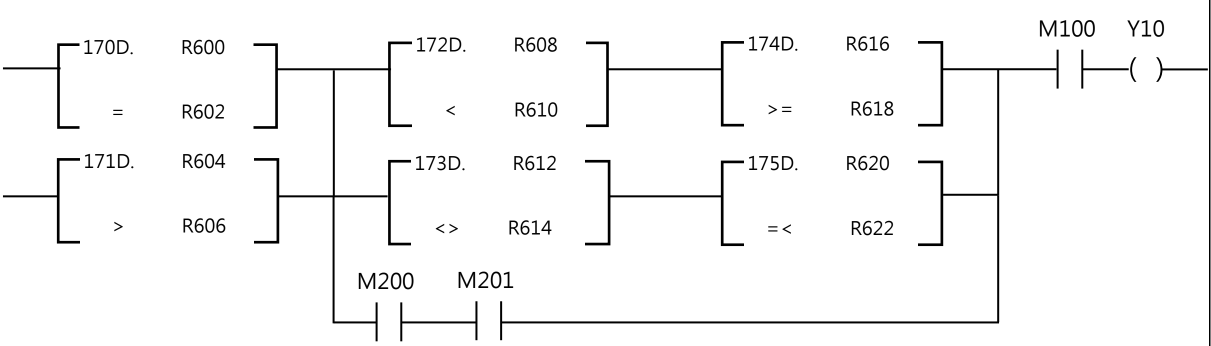
ST

```
IF R0 = R2 AND R4 = R6 AND M0 Then
Y0 := TRUE;
ELSEIF R0 = R2 AND R8 = R10 AND M1 Then
Y1 := TRUE;
ELSE
Y0 := FALSE;
Y1 := FALSE;
END_IF
```

Description: When R0=R2、R4=R6 and M0=1, the output status of Y0 is 1; otherwise it is 0.
R0=R2、R8=R10 and M1=1, the output status of Y1 is 1; otherwise it is 0.

Example 2	

Ladder diagram



ST

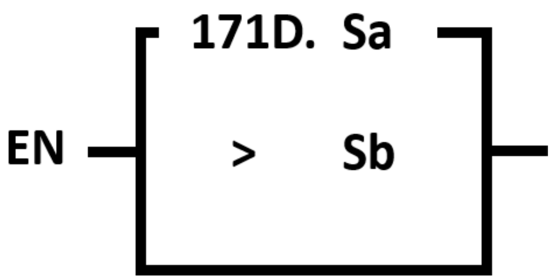
```
IF R600 = R602 OR R604 > R606 Then
IF (R608 < R610 AND R616 >= R618) OR (R612 <> R614 AND R620 =< R622) OR
(M200 AND M201) Then
IF M100 Then
Y10 := TRUE;
END_IF
END_IF
ELSE
Y10 := FALSE;
END_IF
```

Description: When DR600=DR602 or DR604>DR606, after them DR608<DR610 and DR616≧DR618, or DR612≠DR614 and DR620≦DR622, or M200=1and M201=1, and then M100=1, the output status of Y10 is 1; otherwise it is 0.

7-21-2 GREATER THAN COMPARE

FUN171 D >	GREATER THAN COMPARE （Compare whether Sa is greater than Sb）	FUN171 D >

Symbol



Sa：Operand A or the starting address of Sa

Sb：Operand B or the starting address of Sb

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

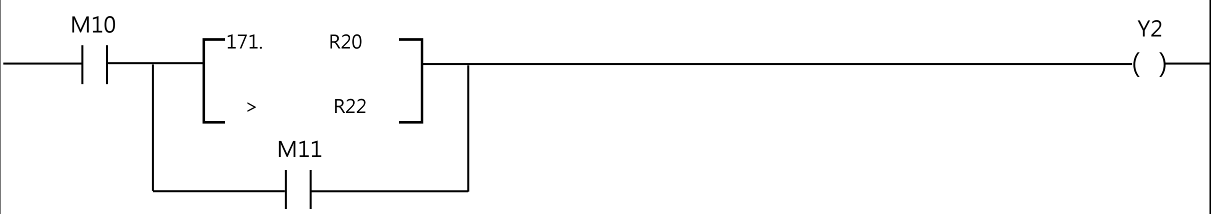
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +- numbers	V,Z P0-P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Description

- When execution input 〝EN〞=1, this instruction will be executed in signed number to compare Sa with Sb. If Sa>Sb, the output is 1; otherwise the output is 0.

Example 1

Ladder diagram

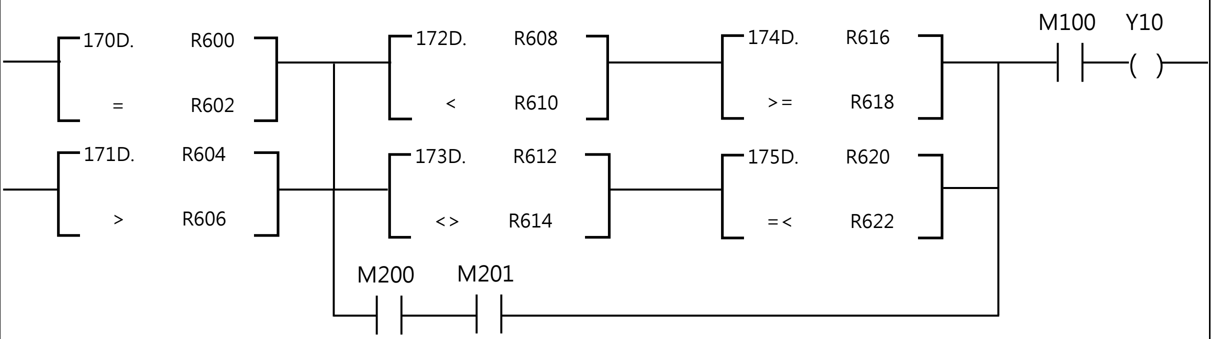


ST

```
IF M10 AND (R20 > R22 OR M11) Then
Y2 := TRUE;
ELSE
Y2 := FALSE;
END_IF
```

Example 2	

Ladder diagram



ST

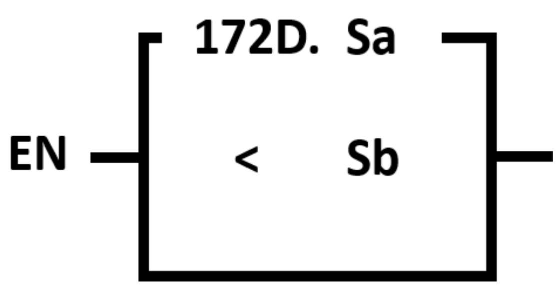
```
IF R600 = R602 OR R604 > R606 Then
IF (R608 < R610 AND R616 >= R618) OR (R612 <> R614 AND R620 =< R622) OR
(M200 AND M201) Then
IF M100 Then
Y10 := TRUE;
END_IF
END_IF
ELSE
Y10 := FALSE;
END_IF
```

Description: When DR600=DR602 or DR604>DR606, after them DR608<DR610 and DR616≧DR618, or DR612≠DR614 and DR620≦DR622, or M200=1and M201=1, and then M100=1, the output status of Y10 is 1; otherwise it is 0.

7-21-3 LESS THAN COMPARE

FUN172 D <	LESS THAN COMPARE (Compare whether Sa is less than Sb)	FUN172 D <

Symbol



Sa ： Operand A or the starting address of Sa

Sb ： Operand B or the starting address of Sb

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

Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +- numbers	V,Z P0-P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Description

- When execution input 〝EN〞 =1, this instruction will be executed in signed number to compare Sa with Sb. If Sa<Sb, the output is 1; otherwise the output is 0.

Example 1

Ladder diagram

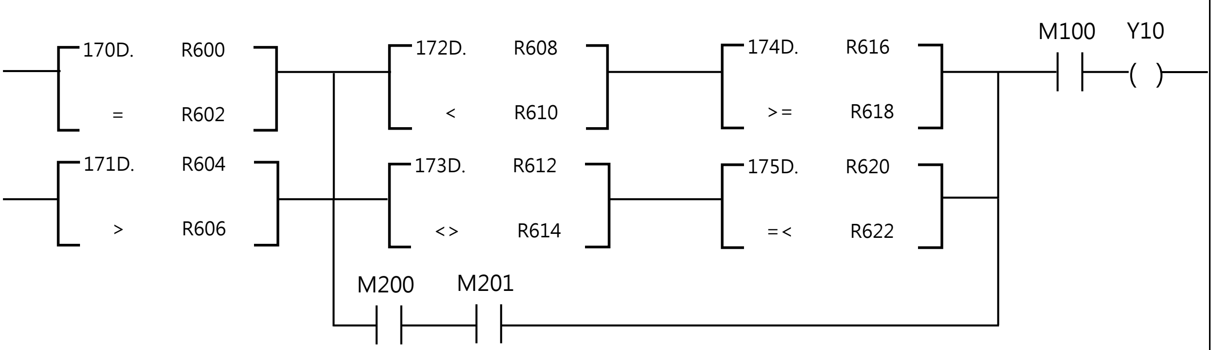
ST

```
IF M10 AND (R20 < R22 OR M11) Then
Y2 := TRUE;
ELSE
Y2 := FALSE;
END_IF
```

Description: When M10=1 、R20 < R22 or M11=1, the output status of Y2 is 1; otherwise it is 0.

FUN172 D <	LESS THAN COMPARE （Compare whether Sa is less than Sb）	FUN172 D <
Example 2		

Ladder diagram



ST

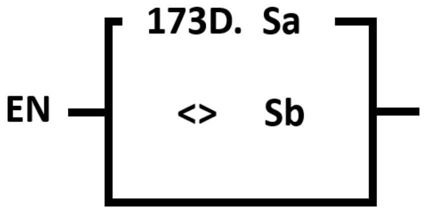
```
IF R600 = R602 OR R604 > R606 Then
IF (R608 < R610 AND R616 >= R618) OR (R612 <> R614 AND R620 =< R622) OR
(M200 AND M201) Then
IF M100 Then
Y10 := TRUE;
END_IF
END_IF
ELSE
Y10 := FALSE;
END_IF
```

Description: When DR600=DR602 or DR604>DR606, after them DR608<DR610 and DR616≧DR618, or DR612≠DR614 and DR620≦DR622, or M200=1and M201=1, and then M100=1, the output status of Y10 is 1; otherwise it is 0.

7-21-4 Not Equal To Compare

FUN173 D <>	NOT EQUAL TO COMPARE (Compare whether Sa is not equal to Sb)	FUN173 D <>

Symbol



Sa：Operand A or the starting address of Sa

Sb：Operand B or the starting address of Sb

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

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +- numbers	V,Z P0-P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Description

- When execution input〝EN〞=1, this instruction will be executed in signed number to compare Sa with Sb. If Sa≠Sb, the output is 1; otherwise the output is 0.

Example 1

Ladder diagram

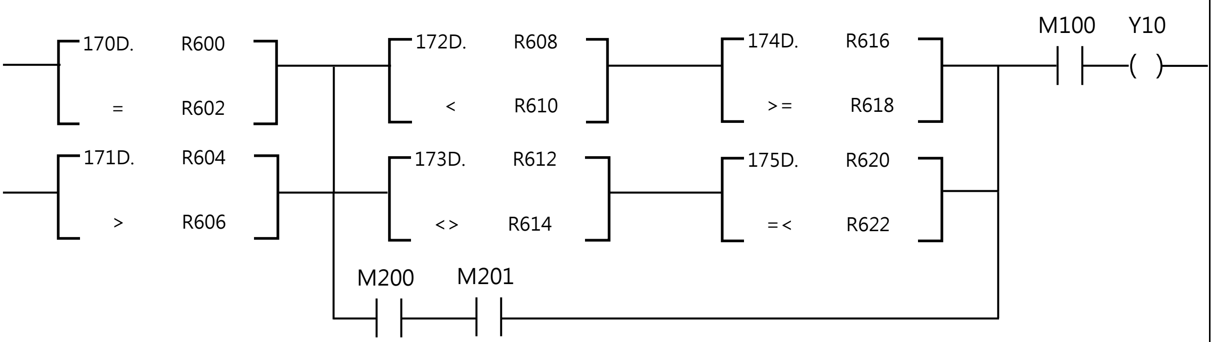
ST

```
IF M10 AND (R20 <> R22 OR M11) Then
Y2 := TRUE;
ELSE
Y2 := FALSE;
END IF
```

Description: When M10=1、R20≠R22 or M11=1, the output status of Y2 is 1; otherwise it is 0.

FUN173 D <>	NOT EQUAL TO COMPARE (Compare whether Sa is not equal to Sb)	FUN173 D <>
Example 2		

Ladder diagram



ST

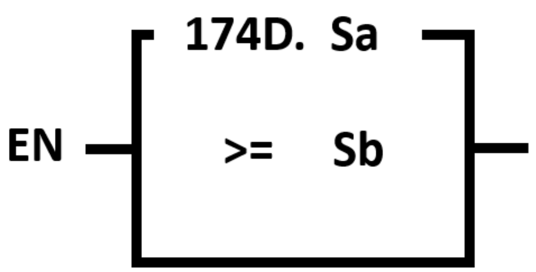
```
IF R600 = R602 OR R604 > R606 Then
IF (R608 < R610 AND R616 >= R618) OR (R612 <> R614 AND R620 =< R622) OR
(M200 AND M201) Then
IF M100 Then
Y10 := TRUE;
END_IF
END_IF
ELSE
Y10 := FALSE;
END_IF
```

Description: When DR600=DR602 or DR604>DR606, after them DR608<DR610 and DR616≧DR618, or DR612≠DR614 and DR620≦DR622, or M200=1and M201=1, and then M100=1, the output status of Y10 is 1; otherwise it is 0.

7-21-5 GREATER THAN OR EQUAL TO COMPARE

FUN174 D >=	GREATER THAN OR EQUAL TO COMPARE	FUN174 D >=

Symbol



Sa ： Operand A or the starting address of Sa

Sb ： Operand B or the starting address of Sb

Sa 、Sb may combine with V 、Z 、P0 ~ P9 for indirect addressing application.

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WY29584	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R34895	R35024 ∣ R35151	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999	16/32-bit + numbers	V,Z P0-P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Description

- When execution input 〝EN〞=1, this instruction will be executed in signed number to compare Sa with Sb. If Sa≧Sb, the output is 1; otherwise the output is 0.

Example 1

Ladder diagram

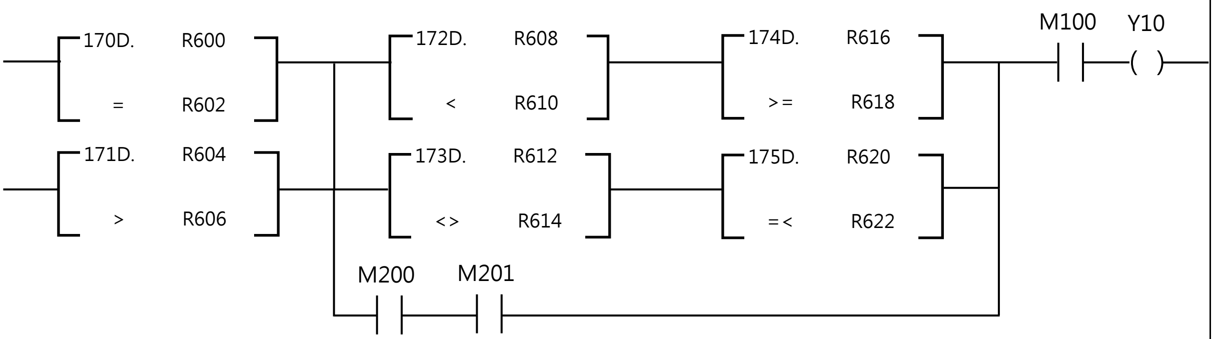
ST

```
IF M10 AND (R20 >= R22 OR M11) Then
Y2 := TRUE;
ELSE
Y2 := FALSE;
END_IF
```

Description: When M10=1 、R20≧R22 or M11=1, the output status of Y2 is 1; otherwise it is 0.

FUN174 D >=	GREATER THAN OR EQUAL TO COMPARE	FUN174 D >=
Example 2		

Ladder diagram



ST

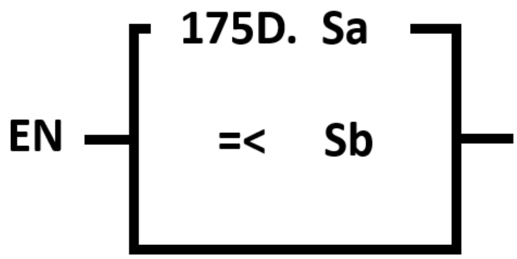
```
IF R600 = R602 OR R604 > R606 Then
IF (R608 < R610 AND R616 >= R618) OR (R612 <> R614 AND R620 =< R622) OR
(M200 AND M201) Then
IF M100 Then
Y10 := TRUE;
END_IF
END_IF
ELSE
Y10 := FALSE;
END_IF
```

Description: When DR600=DR602 or DR604>DR606, after them DR608<DR610 and DR616≧DR618, or DR612≠DR614 and DR620≦DR622, or M200=1and M201=1, and then M100=1, the output status of Y10 is 1; otherwise it is 0.

7-21-6 LESS THAN OR EQUAL TO COMPARE

FUN175 D =<	LESS THAN OR EQUAL TO COMPARE	FUN175 D =<

Symbol



Sa ：Operand A or the starting address of Sa

Sb ：Operand B or the starting address of Sb

Sa 、Sb may combine with V 、Z 、P0 ~ P9 for indirect addressing application.

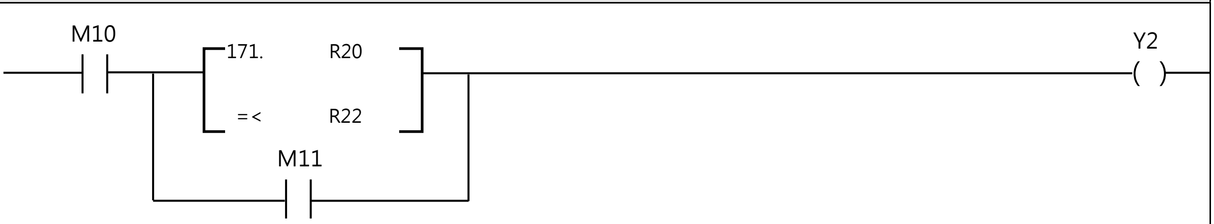
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WY29584	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999	16/32-bit +- numbers	V,Z P0-P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Description

- When execution input 〝EN〞 =1, this instruction will be executed in signed number to compare Sa with Sb. If Sa≦Sb, the output is 1; otherwise the output is 0.

Example 1

Ladder diagram



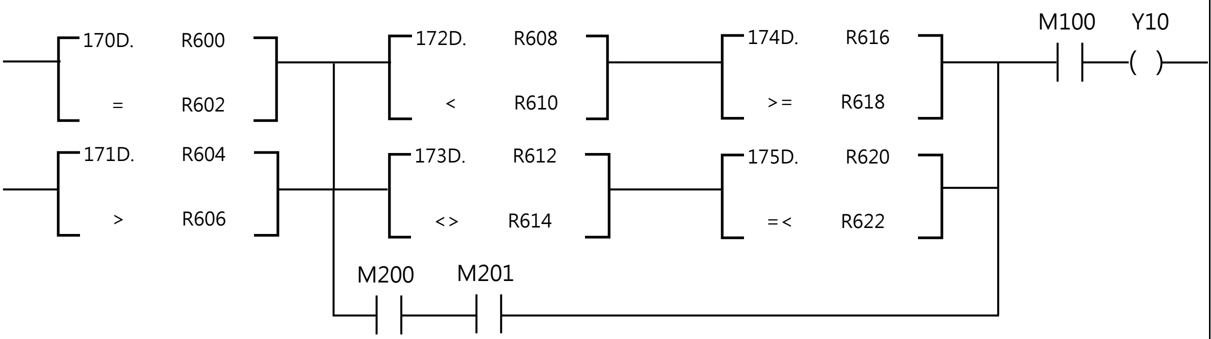
ST

```
IF M10 AND (R20 >= R22 OR M11) Then
Y2 := TRUE;
ELSE
Y2 := FALSE;
END_IF
```

Description: When M10=1 、R20≦R22 or M11=1, the output status of Y2 is 1; otherwise it is 0.

FUN175 D =<	LESS THAN OR EQUAL TO COMPARE	FUN175 D =<
Example 2		

Ladder diagram



ST

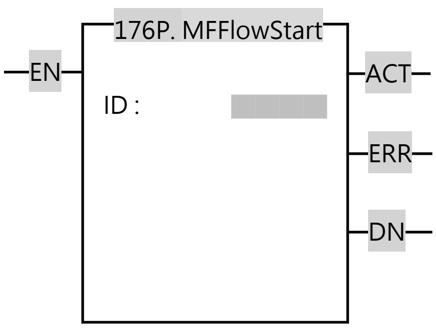
```
IF R600 = R602 OR R604 > R606 Then
IF (R608 < R610 AND R616 >= R618) OR (R612 <> R614 AND R620 =< R622) OR
(M200 AND M201) Then
IF M100 Then
Y10 := TRUE;
END_IF
END_IF
ELSE
Y10 := FALSE;
END_IF
```

Description: When DR600=DR602 or DR604>DR606, after them DR608<DR610 and DR616≧DR618, or DR612≠DR614 and DR620≦DR622, or M200=1and M201=1, and then M100=1, the output status of Y10 is 1; otherwise it is 0.

7-22 Motion Control Instructions

7-22-1 Running motion process (MFFlowStart)

FUN 176 MFFlowStart	Start motion process	FUN 176 MFFlowStart
Symbol	If different axes should be activated at the same time, do not use the ID repeatedly.	



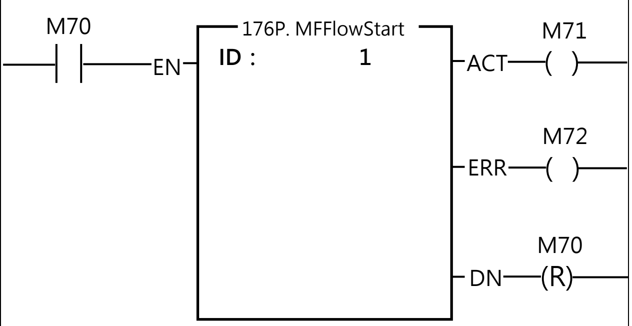
ID: The ID number of the motion process to be started.
EN: = 1. Means the motion process defined by the command will be started.
ACT: = 1. Means the system is running the defined motion process.
ERR: = 1. Means error is found in the motion process.
DN: =1. Means the motion process is completed.

Relay and Register

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WM9104	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R35023	R35024 | R35279	R35280 | R43223	R43224 | R47319	D0 | D11999		V, Z P0 ~ P9
ID	○	○	○	○	○	○	○	○	○	○	○	○	1~16	○

Example

Ladder diagram	ST

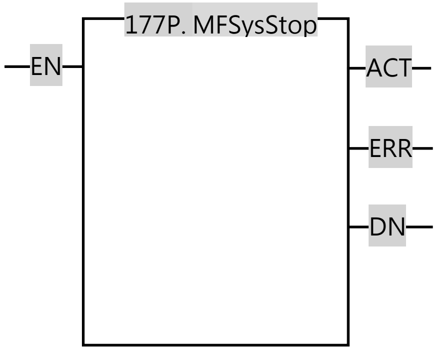
	`MFFlowStart ( EN:= M70, ID:= 1, ACT=> M71, ERR=> M72, DN=> M73);`
	`IF M73 Then`
	`M70 := FALSE;`
	`END_IF`

- When the execution control "EN" = 1, the motion flow corresponding to the UID will be executed.
- If the ID does not correspond to the motion process, ERR = 1 will be triggered.

7-22-2 Stop all motion processes (MFSysStop)

FUN 177 MFSysStop	Stop all motion processes	FUN 177 MFSysStop

Symbol

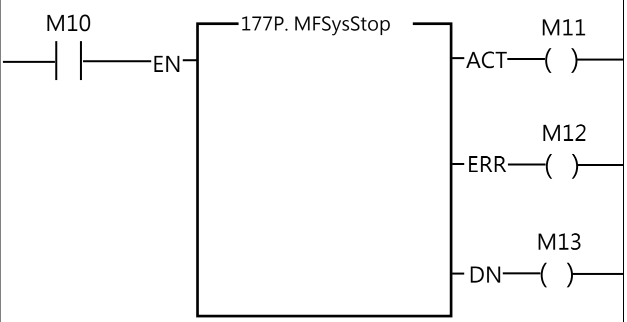


No operands

Description

- Interrupt all motion processes and stop EtherCAT communication. If you want to restart the process, you need to start the EtherCAT communication in MFSysInit.
- EN = 1: Interrupt all motion processes
- EN = 1: Motion control system emergency emergency stop
- ACT = 1: The system is in emergency stop action
- ERR = 1: system emergency stop error
- DN = 1: The system has completed emergency stop

Example

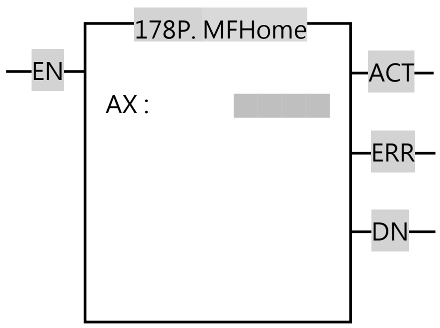
Ladder diagram	ST
	`MFSysStop ( EN:= M10, ACT=> M11, ERR=> M12, DN=> M13);`

- When the execution control "EN" = 1, the motion control in execution will be stopped in an emergency.
- If you want to restart the operation after execution, you need to perform initialization and start.

7-22-3 Home re-set (MFHome)

FUN 178 MFHome	Home re-set (MFHome)	FUN 178 MFHome

Symbol



AX: Means the axis where the Home re-setting will be executed.

Relay and Register

Type	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
Range	WX0 – WX1008	WY0 – WY1008	WM0 – WM9104	WS0 – WS3088	T0 – T1023	C0 – C1279	R0 – R34767	R34768 – R35023	R35024 – R35279	R35280 – R43223	R43224 – R47319	D0 – D11999		V, Z P0 ~ P9
AX							○	○	○	○	○	○	1~16	○

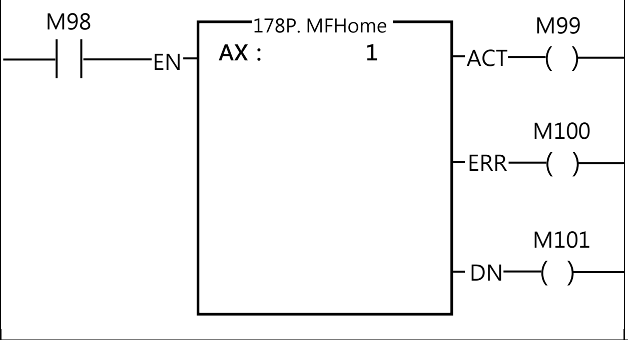
Description

Specify the motion axis to perform homing.

- EN = 1: trigger homing
- ACT = 1: Return-to-origin is in progress
- ERR = 1: Return-to-origin action error
- DN = 1: Return-to-origin is completed
- AX: Axis to execute

Special register
Axis 1: In return-to-origin operation M10621
Axis 1: Return to origin completed M10622

For the modes and details of the HOME command, please refer to Chapter 10.

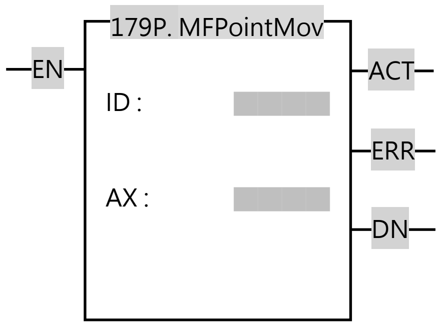
Ladder diagram	ST
	`MFHome ( EN:= M98, AX:= 1, ACT=> M99, ERR=> M100, DN=> M101);`

- When the execution control "EN" = 1, the origin return will be performed according to the parameters on the motion axis setting page.

7-22-4 Position Control (MFPointMov)

Fun179P MFPointMov	Position Control (MFPointMov)	Fun179P MFPointMov

Symbol



PT: Command number of motion point table
AX: Motion control axis number

Relay and Register

Type	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
Range	WX0 ｜ WX1008	WY0 ｜ WY1008	WM0 ｜ WM9104	WS0 ｜ WS3088	T0 ｜ T1023	C0 ｜ C1279	R0 ｜ R34767	R34768 ｜ R35023	R35024 ｜ R35279	R35280 ｜ R43223	R43224 ｜ R47319	D0 ｜ D11999		V, Z P0 ~ P9
ID	○	○	○	○	○	○	○	○	○	○	○	○	1~256	○
AX	○	○	○	○	○	○	○	○	○	○	○	○	1~16	○

Description

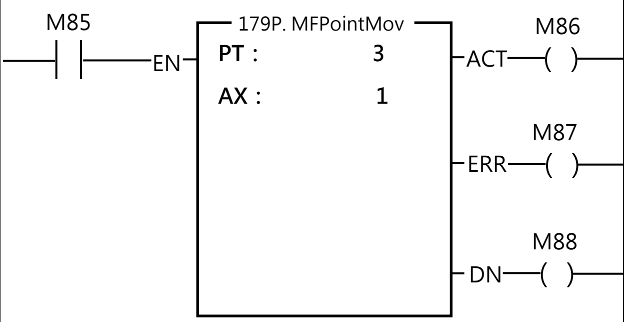
Execute the point table position control instruction.

- EN = 1: trigger position control
- ACT = 1: position control action
- ERR = 1: position control error
- DN = 1: The position control action is completed
- PT: Select the point of the movement point parameter
- AX: Axis to execute

Special registers:

- Axis 1: Position control action M10623
- Axis 1: Position control action completed M10624

Fun179P MFPointMov	Position Control (MFPointMov)	Fun179P MFPointMov
Example		

<table>
<tr><th>Ladder diagram</th><th>ST</th></tr>
<tr><td>


</td><td>

```
MFPointMov ( EN:= M85, PT:= 3, AX:= 1,
ACT=> M86, ERR=> M87, DN=> M88);
```

</td></tr>
</table>

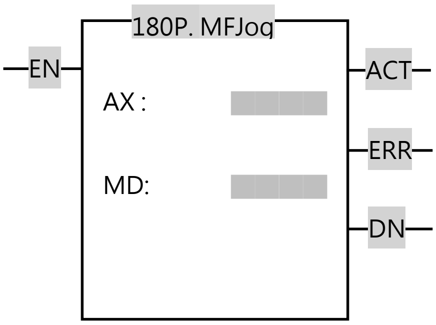
- When the execution control "EN" = 1, the axis specified by AX will execute the point table with the number specified by PT.
- When the execution control "EN" = 0, the movement will stop immediately.
- The following table is used as an example. When PT = 1 and AX = 1, axis 1 will run according to the parameters in point table 1;
  However, if PT=2 and AX=1 was set, it will fail due to the difference from the point table setting, and ERR will be triggered.

	Axis
1	M : Axis_1
2	M : Axis_2
3	M : Axis_1

7-22-5 JOG (MFJog)

Fun 180 MFJog	JOG (MFJog)	Fun 180 MFJog

Symbol



AX: Means the axis where JOG action will be executed.
MD: Mode of execution

Relay and Register

Type	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
Range	WX0 | WX1008	WY0 | WY1008	WM0 | WM9104	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R35023	R35024 | R35279	R35280 | R43223	R43224 | R47319	D0 | D11999		V, Z P0 ~ P9
AX	○	○	○	○	○	○	○	○	○	○	○	○	1~16	○
MD	○	○	○	○	○	○	○	○	○	○	○	○	0~3	○

Description

According to the JOG parameter and mode setting, the specified motion axis executes the JOG function.

- EN = 1: trigger manual control
- D/R = 1 forward / = 0 reverse
- ACT = 1: JOG action
- ERR = 1: JOG error
- DN = 1: JOG action completed
- AX: Axis to execute
- MD: mode 0~mode 3

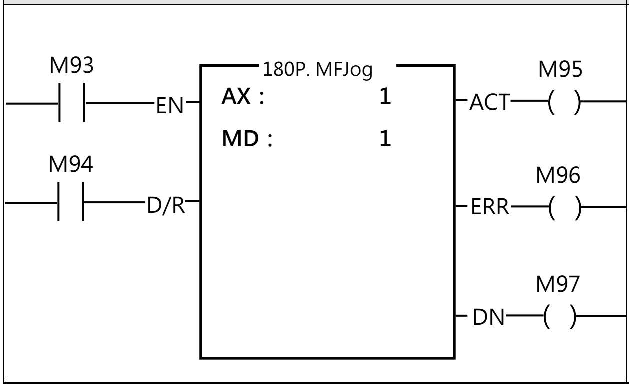
  Mode 0: Continue to advance at the JOG start speed.
  Mode 1: Advance at JOG start speed, advance the jogging distance and then stop.
  Mode 2: Start at the JOG start speed, accelerate to the JOG speed with the JOG acceleration and continue moving forward.
  Mode 3: Start at the JOG start speed, accelerate to the JOG speed with the JOG acceleration, and stop after moving forward.

Special registers

- Axis 1: JOG action M10625
- Axis 1: JOG completed M10626

Please refer to Chapter 11 for JOG instruction modes and details.

Example

Ladder diagram	ST
	MFJog (EN:= M93, D_R:= M94, AX:= 1, MD:= 1, ACT=> M95, ERR=> M96, DN=> M97);

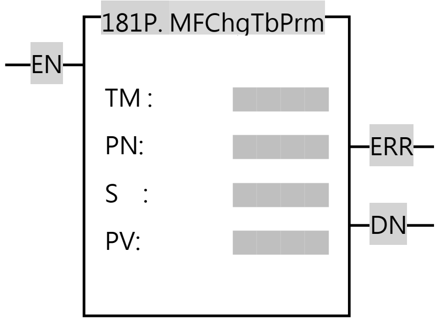
- When the execution control "EN" = 1, the axis specified by AX will execute the mode specified by MD.
- When the execution control "EN" = 0, the movement will stop immediately.
- Take the following table as an example. When AX = 1 and MD = 1, it means axis 1 will run a distance of 100mm at a speed of 1mm/s.

	Axis 1
JOG Start Speed	1mm/s
JOG Speed	10mm/s
JOG Acceleration	1000mm/s²
JOG Deceleration	1000mm/s²
Jog Distance	100mm

7-22-6 Change block parameters (MFChgTbPrm)

FUN181 MFChgTbPrm	Change block parameters (MFChgTbPrm)	FUN181 MFChgTbPrm

Symbol



TM: Flow Block Table
PN: The number of blocks
S: Item Number
PV: Written value

Relay and Register

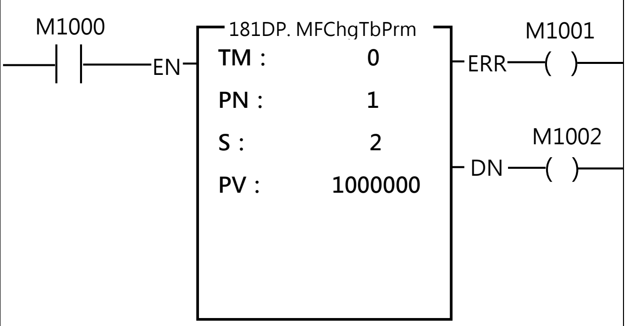
Operand \ Range	HR	IR	OR	SR	ROR	DR	K	XR
	R0 ~ R34767	R34768 ~ R35023	R35024 ~ R35279	R35280 ~ R43223	R43224 ~ R47319	D0 ~ D11999		V, Z P0 ~ P9
TM							0~128	
PN							1~4096	
S	○	○	○	○	○	○	0~50	○
PV	○	○	○	○	○	○	0~2147483647	○

Description

- [Fun181 Change Motion Control Parameters] is used to change a single or a few motion control parameters. If you need to read or write a large number of motion control parameters, you can use [Fun188 Recipe Read] and [Fun189 Recipe Write].
- Operands
  TM table number: 0 point table, 1 axis table, 2 synchronization table, 128 flow table
  PN point number: Correspond to different types of numbers according to the table to be modified by TM, point table number, axis number, process block number
  S item number: please refer to the table below
  PV write value: the value to be written, fixed Double Word.
- When the execution control [EN] is triggered by the upper differential, Fun181 will write the PV value into the specified motion control parameter

- When the execution control [EN] is triggered by the lower differential, all output indications are reset.
- When writing motion control parameters, if there is an error, the output indication [ERR] will be ON.
- When the writing of motion control parameters is completed, the output indication【DN】ON.

Example

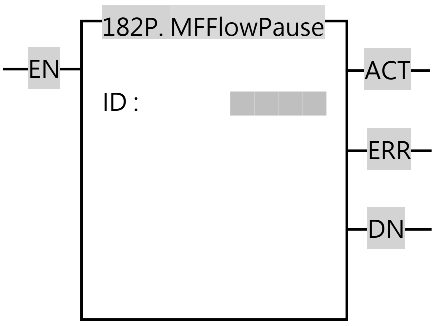
Ladder diagram	ST
	`MFChgTbPrm ( EN:= M1000, TM:= 0, PN:= 1, S:= 2, PV:= 1000000, ERR=> M1001, DN=> M1002);`

- When M1000 OFF→ON, change the point table parameters (TM: 0 point table, PN: 1 point table 1, S: 2 spindle coordinates, PV: change to 1000.000mm), and the spindle movement distance of point table 1 is changed to 1000.000 mm.

7-22-7 Pause Motion Flow (MFFlowPause)

FUN 182 MFFlowPause	Pause Motion Flow (MFFlowPause)	FUN 182 MFFlowPause

Symbol



ID: The motion process to be paused.

Relay and Register

Type	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
Range	WX0 | WX1008	WY0 | WY1008	WM0 | WM9104	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R35023	R35024 | R35279	R35280 | R43223	R43224 | R47319	D0 | D11999		V, Z P0 ~ P9
ID	○	○	○	○	○	○	○	○	○	○	○	○	-32767~32767	○

Description

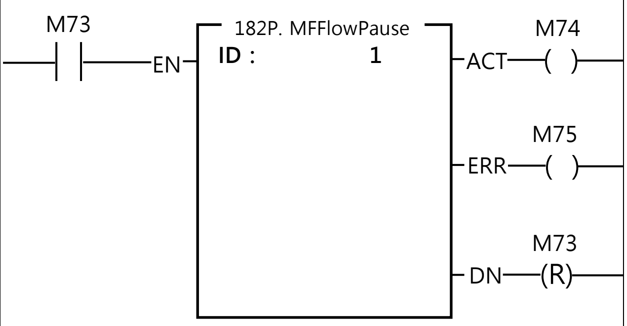
Pause the motion process of the specified ID, and stop after executing the current process block, To resume a paused motion process, you can use Fun183 MFFlowResume to resume execution.

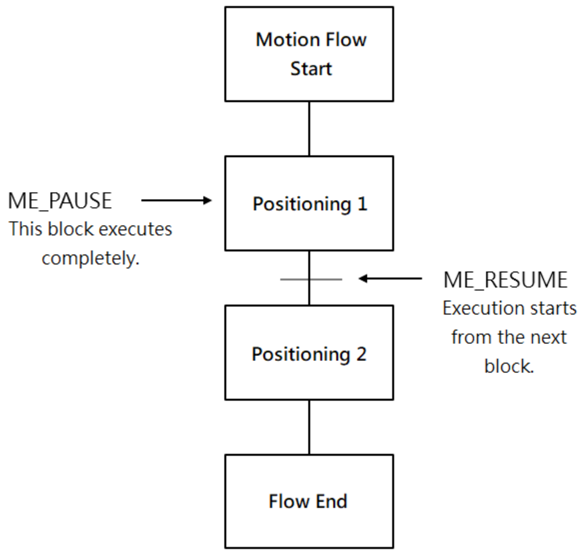
- EN = 1: Stop entering the next step after executing the current process block
- ACT = 1: pause action
- ERR = 1: timeout error
- DN = 1: Pause completed
- ID: UID of the motion process to be paused

FUN 182 MFFlowPause	Pause Motion Cycle (MFFlowPause)	FUN 182 MFFlowPause

Example

Ladder diagram	ST


 	
	`MFFlowPause ( EN:= M73, ID:= 1, ACT=> M74, ERR=> M75, DN=> M76);`
	`IF M76 Then`
	`M73 := FALSE;`
	`END_IF`

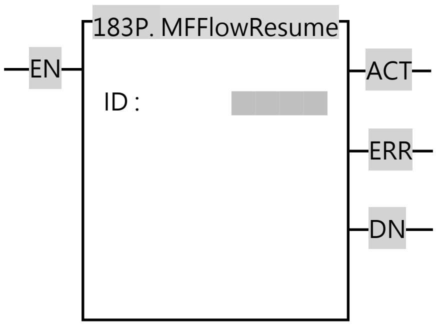


- When the execution control "EN" = 1, it will pause and not execute the next step after executing the current motion flow block.

7-22-8 Resume Motion Process (MFFlowResume)

FUN 183 MFFlowResume	Resume Motion Process (MFFlowResume)	FUN 183 MFFlowResume

Symbol



ID: Means the motion process to be resumed.

Relay and Register

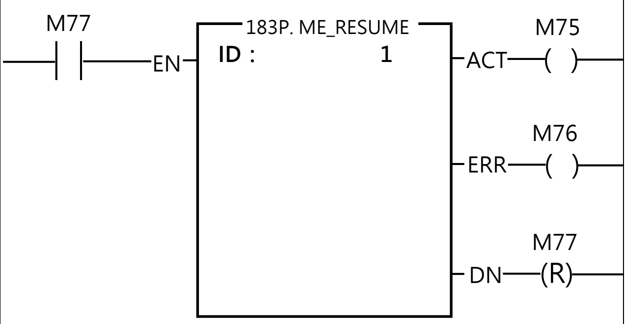
Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ~ WX1008	WY0 ~ WY1008	WM0 ~ WM9104	WS0 ~ WS3088	T0 ~ T1023	C0 ~ C1279	R0 ~ R34767	R34768 ~ R35023	R35024 ~ R35279	R35280 ~ R43223	R43224 ~ R47319	D0 ~ D11999		V, Z P0 ~ P9
ID	○	○	○	○	○	○	○	○	○	○	○	○	-32767~32767	○

Description

Resume the paused or interrupted motion process and continue execution.

- EN = 1: resume motion flow
- ACT = 1: resume motion flow in action
- ERR = 1: Resume movement flow error
- DN = 1: The motion flow resume is completed

Example

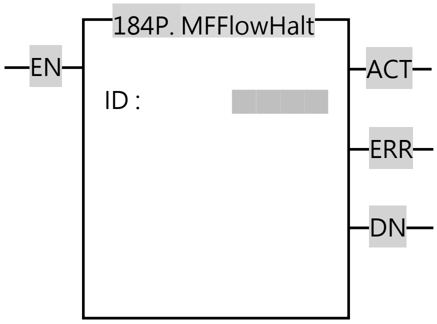
Ladder diagram	ST
	ME_RESUME(EN:= M77, ID:= 1, ACT=> M75, ERR=> M76, DN=> M78); IF M78 Then M77 := FALSE; END_IF

- When the execution control "EN" = 1, the motion process suspended due to the execution of Fun182 (MFFlowPause) or Fun184 (MFFlowHalt) will be resumed.

7-22-9 Motion Process Halt (MFFlowHalt)

FUN 184 MFFlowHalt	Motion Process Halt (MFFlowHalt)	FUN 184 MFFlowHalt

Symbol



ID: Means the motion process to be suspended.

Relay and Register

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ｜ WX1008	WY0 ｜ WY1008	WM0 ｜ WM9104	WS0 ｜ WS3088	T0 ｜ T1023	C0 ｜ C1279	R0 ｜ R34767	R34768 ｜ R35023	R35024 ｜ R35279	R35280 ｜ R43223	R43224 ｜ R47319	D0 ｜ D11999		V, Z P0 ~ P9
ID	○	○	○	○	○	○	○	○	○	○	○	○	-32767~32767	○

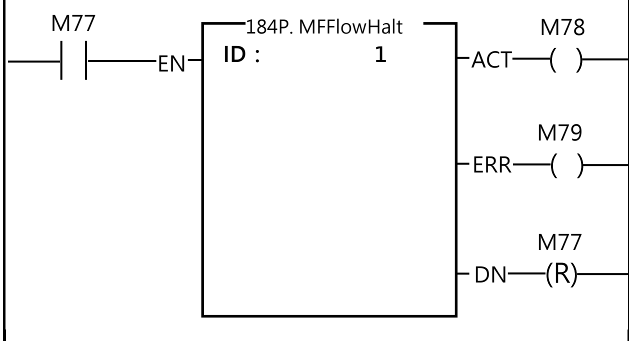
Description

Immediately stops the currently executing process block,
If you want to continue the stopped motion process, you can use Fun183 MFFlowResume to resume execution.

- EN = 1: Halt motion process
- ACT = 1: Halt action
- ERR = 1: Halt error
- DN = 1: Halt complete
- ID: UID of the motion process to be interrupted

FUN 184 MFFlowHalt	Suspend Motion Process (MFFlowHalt)	FUN 184 MFFlowHalt
Example		

Ladder diagram	ST

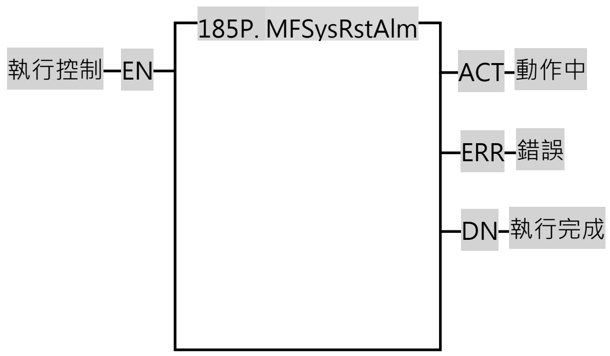
	`MFFlowHalt ( EN:= M77, ID:= 1, ACT=> M78, ERR=> M79, DN=> M80);`
	`IF M80 Then`
	`M77 := FALSE;`
	`END_IF`

- When the execution control "EN" = 1, the running motion flow block will be suspended immediately.

7-22-10 Reset Motion Alarm (MFSysRstAlm)

FUN185 MFSysRstAlm	Reset Motion Alarm (MFSysRstAlm)	FUN185 MFSysRstAlm

Symbol

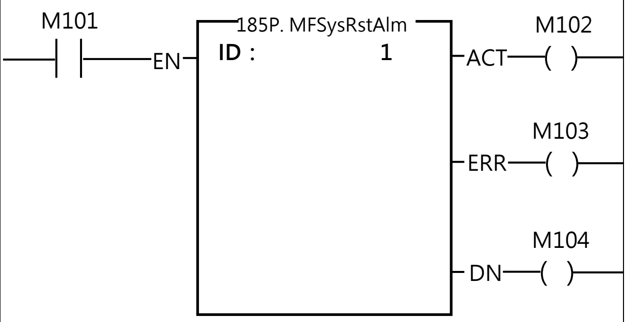


Description

Clears all motion sequences and driver error alerts; however, the communication alarm of the drive cannot be cleared by this command and needs to be powered on again.

- EN = 1: Upper edge trigger clears motion error alarm
- ACT = 1: Clear motion error alarm action
- ERR = 1: Clear motion error alarm error
- DN = 1: Clear motion error alarm completed

Example

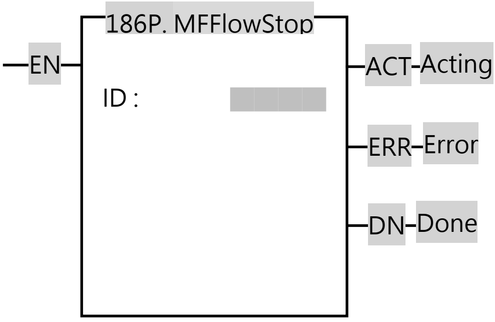
Ladder diagram	ST
	`MFSysRstAlm ( EN:= M101, ACT=> M102, ERR=> M103, DN=> M104);`

- When the execution control "EN" = 1, it will clear the motion process and errors occurred in the driver.

7-22-11 Motion Process Terminate (MFFlowStop)

FUN 186 MFFlowStop	Stop Motion Process (MFFlowStop)	FUN 186 MFFlowStop
Symbol		

Ladder



ID: ID number for the motion process to be finished.

Relay and Register

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 – WX1008	WY0 – WY1008	WM0 – WM9104	WS0 – WS3088	T0 – T1023	C0 – C1279	R0 – R34767	R34768 – R35023	R35024 – R35279	R35280 – R43223	R43224 – R47319	D0 – D11999		V, Z P0 ~ P9
ID	○	○	○	○	○	○	○	○	○	○	○	○	-32767~32767	○

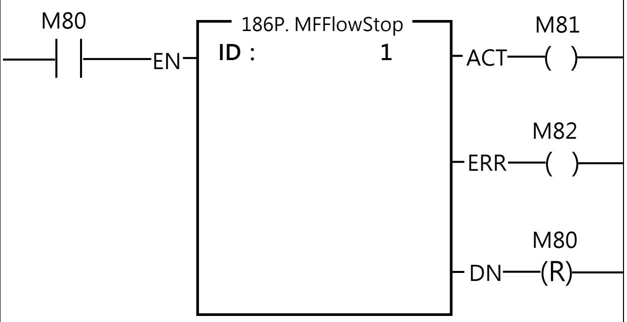
Description	

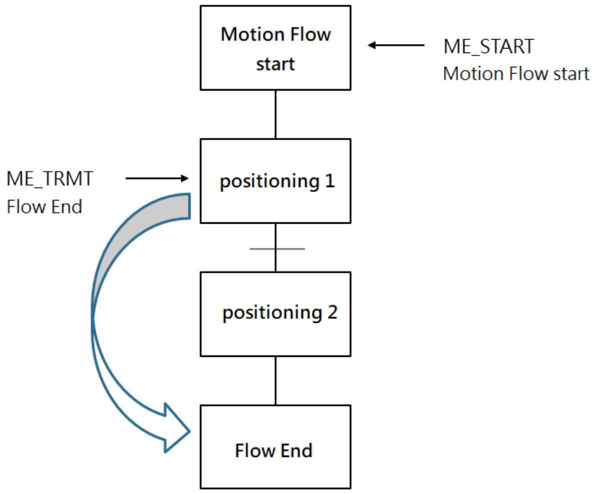
Immediately end the motion process of the specified ID.
When execution of this instruction is complete, MFFlowResume cannot be used to resume execution. Need to use MFFlowStart to restart the process.

- EN = 1: The upper edge triggers the motion process to stop
- ACT = 1: The stop of the motion process is in motion
- ERR = 1: Motion process stop error
- DN = 1: The motion process stop is completed

Example	

Ladder diagram	ST


 	
	`MFFlowStop ( EN:= M80, ID:= 1, ACT=> M81, ERR=> M82, DN=> M83);`
	`IF M83 Then`
	`M80 := FALSE;`
	`END_IF`

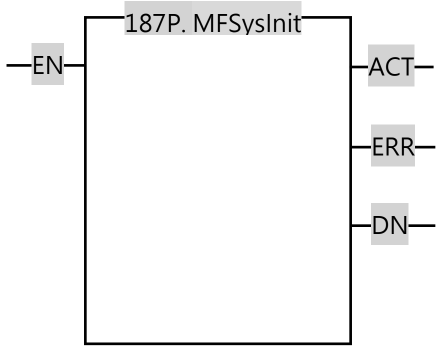


- When the execution control "EN" = 1, the motion process of the specified ID will stop immediately.

7-22-12 Servo Initialization (MFSysInit)

FUN187 MFSysInit	Servo Initialization	FUN187 MFSysInit

Symbol



No operands

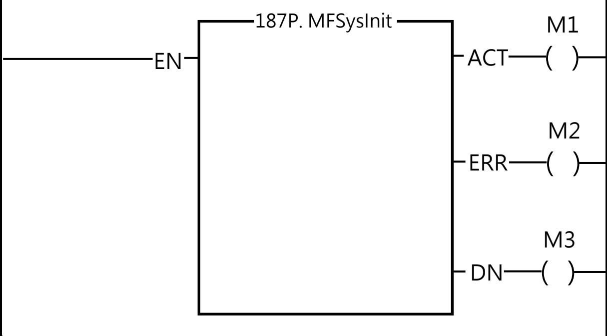
Relay and Register

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ~ WX1008	WY0 ~ WY1008	WM0 ~ WM9104	WS0 ~ WS3088	T0 ~ T1023	C0 ~ C1279	R0 ~ R34767	R34768 ~ R35023	R35024 ~ R35279	R35280 ~ R43223	R43224 ~ R47319	D0 ~ D11999		V, Z P0 ~ P9
ID	○	○	○	○	○	○	○	○	○	○	○	○	1~16	○

Description

- If you want to control the servo through EtherCAT communication, you must execute this command before executing any motion control.
- If you want to use Fun 235 to convert the physical axis to the imaginary axis, it must be executed before this command.
- EN = 1: Start motion control initialization (trigger condition supports up and down differential input)
- ACT = 1: Motion control initialization action
- ERR = 1: Motion control initialization error
- DN = 1: Motion control initialization is complete

Example

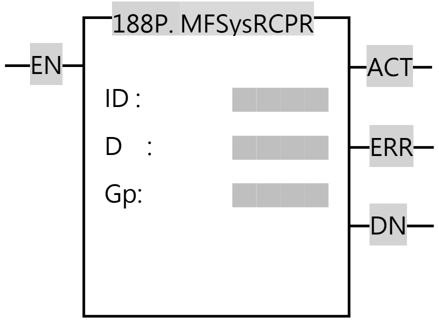
Ladder diagram	ST
	`MFSysInit ( EN:= TRUE, ACT=> M1, ERR=> M2, DN=> M3);`

- When the execution control "EN" = 1, the motion control function initialization action will be executed.
- If there is no response during execution, please confirm whether the sports link setting is consistent with the actual link.
- After initialization, the servo needs to be turned on to continue subsequent operations, such as all axes enable (Servo on) register (M10520).

7-22-13 Recipe Reading (MFSysRCPR)

FUN188 MFSysRCPR	Recipe Reading (MFSysRCPR)	FUN188 MFSysRCPR

Symbol



Md: 0, read parameters from PLC
D: The starting position of the data register for moving into the mapping table
GP: Recipe tables number, starting from 1

Relay and Register

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WM9104	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R35023	R35024 | R35279	R35280 | R43223	R43224 | R47319	D0 | D11999		V, Z P0 ~ P9
ID	○	○	○	○	○	○	○	○	○	○	○	○	0~1	
D	○	○	○	○	○	○	○	○	○	○	○	○		○
Gp	○	○	○	○	○	○	○	○	○	○	○	○	0~100	

Description

- [Fun188 Recipe Read] and [Fun189 Recipe Write] are used to read or write a large number of motion control parameters. If you only need to modify a single or a few parameters, you can use [Fun181 Change Motion Control Parameters] or [Fun198 Mapping Table].
- Parameters can only be read when the axis stops.
- Operands
  Md mode: 0 use PLC register
  D formula starting register: the initial address of the register to be stored after reading the formula table
  Gp reads the column of the recipe table: reads the column of the recipe table, 0 reads all
- When the execution control [EN] is triggered by the upper differential, Fun188 will read the specified recipe to the specified register.
  When the execution control [EN] is triggered by the lower differential, all output indications are reset.
- When the recipe is read, the output indication [ACT] is ON.
- When reading the recipe, if there is an error, the output indication [ERR] will be ON.
- When the reading of the recipe is completed, the output indication [DN] ON.

Recipe Table

【Project Management】>【Motion Control】>【Motion Recipe】

Main0 × Motion Recipe Table ×

	Table	Index	Length	Start Address	End Address
1	Position Table	1	1	R0	R49
2	Axis Table	1	1	R50	R119
3	Sync Table	1	1	R120	R269

- Motion Recipe table
  Tables: Point table, Axis table, Synchronization table
  Index: Point table (number of points), Axis table (number of axes), Synchronization table (number of axes)
  Length: Continuous point table or continuous axis
  Start address: The start address of the register for reading and writing recipes
- Please refer to the following table for the definition of the register value of the motion recipe table

Recipe Point Table

Start Addres+N	Item	Size	Type	L	Definition
R+0	Operation Mode	WORD	INT	1	0. Unuse 1. Single/ABS 2. Single/INC 3. Linear(2Axis)/ABS 4. Linear(2Axis)/INC 5. Linear(3Axis)/ABS 6. Linear(3Axis)/INC 7. Linear(4Axis)/ABS 8. Linear(4Axis)/INC 9. Arc/ABS 10. Arc/ INC 11. Arc 3D/ABS 12. Arc 3D/ INC 13. Helical/ABS 14. Helical/ INC 15. Single Velocity
R+1	Accerlation Profile	WORD	INT	1	0. T Curve 1. S Curve
R+2	Master Axis	WORD	INT	1	1~16 Non use = 0

R+3	Interpolation 1	WORD	INT	1	1~16 Non use = 0
R+4	Interpolation 2	WORD	INT	1	1~16 Non use = 0
R+5	Interpolation 3	WORD	INT	1	1~16 Non use = 0
R+6	Target Position Master Axis	DWORD	INT	2	Precision: Decimal Place (negative number allow)
R+8	Target Position Interpolation 1	DWORD	INT	2	Precision: Decimal Place (negative number allow)
R+10	Target Position Interpolation 2	DWORD	INT	2	Precision: Decimal Place (negative number allow)
R+12	Target Position Interpolation 3	DWORD	INT	2	Precision: Decimal Place (negative number allow)
R+14	Velocity	DWORD	INT	2	Precision: Decimal Place (positive number only)
R+16	Acceleration	DWORD	INT	2	Precision: Decimal Place (positive number only)
R+18	Deceleration	DWORD	INT	2	Precision: Decimal Place (positive number only)
R+20	Acceleration S Curve	WORD	INT	1	Precision: 0.1
R+21	Deceleration S Curve	WORD	INT	1	Precision: 0.1
R+22	Arc Mode	WORD	INT	1	0. Border Point 1. Center 2. Radius
R+23	Arc Direction	WORD	INT	1	0. CW 1. CCW
R+24	Arc (Border/Center) X coordinate	DWORD	INT	2	Precision: Decimal Place (negative number allow)
R+26	Arc (Border/Center) Y coordinate	DWORD	INT	2	Precision: Decimal Place (negative number allow)
R+28	Arc Radius	DWORD	INT	2	Precision: Decimal Place (positive number only)

R+30	Aux Radius	DWORD	INT	2	Precision: Decimal Place (positive number only)
R+32	Standby Time	DWORD	UINT	2	Unit: ms
R+34	Continuous Point	WORD	INT	1	1~1024 End = 0
R+35	Circle Revolution	WORD	UINT	1	0~65535
R+36	Continuous Mode	WORD	INT	1	0. Standby 1. Next Point Speed Continue 2. Current Point Speed Continue 3. Starting Speed Continue
R+37-41	Reserve			5	
R+42	Arc (Border/Center) Z coordinate	DWORD	INT	2	Precision: Decimal Place (negative number allow)

Recipe Axis Table

Start Addres+N	Item	Size	Type	L	Definition
R+0	Encoder Type	WORD		1	0 = Incremental 1 = Absolute
R+1	Unit	WORD		1	0. PLS 1. Mm 2. Deg 3. inch
R+2	Decimal Point	WORD		1	1000: 1 100: 0.1 10: 0.01 1: 0.001
R+3	Pulse/Revolution	DWORD		2	Precision: Decimal Place
R+5	Unit/Revolution	DWORD		2	Precision: Decimal Place
R+7	Velocity Unit	DWORD		1	0. PLS/Sec 1. PLS/min 2. RPM
R+8	Velocity Gain	DWORD		2	Precision: 0.001
R+10	Start Velocity	DWORD		2	Precision: Decimal Place
R+12	Max Motor Velocity	DWORD		2	Precision: 1 Unit: RPM
R+14	Default Acceleration	DWORD		2	Precision: Decimal Place
R+16	Default Deceleration	DWORD		2	Precision: Decimal Place
R+18	Soft Limit(+)	DWORD		2	Precision: Decimal Place (positive number only)
R+20	Soft Limit(-)	DWORD		2	Precision: Decimal Place (positive number only)
R+22	Following Error Window	DWORD		2	Precision: Decimal Place
R+24	Following Error Timeout	DWORD		2	Unit: ms
R+26	Pos Done Tolerance	DWORD		2	Precision: Decimal Place

R+28	Pos Done Check Time	DWORD		2	Unit: ms
R+30	Maximum Motor Torque	WORD		1	Precision: 0.1
R+31	Maximum Torque Limit(+)	WORD		1	Precision: 0.1
R+32	Maximum Torque Limit(-)	WORD		1	Precision: 0.1
R+33	Touch Probe1 Source	WORD		1	0. Disable 1. Input 2. Z Signal
R+34	Touch Probe1 Mode	WORD		1	0. Rising Edge Single 1. Rising Edge Continue 2. Falling Edge Single 3. Falling Edge Continue
R+35	Touch Probe2 Source	WORD		1	0. Disable 1. Input 2. Z Signal
R+36	Touch Probe2 Mode	WORD		1	0. Rising Edge Single 1. Rising Edge Continue 2. Falling Edge Single 3. Falling Edge Continue
R+37-40	Reserve			4	
R+41	Stop Mode	WORD		1	5. Deceleration Stop 7. Immediately Stop
R+42	Stop Deceleration	DWORD		2	Precision: Decimal Place
R+44	Homing Mode	WORD		1	99. Homing on current position 100. Forward-Falling Trigger 101. Backward-Falling Trigger 102. Z Signal-Forward-Rising Trigger 103. Z Signal-Forward-Falling Trigger 104. Forward- Rising Trigger 105. Backward-Rising Trigger 106. Z Signal-Backward-Rising Trigger

					107. Z Signal-Backward-Falling Trigger
R+45	Homing IO Source	WORD		1	0. From Servo Driver 1. From PLC
R+46	Homing Start Direction	WORD		1	0. Negative 1. Positive
R+47	Homing Origin Offset	DWORD		2	Precision: Decimal Place (negative number allow)
R+49	Homing Find Velocity	DWORD		2	Precision: Decimal Place
R+51	Homing Creep Velocity	DWORD		2	Precision: Decimal Place
R+53	Homing Deceleration	DWORD		2	Precision: Decimal Place
R+55	Limit Switch(-)(DI)	WORD		1	
R+56	Limit Switch(+)(DI)	WORD		1	
R+57	Homing Switch(DI)	WORD		1	
R+58	Homing Z Count	DWORD		2	
R+60	Jogging Base Velocity	DWORD		2	Precision: Decimal Place
R+62	Jogging Velocity	DWORD		2	Precision: Decimal Place
R+64	Jogging Acceleration	DWORD		2	Precision: Decimal Place
R+66	Jogging Deceleration	DWORD		2	Precision: Decimal Place
R+68	Inching Distance	DWORD		2	Precision: Decimal Place

Recipe Synchronous Table

Start Addres+N	Item	Size	Type	L	Definition
R+0	Input axis coordinate Unit	WORD		1	
R+1	Input axis coordinate decimal point	WORD		1	
R+2	Input axis period	DWORD		2	Precision: Decimal Place
R+4	Clutch OFF sliding time at deceleration stop	DWORD		2	
R+6	Input axis phase init method	WORD		1	
R+7	Sync master axis phase default value	DWORD		2	Precision: Decimal Place
R+9	Master axis phase default value after phase compensation	DWORD		2	Precision: Decimal Place
R+11	Main clutch input axis phase default value	DWORD		2	Precision: Decimal Place
R+13	Auxiliary clutch input axis phase default value	DWORD		2	Precision: Decimal Place
R+15	Cam input axis/clutch output axis	WORD		1	

	phase init method				
R+16	Main clutch output axis phase default value	DWORD		2	Precision: Decimal Place
R+18	Auxiliary clutch output axis phase default value	DWORD		2	Precision: Decimal Place
R+20	Reserve	DWORD		2	
R+22	Cam input axis phase default value	DWORD		2	Precision: Decimal Place
R+24	Cam output axis base coordinate	DWORD		2	Precision: Decimal Place
R+26	Master Axis 1 input selection	WORD		1	
R+27	Master Axis 1 external reference number	WORD		1	
R+28	Master Axis 1 prevent reverse	WORD		1	
R+29	Master Axis 1 coordinate transformation setting	WORD		1	
R+30	Master Axis 1 coordinate transformation numerator	DWORD		2	
R+32	Master Axis 1 coordinate transformation denominator	DWORD		2	

R+34	Master Axis 2 input selection	WORD		1	
R+35	Master Axis 2 external reference number	WORD		1	
R+36	Master Axis 2 prevent reverse	WORD		1	
R+37	Master Axis 2 coordinate transformation setting	WORD		1	
R+38	Master Axis 2 coordinate transformation numerator	DWORD		2	
R+40	Master Axis 2 coordinate transformation denominator	DWORD		2	
R+42	Aux Axis input selection	WORD		1	
R+43	Aux Axis external reference number	WORD		1	
R+44	Aux Axis prevent reverse	WORD		1	
R+45	Aux Axis coordinate transformation setting	WORD		1	
R+46	Aux Axis coordinate transformation numerator	DWORD		2	

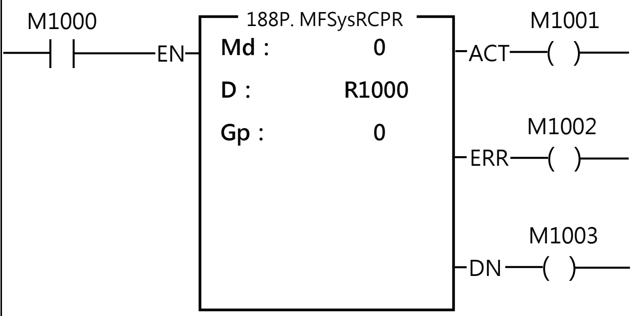
R+48	Aux Axis coordinate transformation denominator	DWORD		2	
R+50	Master Axis compensation command value	DWORD		2	Precision: Decimal Place
R+52	Master Axis compensation change mode	WORD		1	
R+53	Master Axis compensation change time	DWORD		2	
R+55	Aux Axis compensation command value	DWORD		2	Precision: Decimal Place
R+57	Aux Axis compensation change mode	WORD		1	
R+58	Aux Axis compensation change time	DWORD		2	
R+60	Variable gear retio numerator	DWORD		2	
R+62	Variable gear retio denominator	DWORD		2	
R+64	Gear retio change mode	WORD		1	
R+65	Variable gear retio change time	DWORD		2	
R+67	Main clutch ON condition	WORD		1	

R+68	Main clutch ON setting value	DWORD		2	Precision: Decimal Place
R+70	Main clutch ON delay	DWORD		2	Precision: Decimal Place
R+72	Reserve	WORD		1	
R+73	Main clutch ON connection method	WORD		1	
R+74	Reserve	WORD		1	
R+75	Main clutch ON sliding curve	WORD		1	
R+76	Reserve	DWORD		2	
R+78	Main clutch ON sliding time	DWORD		2	
R+80	Main clutch ON following time	DWORD		2	
R+82	Main clutch ON follow-ups	DWORD		2	Precision: Decimal Place
R+84	Main clutch OFF condition	WORD		1	
R+85	Main clutch OFF setting value	DWORD		2	Precision: Decimal Place
R+87	Main clutch OFF delay	DWORD		2	Precision: Decimal Place
R+87	Reserve	WORD		1	
R+90	Main clutch OFF connection method	WORD		1	
R+91	Reserve	WORD		1	
R+92	Main clutch OFF sliding curve	WORD		1	
R+93	Reserve	DWORD		2	
R+95	Main clutch OFF sliding time	DWORD		2	

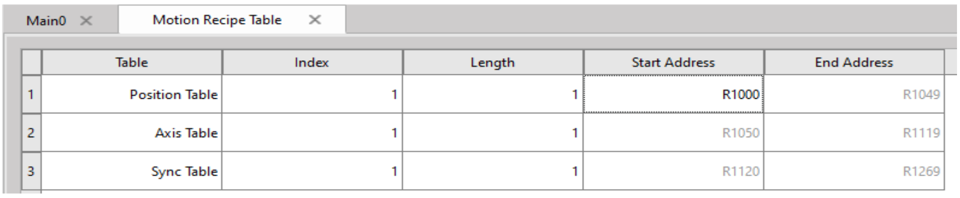
R+97	Aux clutch ON condition	WORD		1	
R+98	Aux clutch ON setting value	DWORD		2	Precision: Decimal Place
R+100	Aux clutch ON delay	DWORD		2	Precision: Decimal Place
R+102	Reserve	WORD		1	
R+103	Aux clutch ON connection method	WORD		1	
R+104	Reserve	WORD		1	
R+105	Aux clutch ON sliding curve	WORD		1	
R+106	Reserve	DWORD		2	
R+108	Aux clutch ON sliding time	DWORD		2	
R+110	Aux clutch ON following time	DWORD		2	
R+112	Aux clutch ON follow-ups	DWORD		2	Precision: Decimal Place
R+114	Aux clutch OFF condition	WORD		1	
R+115	Aux clutch OFF setting value	DWORD		2	Precision: Decimal Place
R+117	Aux clutch OFF delay	DWORD		2	Precision: Decimal Place
R+119	Reserve	WORD		1	
R+120	Aux clutch OFF connection method	WORD		1	
R+121	Reserve	WORD		1	
R+122	Aux clutch OFF sliding curve	WORD		1	
R+123	Reserve	DWORD		2	
R+125	Aux clutch OFF sliding time	DWORD		2	
R+127	Reserve	WORD*5		5	

R+132	Step Angle Compensation Base speed	DWORD		2	Precision: Decimal Place
R+134	Step Angle Compensation Base value	DWORD		2	Precision: Decimal Place
R+136	Step Angle Compensation value change mode	WORD		1	
R+137	Step Angle Compensation value change time	DWORD		2	
R+139	Cam data No.	WORD		1	
R+140	Cam stroke	DWORD		2	Precision: Decimal Place
R+142	Cam contact output No.	WORD		1	
R+143	Output filter time constant	DWORD		2	
R+145-149	Reserve				

Example

Ladder diagram	ST
	`MFSysRCPR ( EN:= M1000, Md:= 0, D:= R1000, Gp:= 0, ACT=> M1001, ERR=> M1002, DN=> M1003);`

Motion Recipe Table



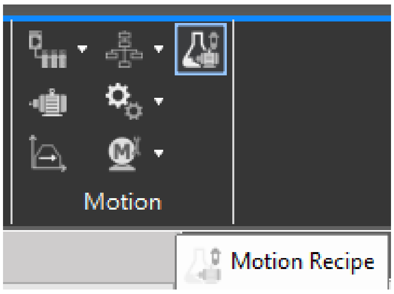
Main0 Motion Recipe Table

	Table	Index	Length	Start Address	End Address
1	Position Table	1	1	R1000	R1049
2	Axis Table	1	1	R1050	R1119
3	Sync Table	1	1	R1120	R1269

- When M1000 is from OFF→ON, read all recipe tables and store them in R1000.
- Read the parameters of PLC point table 1 and store them in R1000~R1049
- Read the parameters of the PLC axis table (axis 1) and store them in R1050~R1119
- Read the parameters of the PLC synchronization table (axis 1) and store them in R1120~R1269

FUN188 MFSysRCPR	Motion Control Recipe Writing (MFSysRCPR)	FUN188 MFSysRCPR
Example		

Find in “Project”-> “Motion recipe”



Take the target position of the point table as an example for mapping; the following figure is the setting page:

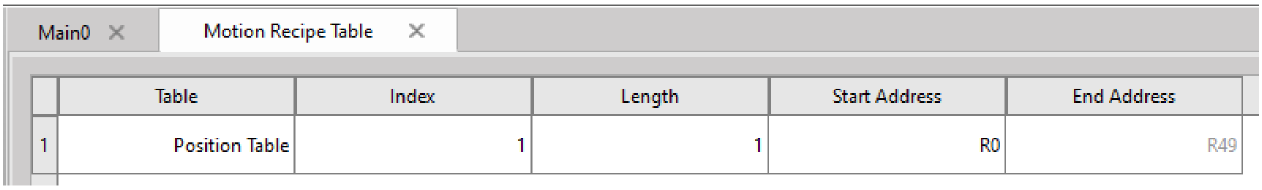
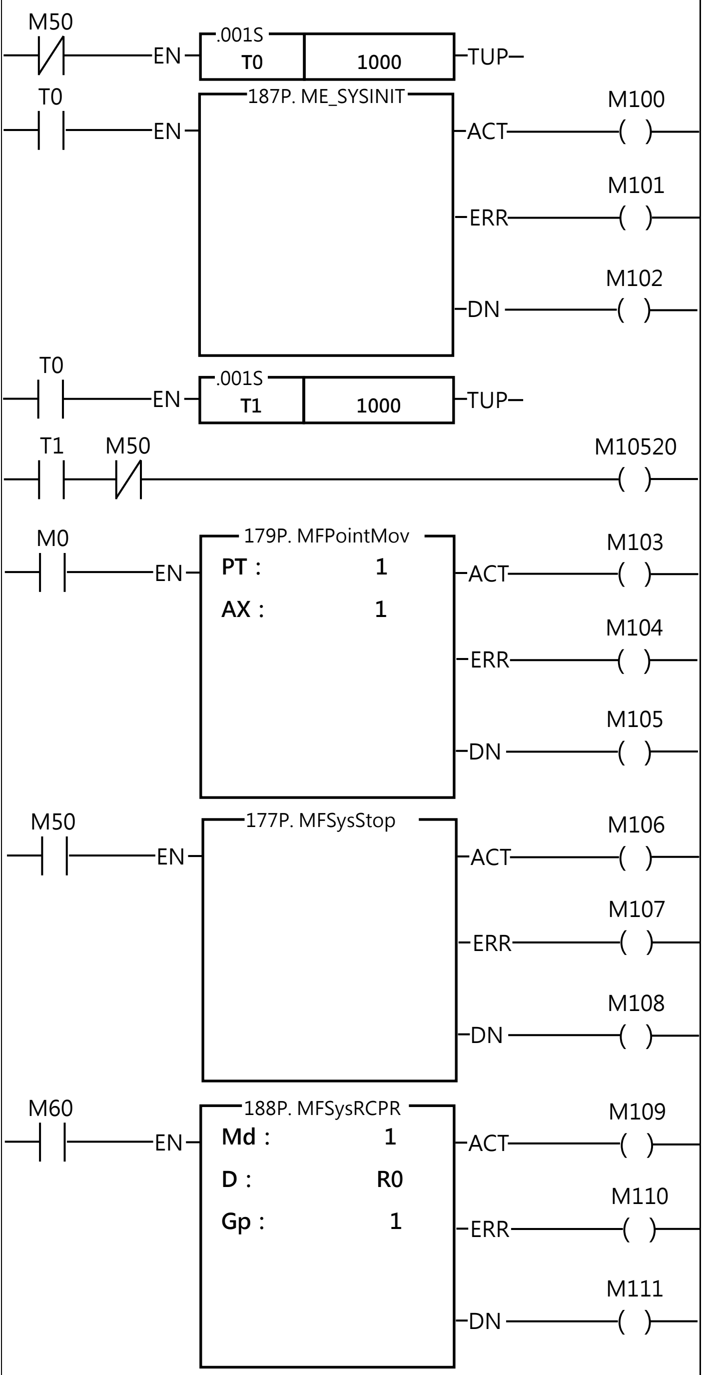


	Table	Index	Length	Start Address	End Address
1	Position Table	1	1	R0	R49

The following is a sample program using the point table as an example:

FUN188 MFSysRCPR	Motion Control Recipe Writing (BSysRCPR)	FUN188 BSysRCPR

Ladder diagram (sub)	ST
see below	see below

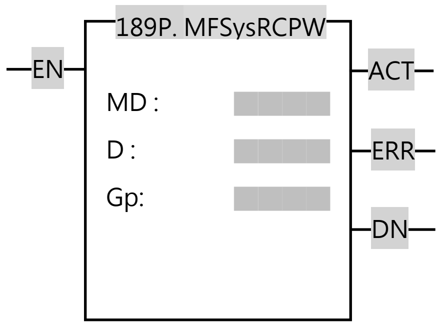



```
Timer( EN:= M50 = FALSE,
      T:= 0,
      PV:= 1000,
      IsTimeout=> T0);
ME_SYSINIT( EN:= T0, ACT=> M100,
ERR=> M101, DN=> M102);
Timer( EN:= T0,
      T:= 1,
      PV:= 1000,
      IsTimeout=> T1);
IF T1 AND M50 = FALSE Then
M10520 := TRUE;
END_IF
MFPointMov ( EN:= M0, PT:= 1,
AX:= 1, ACT=> M103, ERR=> M104,
DN=> M105);
MFSysStop ( EN:= M50, ACT=> M106,
ERR=> M107, DN=> M108);
MFSysRCPR ( EN:= M60, Md:= 1, D:=
R0, Gp:= 1, ACT=> M109, ERR=>
M110, DN=> M111);
```

In this example, the motion control system will be initialized one second after the first execution, and the other axes will be Servo On after one second. When M60 is on, all contents of the midpoint table will be moved into recipe 1. Because R0 is the start register to be moved into the point table, then turn M0 ON to execute the point table, and its setting will follow the position of DR6.
If you need to restore the recipe table before use, you can turn on the M50 and turn it off to restart the Motion control system.

7-22-14 Motion Control Recipe Writing (MFSysRCPW)

FUN189 MFSysRCPW	Motion Control Recipe Writing (MFSysRCPW)	FUN189 MFSysRCPW
Symbol		



Md: 0, write parameters to PLC
D: The starting position of the register for reading the mapping table data
GP: Number of the recipe table, starting from 1

Relay and Register

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ｜ WX1008	WY0 ｜ WY1008	WM0 ｜ WM9104	WS0 ｜ WS3088	T0 ｜ T1023	C0 ｜ C1279	R0 ｜ R34767	R34768 ｜ R35023	R35024 ｜ R35279	R35280 ｜ R43223	R43224 ｜ R47319	D0 ｜ D11999		V, Z P0 ~ P9
ID	○	○	○	○	○	○	○	○	○	○	○	○	0~1	
D	○	○	○	○	○	○	○	○	○	○	○	○		○
Gp	○	○	○	○	○	○	○	○	○	○	○	○	0~100	

Description

- [Fun188 Recipe Read] and [Fun189 Recipe Write] are used to read or write a large number of motion cor you can use [Fun181 Change Motion Control Parameters] or [Fun198 Mapping Table].
- Parameters can only be written when the axis stops.
- Operands
  Md mode: 0 use PLC register
  D recipe starting register: the starting address of the register to be written into the recipe table
  Gp writes the column of the formula table: writes the column of the recipe table, 0 writes all
- When the execution control [EN] is triggered by the upper differential, Fun188 will write the specified re
  When the execution control [EN] is triggered by the lower differential, all output indications are reset
- When writing into the recipe, the output indication [ACT] ON.
- When writing in the recipe, if there is an error, the output indication [ERR] ON.
- When writing the recipe is completed, the output indication [DN] ON.

Recipe Table

【Project Management】>【Motion Control】>【Motion Recipe】

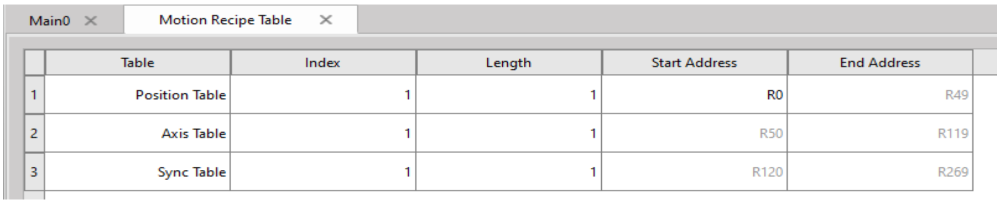


	Table	Index	Length	Start Address	End Address
1	Position Table	1	1	R0	R49
2	Axis Table	1	1	R50	R119
3	Sync Table	1	1	R120	R269

- Motion Recipe table
  Tables: Point table, Axis table, Synchronization table
  Index: Point table (number of points), Axis table (number of axes), Synchronization table (number of axes)
  Length: continuous point table or continuous axis
  Start address: The start address of the register for reading and writing recipes
- Please refer to the following table for the definition of the temporary register value of the motion recipe table.

Recipe Point Table

Start Addres+N	Item	Size	Type	L	Definition
R+0	Operation Mode	WORD	INT	1	16. Unuse 17. Single/ABS 18. Single/INC 19. Linear(2Axis)/ABS 20. Linear(2Axis)/INC 21. Linear(3Axis)/ABS 22. Linear(3Axis)/INC 23. Linear(4Axis)/ABS 24. Linear(4Axis)/INC 25. Arc/ABS 26. Arc/ INC 27. Arc 3D/ABS 28. Arc 3D/ INC 29. Helical/ABS 30. Helical/ INC 31. Single Velocity
R+1	Accerlation Profile	WORD	INT	1	2. T Curve 3. S Curve
R+2	Master Axis	WORD	INT	1	1~16 Non use = 0

R+3	Interpolation 1	WORD	INT	1	1~16 Non use = 0
R+4	Interpolation 2	WORD	INT	1	1~16 Non use = 0
R+5	Interpolation 3	WORD	INT	1	1~16 Non use = 0
R+6	Target Position Master Axis	DWORD	INT	2	Precision: Decimal Place (negative number allow)
R+8	Target Position Interpolation 1	DWORD	INT	2	Precision: Decimal Place (negative number allow)
R+10	Target Position Interpolation 2	DWORD	INT	2	Precision: Decimal Place (negative number allow)
R+12	Target Position Interpolation 3	DWORD	INT	2	Precision: Decimal Place (negative number allow)
R+14	Velocity	DWORD	INT	2	Precision: Decimal Place (positive number only)
R+16	Acceleration	DWORD	INT	2	Precision: Decimal Place (positive number only)
R+18	Deceleration	DWORD	INT	2	Precision: Decimal Place (positive number only)
R+20	Acceleration S Curve	WORD	INT	1	Precision: 0.1
R+21	Deceleration S Curve	WORD	INT	1	Precision: 0.1
R+22	Arc Mode	WORD	INT	1	3. Border Point 4. Center 5. Radius
R+23	Arc Direction	WORD	INT	1	2. CW 3. CCW
R+24	Arc (Border/Center) X coordinate	DWORD	INT	2	Precision: Decimal Place (negative number allow)
R+26	Arc (Border/Center) Y coordinate	DWORD	INT	2	Precision: Decimal Place (negative number allow)
R+28	Arc Radius	DWORD	INT	2	Precision: Decimal Place (positive number only)

R+30	Aux Radius	DWORD	INT	2	Precision: Decimal Place (positive number only)
R+32	Standby Time	DWORD	UINT	2	Unit: ms
R+34	Continuous Point	WORD	INT	1	1~1024 End = 0
R+35	Circle Revolution	WORD	UINT	1	0~65535
R+36	Continuous Mode	WORD	INT	1	4. Standby 5. Next Point Speed Continue 6. Current Point Speed Continue 7. Starting Speed Continue
R+37-41	Reserve			5	
R+42	Arc (Border/Center) Z coordinate	DWORD	INT	2	Precision: Decimal Place (negative number allow)

Recipe Axis Table

Start Addres+N	Item	Size	Type	L	Definition
R+0	Encoder Type	WORD		1	0 = Incremental 1 = Absolute
R+1	Unit	WORD		1	4. PLS 5. Mm 6. Deg 7. inch
R+2	Decimal Point	WORD		1	1000: 1 100: 0.1 10: 0.01 1: 0.001
R+3	Pulse/Revolution	DWORD		2	Precision: Decimal Place
R+5	Unit/Revolution	DWORD		2	Precision: Decimal Place
R+7	Velocity Unit	DWORD		1	3. PLS/Sec 4. PLS/min 5. RPM
R+8	Velocity Gain	DWORD		2	Precision: 0.001
R+10	Start Velocity	DWORD		2	Precision: Decimal Place
R+12	Max Motor Velocity	DWORD		2	Precision: 1 Unit: RPM
R+14	Default Acceleration	DWORD		2	Precision: Decimal Place
R+16	Default Deceleration	DWORD		2	Precision: Decimal Place
R+18	Soft Limit(+)	DWORD		2	Precision: Decimal Place (positive number only)
R+20	Soft Limit(-)	DWORD		2	Precision: Decimal Place (positive number only)
R+22	Following Error Window	DWORD		2	Precision: Decimal Place
R+24	Following Error Timeout	DWORD		2	Unit: ms
R+26	Pos Done Tolerance	DWORD		2	Precision: Decimal Place

R+28	Pos Done Check Time	DWORD		2	Unit: ms
R+30	Maximum Motor Torque	WORD		1	Precision: 0.1
R+31	Maximum Torque Limit(+)	WORD		1	Precision: 0.1
R+32	Maximum Torque Limit(-)	WORD		1	Precision: 0.1
R+33	Touch Probe1 Source	WORD		1	3. Disable 4. Input 5. Z Signal
R+34	Touch Probe1 Mode	WORD		1	4. Rising Edge Single 5. Rising Edge Continue 6. Falling Edge Single 7. Falling Edge Continue
R+35	Touch Probe2 Source	WORD		1	3. Disable 4. Input 5. Z Signal
R+36	Touch Probe2 Mode	WORD		1	4. Rising Edge Single 5. Rising Edge Continue 6. Falling Edge Single 7. Falling Edge Continue
R+37-40	Reserve			4	
R+41	Stop Mode	WORD		1	5. Deceleration Stop 7. Immediately Stop
R+42	Stop Deceleration	DWORD		2	Precision: Decimal Place
R+44	Homing Mode	WORD		1	99. Homing on current position 100. Forward-Falling Trigger 101. Backward-Falling Trigger 102. Z Signal-Forward-Rising Trigger 103. Z Signal-Forward-Falling Trigger 104. Forward- Rising Trigger 105. Backward-Rising Trigger 106. Z Signal-Backward-Rising Trigger

					107. Z Signal-Backward-Falling Trigger
R+45	Homing IO Source	WORD		1	2. From Servo Driver 3. From PLC
R+46	Homing Start Direction	WORD		1	2. Negative 3. Positive
R+47	Homing Origin Offset	DWORD		2	Precision: Decimal Place (negative number allow)
R+49	Homing Find Velocity	DWORD		2	Precision: Decimal Place
R+51	Homing Creep Velocity	DWORD		2	Precision: Decimal Place
R+53	Homing Deceleration	DWORD		2	Precision: Decimal Place
R+55	Limit Switch(-)(DI)	WORD		1	
R+56	Limit Switch(+)(DI)	WORD		1	
R+57	Homing Switch(DI)	WORD		1	
R+58	Homing Z Count	DWORD		2	
R+60	Jogging Base Velocity	DWORD		2	Precision: Decimal Place
R+62	Jogging Velocity	DWORD		2	Precision: Decimal Place
R+64	Jogging Acceleration	DWORD		2	Precision: Decimal Place
R+66	Jogging Deceleration	DWORD		2	Precision: Decimal Place
R+68	Inching Distance	DWORD		2	Precision: Decimal Place

Recipe Synchronous Table

Start Addres+N	Item	Size	Type	L	Definition
R+0	Input axis coordinate Unit	WORD		1	
R+1	Input axis coordinate decimal point	WORD		1	
R+2	Input axis period	DWORD		2	Precision: Decimal Place
R+4	Clutch OFF sliding time at deceleration stop	DWORD		2	
R+6	Input axis phase init method	WORD		1	
R+7	Sync master axis phase default value	DWORD		2	Precision: Decimal Place
R+9	Master axis phase default value after phase compensation	DWORD		2	Precision: Decimal Place
R+11	Main clutch input axis phase default value	DWORD		2	Precision: Decimal Place
R+13	Auxiliary clutch input axis phase default value	DWORD		2	Precision: Decimal Place
R+15	Cam input axis/clutch output axis	WORD		1	

	phase init method				
R+16	Main clutch output axis phase default value	DWORD		2	Precision: Decimal Place
R+18	Auxiliary clutch output axis phase default value	DWORD		2	Precision: Decimal Place
R+20	Reserve	DWORD		2	
R+22	Cam input axis phase default value	DWORD		2	Precision: Decimal Place
R+24	Cam output axis base coordinate	DWORD		2	Precision: Decimal Place
R+26	Master Axis 1 input selection	WORD		1	
R+27	Master Axis 1 external reference number	WORD		1	
R+28	Master Axis 1 prevent reverse	WORD		1	
R+29	Master Axis 1 coordinate transformation setting	WORD		1	
R+30	Master Axis 1 coordinate transformation numerator	DWORD		2	
R+32	Master Axis 1 coordinate transformation denominator	DWORD		2	

R+34	Master Axis 2 input selection	WORD		1	
R+35	Master Axis 2 external reference number	WORD		1	
R+36	Master Axis 2 prevent reverse	WORD		1	
R+37	Master Axis 2 coordinate transformation setting	WORD		1	
R+38	Master Axis 2 coordinate transformation numerator	DWORD		2	
R+40	Master Axis 2 coordinate transformation denominator	DWORD		2	
R+42	Aux Axis input selection	WORD		1	
R+43	Aux Axis external reference number	WORD		1	
R+44	Aux Axis prevent reverse	WORD		1	
R+45	Aux Axis coordinate transformation setting	WORD		1	
R+46	Aux Axis coordinate transformation numerator	DWORD		2	

R+48	Aux Axis coordinate transformation denominator	DWORD		2	
R+50	Master Axis compensation command value	DWORD		2	Precision: Decimal Place
R+52	Master Axis compensation change mode	WORD		1	
R+53	Master Axis compensation change time	DWORD		2	
R+55	Aux Axis compensation command value	DWORD		2	Precision: Decimal Place
R+57	Aux Axis compensation change mode	WORD		1	
R+58	Aux Axis compensation change time	DWORD		2	
R+60	Variable gear retio numerator	DWORD		2	
R+62	Variable gear retio denominator	DWORD		2	
R+64	Gear retio change mode	WORD		1	
R+65	Variable gear retio change time	DWORD		2	
R+67	Main clutch ON condition	WORD		1	

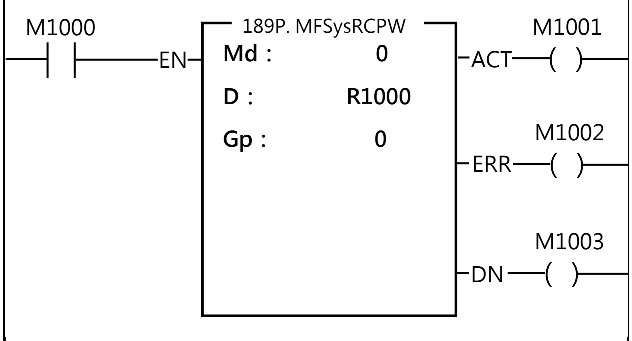
R+68	Main clutch ON setting value	DWORD		2	Precision: Decimal Place
R+70	Main clutch ON delay	DWORD		2	Precision: Decimal Place
R+72	Reserve	WORD		1	
R+73	Main clutch ON connection method	WORD		1	
R+74	Reserve	WORD		1	
R+75	Main clutch ON sliding curve	WORD		1	
R+76	Reserve	DWORD		2	
R+78	Main clutch ON sliding time	DWORD		2	
R+80	Main clutch ON following time	DWORD		2	
R+82	Main clutch ON follow-ups	DWORD		2	Precision: Decimal Place
R+84	Main clutch OFF condition	WORD		1	
R+85	Main clutch OFF setting value	DWORD		2	Precision: Decimal Place
R+87	Main clutch OFF delay	DWORD		2	Precision: Decimal Place
R+87	Reserve	WORD		1	
R+90	Main clutch OFF connection method	WORD		1	
R+91	Reserve	WORD		1	
R+92	Main clutch OFF sliding curve	WORD		1	
R+93	Reserve	DWORD		2	
R+95	Main clutch OFF sliding time	DWORD		2	

R+97	Aux clutch ON condition	WORD		1	
R+98	Aux clutch ON setting value	DWORD		2	Precision: Decimal Place
R+100	Aux clutch ON delay	DWORD		2	Precision: Decimal Place
R+102	Reserve	WORD		1	
R+103	Aux clutch ON connection method	WORD		1	
R+104	Reserve	WORD		1	
R+105	Aux clutch ON sliding curve	WORD		1	
R+106	Reserve	DWORD		2	
R+108	Aux clutch ON sliding time	DWORD		2	
R+110	Aux clutch ON following time	DWORD		2	
R+112	Aux clutch ON follow-ups	DWORD		2	Precision: Decimal Place
R+114	Aux clutch OFF condition	WORD		1	
R+115	Aux clutch OFF setting value	DWORD		2	Precision: Decimal Place
R+117	Aux clutch OFF delay	DWORD		2	Precision: Decimal Place
R+119	Reserve	WORD		1	
R+120	Aux clutch OFF connection method	WORD		1	
R+121	Reserve	WORD		1	
R+122	Aux clutch OFF sliding curve	WORD		1	
R+123	Reserve	DWORD		2	
R+125	Aux clutch OFF sliding time	DWORD		2	
R+127	Reserve	WORD*5		5	

R+132	Step Angle Compensation Base speed	DWORD		2	Precision: Decimal Place
R+134	Step Angle Compensation Base value	DWORD		2	Precision: Decimal Place
R+136	Step Angle Compensation value change mode	WORD		1	
R+137	Step Angle Compensation value change time	DWORD		2	
R+139	Cam data No.	WORD		1	
R+140	Cam stroke	DWORD		2	Precision: Decimal Place
R+142	Cam contact output No.	WORD		1	
R+143	Output filter time constant	DWORD		2	
R+145-149	Reserve				

FUN189 MFSysRCPW	Motion Control Recipe Writing (MFSysRCPW)	FUN189 MFSysRCPW

Example

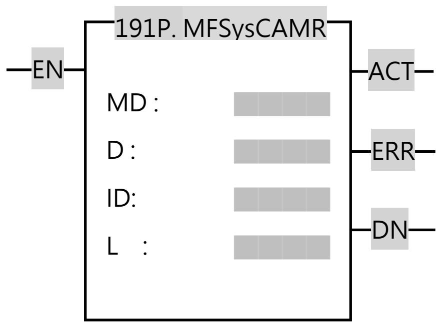
Ladder diagram	ST
	`MFSysRCPW ( EN:= M1000, Md:= 0, D:= R1000, Gp:= 0, ACT=> M1001, ERR=> M1002, DN=> M1003);`

- When M1000 is from OFF→ON, write all recipe tables from R1000.

7-22-15 Cam Read (MFSysCAMR)

FUN191 MFSysCAMR	Motion Control Cam Read	FUN191 MFSysCAMR

Symbol



Md: Mode
D: Cam initial register
ID: Cam number
L: Cam resolution

Relay and Register

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WM9104	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R35023	R35024 ∣ R35279	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999		V, Z P0 ~ P9
Md	○	○	○	○	○	○	○	○	○	○	○	○	0~1	
D	○	○	○	○	○	○	○	○	○	○	○	○		○
ID	○	○	○	○	○	○	○	○	○	○	○	○	1~16	
L	○	○	○	○	○	○	○	○	○	○	○	○	2048~32767	

Description

Operands

Md mode: 0 use PLC register

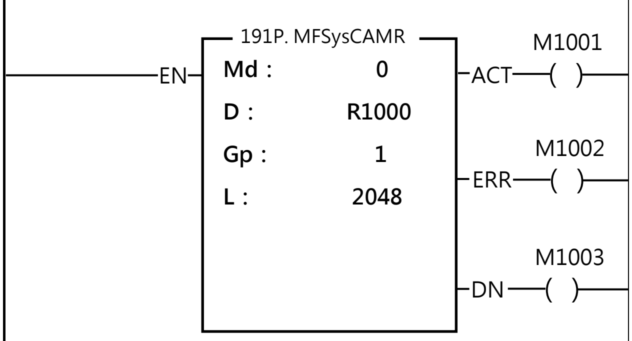
D cam start register: The start address of the register to be stored after reading the cam

ID cam number: Cam number

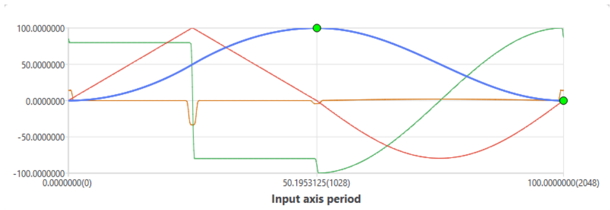
L cam resolution: The length of the temporary register to be stored after reading the cam

- When the execution control [EN] is triggered by the upper differential, Fun191 will read the specified cam to the specified register.
  When the execution control [EN] is triggered by the lower differential, all output indications are reset.
- When the cam is being read, the output indication [ACT] is ON.
- When reading the cam, if there is an error, the output indication [ERR] will be ON.
- When the reading of the cam is completed, the output indication [DN] ON.

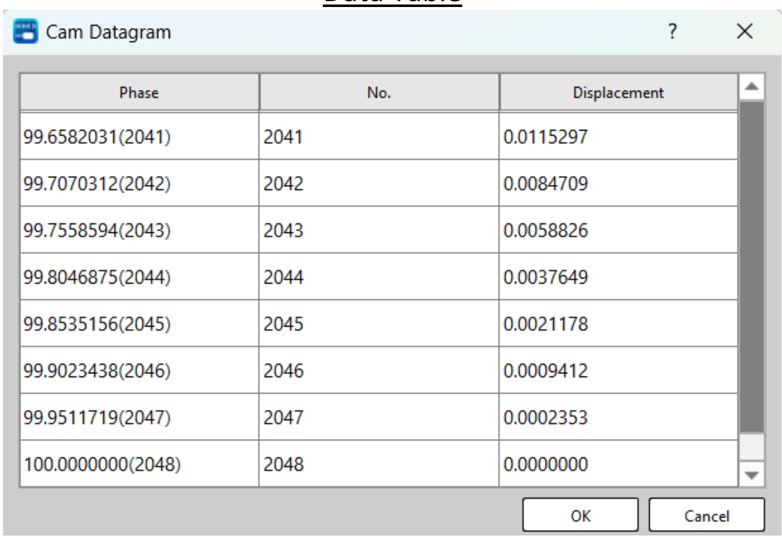
Example

Ladder diagram	ST
	`MFSysCAMR ( EN:= TRUE, Md:= 0, D:= R1000, ID:= 1, L:= 2048, ACT=> M1001, ERR=> M1002, DN=> M1003);`

CAM



Data Table



Cam Datagram

Phase	No.	Displacement
99.6582031(2041)	2041	0.0115297
99.7070312(2042)	2042	0.0084709
99.7558594(2043)	2043	0.0058826
99.8046875(2044)	2044	0.0037649
99.8535156(2045)	2045	0.0021178
99.9023438(2046)	2046	0.0009412
99.9511719(2047)	2047	0.0002353
100.0000000(2048)	2048	0.0000000

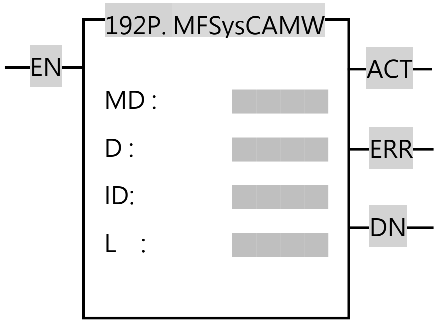
OK Cancel

When M1000 is from OFF→ON, read the cam ID: 1 data table and store 2048 in DR1000~DR5094.

7-22-16 Cam Write (MFSysCAMW)

FUN192 MFSysCAMW	Motion Control Cam Write (MFSysCAMW)	FUN192 MFSysCAMW

Symbol



Md: Mode
D: Cam initial register
ID: Cam number
L: Cam resolution

Relay and Register

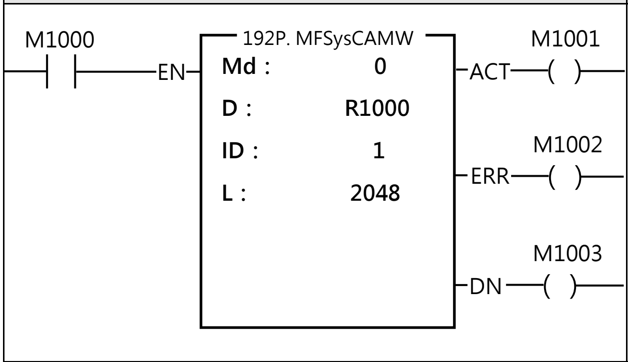
Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ~ WX1008	WY0 ~ WY1008	WM0 ~ WM9104	WS0 ~ WS3088	T0 ~ T1023	C0 ~ C1279	R0 ~ R34767	R34768 ~ R35023	R35024 ~ R35279	R35280 ~ R43223	R43224 ~ R47319	D0 ~ D11999		V, Z P0 ~ P9
Md	○	○	○	○	○	○	○	○	○	○	○	○	0~1	
D	○	○	○	○	○	○	○	○	○	○	○	○		○
ID	○	○	○	○	○	○	○	○	○	○	○	○	1~16	
L	○	○	○	○	○	○	○	○	○	○	○	○	2048~32767	

Description

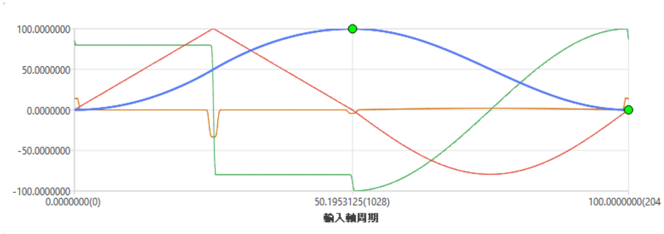
Operands
Md mode: 0 use PLC register
D cam start register: The start address of the register to be stored after reading the cam
ID cam number: Cam number
L cam resolution: The length of the temporary register to be stored after reading the cam

- When the execution control [EN] is triggered by the upper differential, Fun191 will read the specified cam to the specified register.
  When the execution control [EN] is triggered by the lower differential, all output indications are reset.
- When the cam is being read, the output indication [ACT] is ON.
- When reading the cam, if there is an error, the output indication [ERR] will be ON.
- When the reading of the cam is completed, the output indication [DN] ON.

Example	

Ladder diagram	ST
	`MFSysCAMW ( EN:= M1000, Md:= 0, D:= R1000, ID:= 1, L:= 2048, ACT=> M1001, ERR=> M1002, DN=> M1003);`

CAM



Data Table


凸輪資料表

相位	編號	位移
99.6582031(2041)	2041	0.0116203
99.7070312(2042)	2042	0.0085375
99.7558594(2043)	2043	0.0059289
99.8046875(2044)	2044	0.0037945
99.8535156(2045)	2045	0.0021344
99.9023438(2046)	2046	0.0009486
99.9511719(2047)	2047	0.0002372
100.0000000(2048)	2048	0.0000000

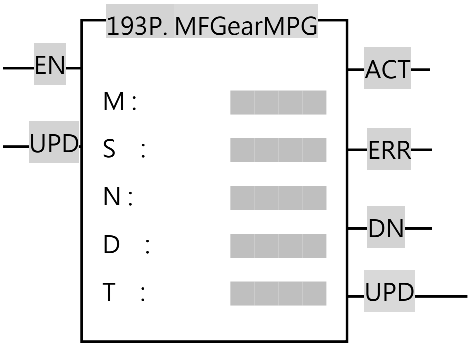
確定 取消

- When M1000 is from OFF→ON, write the cam ID from DR1000~DR5094: 1 data table 2048.

7-22-17 Handwheel (MFGearMPG)

FUN193 MFGearMPG	Handwheel (MFGearMPG)	FUN193 MFGearMPG

Symbol



Operand

M: EtherCat spindle number
S: EtherCat auxiliary shaft number
N: Gear ratio numerator
D: Gear ratio denominator
T: Conversion time (in ms)

Relay and Register

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ~ WX1008	WY0 ~ WY1008	WM0 ~ WM9104	WS0 ~ WS3088	T0 ~ T1023	C0 ~ C1279	R0 ~ R34767	R34768 ~ R35023	R35024 ~ R35279	R35280 ~ R43223	R43224 ~ R47319	D0 ~ D11999		V, Z P0 ~ P9
M	○	○	○	○	○	○	○	○	○	○	○	○	1~16,100~108	
S	○	○	○	○	○	○	○	○	○	○	○	○	1~16	○
N	○	○	○	○	○	○	○	○	○	○	○	○		
D	○	○	○	○	○	○	○	○	○	○	○	○		
T	○	○	○	○	○	○	○	○	○	○	○	○		

Description

- Fun 193 (EtherCAT hand wheel) integrates position-synchronized hand wheel related settings to provide users with quick setting of the hand wheel.
- Operands

  M spindle input source: EtherCAT_axis number 1~1
  : Encoder_Gray code 100 (X8~X15)
  : Encoder_hardware high-speed counter number 101 ~ 104 (HSC4~HSC7)
  S slave axis output target: EtherCAT_axis number 1~16
  ([M spindle input source] cannot be the same as [S slave axis output target])
  N Variable gear ratio numerator: positive and negative numbers, including the [decimal point position] of the [motion axis setting] in the [motion control]
  ([Axis unit] set mm, [Decimal point position] set 0.001, N: DR0 = 1000 is equal to 1.000mm)

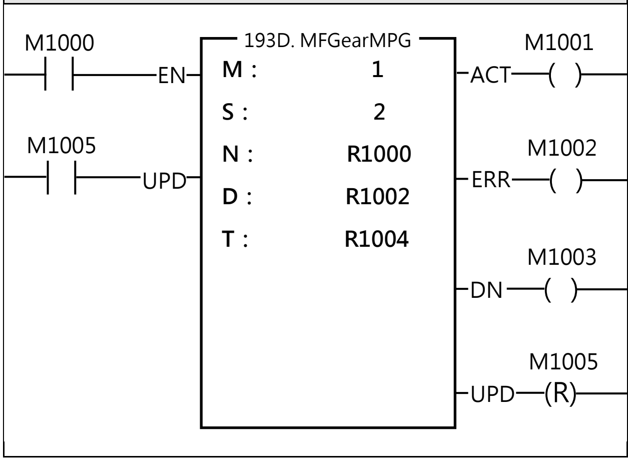
D Variable gear ratio denominator: positive number (a real number greater than zero), including the [decimal point position] of [motion axis setting] in [motion control]

T conversion time (ms): positive number (real number greater than zero), the unit is ms

- When the execution control [EN] is triggered by the upper differential, Fun193 uses the current parameters to start the synchronous control of the handwheel position.

  When the execution control [EN] is triggered by the lower differential, Fun193 stops the synchronous control of the handwheel position and resets all output indicators.
- In handwheel synchronous control, if the update parameter [UPD] changes to 1, this command will update the handwheel parameters (N, D, T) immediately.
- When the hand wheel is under synchronous control, the output indication [ACT] is ON.
- During the synchronous control of the manual wheel, if an error occurs, the output indication [ERR] will be ON.
- When the update of the manual wheel parameters is completed, the output indication [UPD] ON.

FUN193 MFGearMPG	Handwheel (MFGearMPG)	FUN193 MFGearMPG
Example		

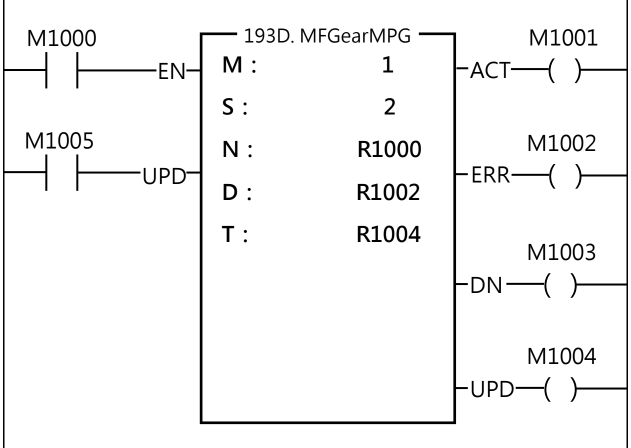
Ladder diagram	ST
	MFGearMPG _D(EN:= M1000, UPD_in:= M1005, M:= 1, S:= 2, N:= R1000, D:= R1002, T:= R1004, ACT=> M1001, ERR=> M1002, DN=> M1003, UPD_out=> M1004); IF M1004 THEN M1005 := FALSE; END_IF


```
MFGearMPG _D( EN:= M1000, UPD_in:=
M1005, M:= 1, S:= 2, N:= R1000, D:=
R1002, T:= R1004, ACT=> M1001, ERR=>
M1002, DN=> M1003, UPD_out=> M1004);
IF M1004 THEN
M1005 := FALSE;
END_IF
```

Axis Parameter Setting

		1.Axis_1	2.Axis_2
Basic Setting	Axis Name	Axis_1	Axis_2
	Axis Type	Virtual Servo	Virtual Servo
	Encoder Type	Incremental	Incremental
Unit Setting	Unit	mm	mm
	Decimal Point	0.001	0.001
	Pulse/Revolution	131072 PLS/Rev	131072 PLS/Rev
	Unit/Revolution	1.000 mm/Rev	1.000 mm/Rev
	Velocity Unit	Command Position/sec	Command Position/sec
	Velocity Gain	1.000	1.000

When M1000 is from OFF→ON, start the manual wheel synchronization according to the current Fun193 parameters (M: EtherCAT axis 1, N: EtherCAT axis 2, N: variable gear ratio numerator 0.001, D variable gear ratio denominator 0.001, T: 1ms).

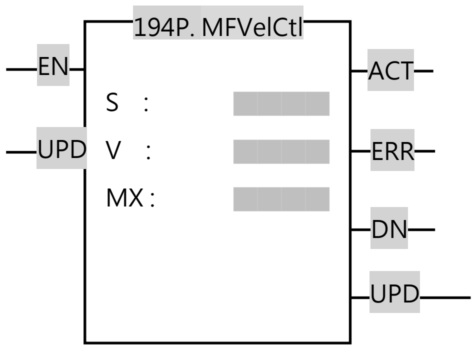
Ladder diagram	ST
	`MFGearMPG _D( EN:= M1000, UPD_in:= M1005, M:= 1, S:= 2, N:= R1000, D:= R1002, T:= R1004, ACT=> M1001, ERR=> M1002, DN=> M1003, UPD_out=> M1004);`

After changing the parameters (D variable gear ratio denominator 0.002), when M1005 is from OFF→ON, update the hand wheel according to the changed parameters. After the parameter update is completed, the output indication [UPD] is ON, and the stroke of the slave axis of the hand wheel is halved.

7-22-18 Velocity Control Mode (MFVelCtl)

FUN194 MFVelCtl	Velocity Control Mode (MFVelCtl)	FUN194 MFVelCtl

Symbol



S: Axis number
V: Speed command
MX: Maximum torque

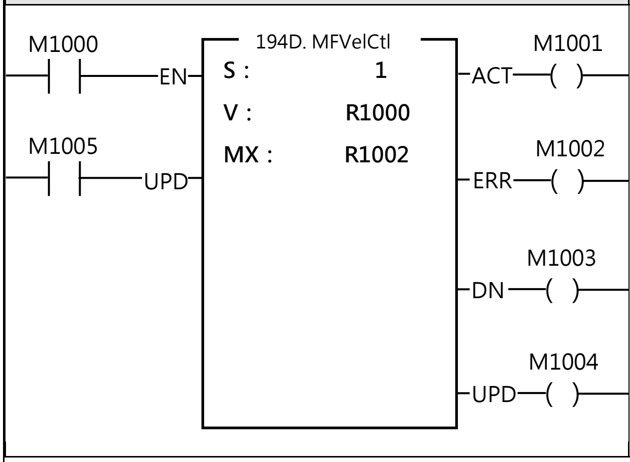
Relay and Register

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WM9104	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R35023	R35024 ∣ R35279	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999		V, Z P0 ~ P9
S	○	○	○	○	○	○	○	○	○	○	○	○	1~16	
V	○	○	○	○	○	○	○	○	○	○	○	○		○
MX	○	○	○	○	○	○	○	○	○	○	○	○		

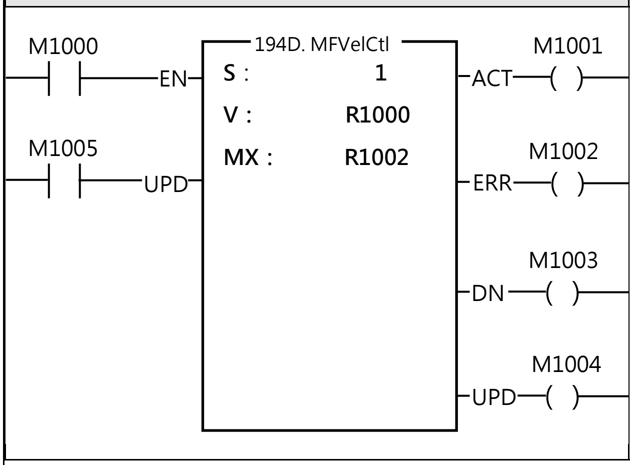
Description

- Operands
  S speed control axis: EtherCAT_ axis number 1~16
  V speed: speed setting value, unit Pulses/s
  MX maximum torque limit: when the speed cannot reach the speed setting value, the maximum torque limit, 0 equals no limit, unit 0.0%
- When the execution control [EN] is triggered by the upper differential, Fun194 uses the current parameters to start the axis speed control
  When the execution control [EN] is triggered by the lower differential, Fun194 stops the axis speed control and resets all output indications
- In axis speed control, if the update parameter [UPD] becomes 1, this command will update the speed control parameters (V, MX) immediately.
- When the axis speed is under control, the output indicator [ACT] ON.
- During axis speed control, if an error occurs, the output indication [ERR] will be ON.
- When updating the speed control parameters is completed, the output indication [UPD] ON.

FUN194 MFVelCtl	Velocity Control Mode (MFVelCtl)	FUN194 MFVelCtl
Example		

Ladder diagram	ST
	`MFVelCtl_D( EN:= M1000, UPD_in:= M1005, S:= 1, V_n:= R1000, MX:= R1002, ACT=> M1001, ERR=> M1002, DN=> M1003, UPD_out=> M1004);`

- When M1000 is from OFF→ON, start velocity control according to the current Fun194 parameters (S: EtherCAT axis 1, V: 131072 Pulses per second, MX: no torque limit).

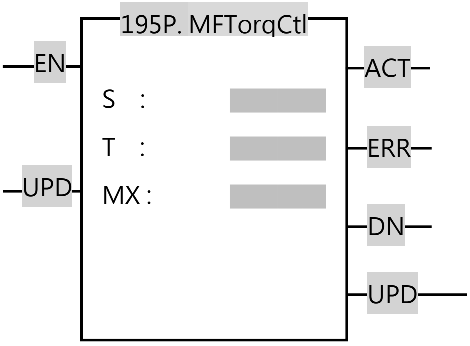
Ladder diagram	ST
	`MFVelCtl_D( EN:= M1000, UPD_in:= M1005, S:= 1, V_n:= R1000, MX:= R1002, ACT=> M1001, ERR=> M1002, DN=> M1003, UPD_out=> M1004);`

- After changing the parameter (V: 262144 Pulses per second), when M1005 changes from OFF to ON, the parameter update is completed according to the changed parameter update speed, and the output indicator M1004 [UPD] ON is turned on, and the speed doubles.

7-22-19 Torque Control Mode (MFTorqCtl)

FUN195 MFTorqCtl	Torque Control Mode (MFTorqCtl)	FUN195 MFTorqCtl

Symbol



S: Axis number
V: Speed command
MX: Maximum torque

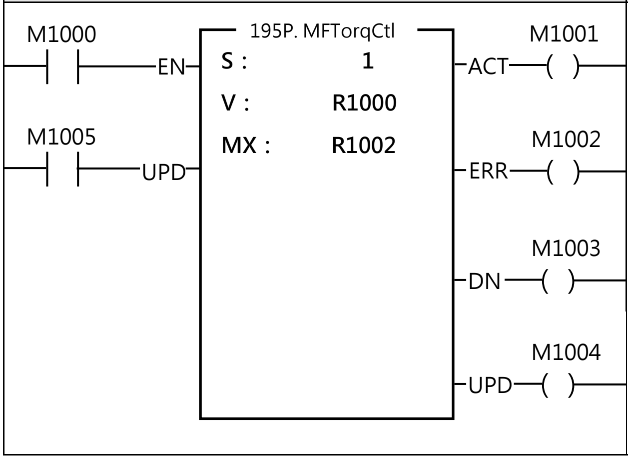
Relay and Register

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ︱ WX1008	WY0 ︱ WY1008	WM0 ︱ WM9104	WS0 ︱ WS3088	T0 ︱ T1023	C0 ︱ C1279	R0 ︱ R34767	R34768 ︱ R35023	R35024 ︱ R35279	R35280 ︱ R43223	R43224 ︱ R47319	D0 ︱ D11999		V, Z P0 ~ P9
S	○	○	○	○	○	○	○	○	○	○	○	○	1~16	
T	○	○	○	○	○	○	○	○	○	○	○	○		○
MX	○	○	○	○	○	○	○	○	○	○	○	○		

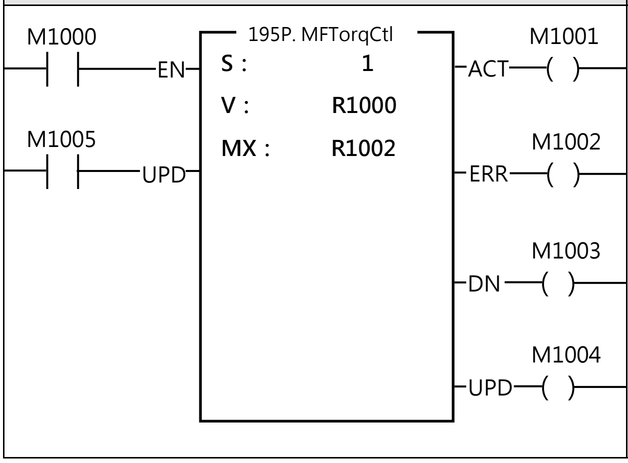
Description

- Operands
  S torque control axis: EtherCAT_ axis number 1~16
  T torque: Torque setting value, unit 0.0%
  MX Maximum speed limit: When the torque cannot reach the torque setting value, the maximum speed limit, 0 equals no limit, unit rpm
- When the execution control [EN] is triggered by the upper differential, Fun195 uses the current parameters to start the shaft torque control
  When the execution control [EN] is triggered by the lower differential, Fun195 stops the shaft torque control and resets all output indications
- In axis torque control, if the update parameter [UPD] becomes 1, this command will update the torque control parameters (T, MX) immediately.
- When the axis torque is under control, the output indicator [ACT] ON.
- During axis torque control, if an error occurs, the output indication [ERR] will be ON.
- When updating the torque control parameters is completed, the output indication [UPD] ON.

FUN195 MFTorqCtl	Torque Control Mode (MFTorqCtl)	FUN195 MFTorqCtl
Example		

Ladder diagram	ST
	`MFTorqCtl ( EN:= M1000, UPD_in:= M1005, S:= 1, V_n:= R1000, MX:= R1002, ACT=> M1001, ERR=> M1002, DN=> M1003, UPD_out=> M1004);`

- When M1000 is from OFF to ON, torque control is started according to the current Fun194 parameters (S: EtherCAT axis 1, T: 5.0%, MX: no speed limit).

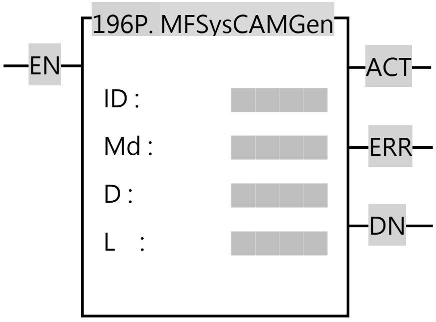
Ladder diagram	ST
	`MFTorqCtl ( EN:= M1000, UPD_in:= M1005, S:= 1, V_n:= R1000, MX:= R1002, ACT=> M1001, ERR=> M1002, DN=> M1003, UPD_out=> M1004);`

- After changing the parameter (T: 10.0%), when M1005 changes from OFF→ON, the torque will be updated according to the changed parameter. After the parameter update is completed, the output indication M1004 [UPD] ON will double the torque.

7-22-20 Cam Generate (MFSysCAMGen)

FUN196 MFSysCAM Gen	Cam Generate (MFSysCAMGen)	FUN196 MFSysCAM Gen

Symbol



ID: Axis number
Md: Mode
D: Start address of scratchpad
L: Length

Relay and Register

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0~WX1008	WY0~WY1008	WM0~WM9104	WS0~WS3088	T0~T1023	C0~C1279	R0~R34767	R34768~R35023	R35024~R35279	R35280~R43223	R43224~R47319	D0~D11999		V, Z P0 ~ P9
ID	○	○	○	○	○	○	○	○	○	○	○	○	1~16	
Md													0~1	
D	○	○	○	○	○	○	○	○	○	○	○	○		
L	○	○	○	○	○	○	○	○	○	○	○	○		

Description

- Operands
  ID cam number: 1~16
  Md cam generation mode: 0 is the same as cam table, 1 is chasing shear curve
  D register start bit: set the start register of the cam
  L The number of cam curve segments: Mode 0 only has the setting of each segment of the cam, and other modes do not need to be set
- When the execution control [EN] is triggered by the upper differential, Fun196 will generate the cam according to the set mode.
  When the execution control [EN] is triggered by the lower differential, all output indications are reset.
- When the cam is being generated, the output indication [ACT] ON.
- When the cam is generating, if there is an error, the output indication [ERR] will be ON.
- When the cam generation is completed, the output indication [DN] will be ON.

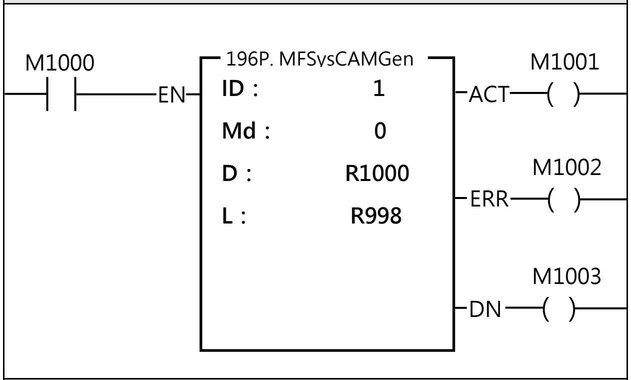
FUN196 MFSysCAM Gen	Cam generate (MFSysCAMGen)	FUN196 MFSysCAM Gen

Mode 0

Register	Item	Definition	
D+0	Start Phase	0~ Cam Resolution	First Cam Segment
D+2	End Phase	0~ Cam Resolution (Must seamlessly transition to the start phase of the next segment)	
D+4	Offset	0~1000000000 (0~100.0000000%)	
D+6	Cam Profile	0:Constant Velocity 1:Constant Acceleration 2:Cycloid 3:Simple Harmonic 4:Modified Constant Velocity 5: Modified Trapezoid 6: Modified Harmonic 7: Trapezoid 8:One-Dwell Cycloid, m=1 9: One-Dwell Cycloid, m=2/3 10: One-Dwell Trapezoid, Ferguson 11: One-Dwell Modified Harmonic 12: One-Dwell Trapezoid 13: One-Dwell Modified Trapezoid 14: One-Dwell Modified Constant Velocity 15:NC2 16:Asymmertic Cycloid 17: Asymmertic Modified Trapezoid 18:Cubic Curve 19:Quintic Curve	
D+8	Start Speed	Round to 3 decimal places	
D+10	End Speed	Round to 3 decimal places	
D+12	Start Acceleration	Round to 3 decimal places	
D+14	End Acceleration	Round to 3 decimal places	
D+15	Start Phase	0~ Cam Resolution	Second Cam Segment
D+16	End Phase	0~ Cam Resolution (Must seamlessly transition to the start phase of the next segment)	
		⋮	

Example	

Ladder diagram



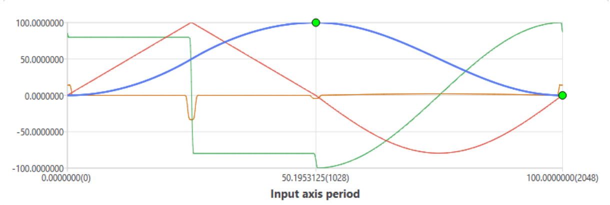
ST

```
MFSysCAMGen ( EN:= M1000, ID:= 1, Md:=
0, D:= R1000, L:= R998, ACT=> M1001,
ERR=> M1002, DN=> M1003);
```

D

Register	Item	Definition	
R1000	Start Phase	0	First Cam Segment
R1001	End Phase	1028	
R1002	Offset	1000000000	
R1003	Cam Profile	1:Constant Acceleration	
R1004	Start Speed	0	
R1005	End Speed	0	
R1006	Start Acceleration	0	
R1007	End Acceleration	0	
R1008	Start Phase	1028	Second Cam Segment
R1009	End Phase	2048	
R1010	Offset	0	
R1011	Cam Profile	3:Simple Harmonic	
R1012	Start Speed	0	
R1013	End Speed	0	
R1014	Start Acceleration	0	
R1015	End Acceleration	0	

CAM



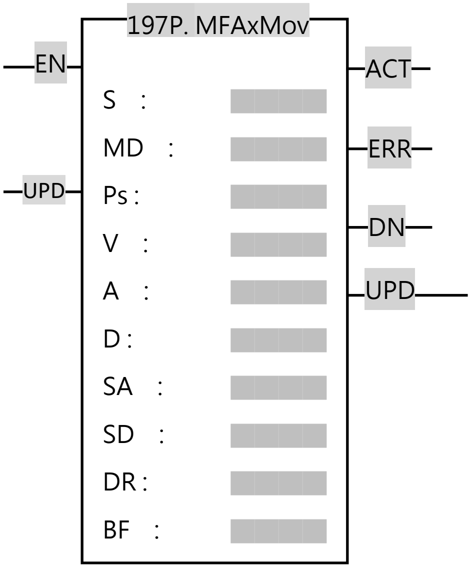
Cam Parameter

	開始相位	結束相位	偏移	凸輪輪廓
1	0.0000000(0)	50.1953125(1028)	100.0000000	等加速度
2	50.1953125(1028)	100.0000000(2048)	0.0000000	簡諧

- When M1000 is from OFF to ON, the cam is generated according to the current Fun196 number (ID: cam number 1, Md: mode 0, D: setting the cam generation parameters from R1000, L: 2-stage cam curve).

7-22-21 Axis Movement (MFAxMov)

FUN197 MFAxMov	Axis Movement	FUN197 MFAxMov
Symbol		



S: EtherCAT control axis
MD: Operating mode
PS: Target position
V: Speed
A: Acceleration
D: Deceleration
SA: S acceleration curve %
SD: S deceleration curve%
DR: Direction
BF: Speed continuous mode

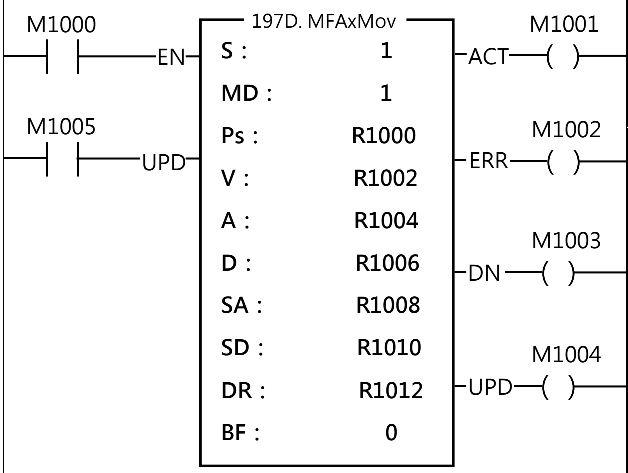
Relay and Register

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 – WX1008	WY0 – WY1008	WM0 – WM9104	WS0 – WS3088	T0 – T1023	C0 – C1279	R0 – R34767	R34768 – R35023	R35024 – R35279	R35280 – R43223	R43224 – R47319	D0 – D11999		V, Z P0 ~ P9
S	○	○	○	○	○	○	○	○	○	○	○	○	1~16	
MD													0~1	
Ps	○	○	○	○	○	○	○	○	○	○	○	○		
V	○	○	○	○	○	○	○	○	○	○	○	○		
A	○	○	○	○	○	○	○	○	○	○	○	○		
D	○	○	○	○	○	○	○	○	○	○	○	○		
SA	○	○	○	○	○	○	○	○	○	○	○	○		
SD	○	○	○	○	○	○	○	○	○	○	○	○		
DR	○	○	○	○	○	○	○	○	○	○	○	○	1~2	
BF													0~5	

FUN197 MFAxMov	Axis Movement	FUN197 MFAxMov
Description		

- Operands
  S EtherCAT control axis: EtherCAT_ axis number 1~16
  MD operation mode: 0 absolute, 1 relative
  PS target position: positive and negative numbers, including the [decimal point position] of the [motion axis setting] in the [motion control]
  ([Axis unit] set mm, [Decimal point position] set 0.001, PS: DR0 = 1000 is equal to 1.000mm)
  V speed: positive number (a real number greater than zero), including the [decimal point position] of the [motion axis setting] in the [motion control]
  A Acceleration: positive number (a real number greater than zero), including the [decimal point position] of the [motion axis setting] in the [motion control]
  D Deceleration: positive number (a real number greater than zero), including the [decimal point position] of the [motion axis setting] in the [motion control]
  SA S acceleration curve %: positive integer, 0~1000 ‰
  SD S deceleration curve %: positive integer, 0~1000 ‰
  DR direction: 1 positive direction, 2 negative direction
  BF: Speed continuous mode: 0 executes the current command immediately, 1 waits for the end of the previous command, 2 selects the lower speed continuous, 3 selects the previous command speed continuous, 4 selects the current command speed continuous, 5 selects the higher speed continuous
- When the execution control [EN] is triggered by the upper differential, Fun197 executes the axis position control.
  When the execution control [EN] is triggered by the lower differential, Fun197 stops the axis position control and resets all output indications.
- In axis position control, if the update parameter [UPD] becomes 1, this command will immediately update the position control parameters (S, PS, V, A, D, SA, SD, DR).
- When the axis position is under control, the output indicator [ACT] ON.
- During axis position control, if an error occurs, the output indication [ERR] will be ON.
- When the axis position control is completed, the output indication [DN] will be ON.
- When updating the position control parameters is completed, the output indication [UPD] ON.

Example	

Ladder diagram	ST
	MFAxMov_D(EN:= M1000, UPD_in:= M1005, S:= 1, MD:= 1, Ps:= R1000, V_n:= R1002, A:= R1004, D:= R1006, SA:= R1008, SD:= R1010, DR:= R1012, BF:= 0, ACT=> M1001, ERR=> M1002, DN=> M1003, UPD_out=> M1004);


```
MFAxMov_D( EN:= M1000, UPD_in:= M1005,
S:= 1, MD:= 1, Ps:= R1000, V_n:= R1002,
A:= R1004, D:= R1006, SA:= R1008, SD:=
R1010, DR:= R1012, BF:= 0, ACT=> M1001,
ERR=> M1002, DN=> M1003, UPD_out=>
M1004);
```

Axis Parameter Setting

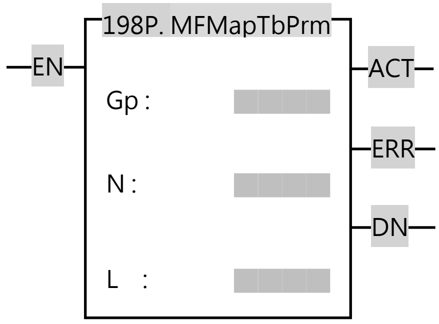
		1.Axis_1	2.Axis_2
Basic Setting	Axis Name	Axis_1	Axis_2
	Axis Type	Virtual Servo	Virtual Servo
	Encoder Type	Incremental	Incremental
Unit Setting	Unit	mm	mm
	Decimal Point	0.001	0.001
	Pulse/Revolution	131072 PLS/Rev	131072 PLS/Rev
	Unit/Revolution	1.000 mm/Rev	1.000 mm/Rev
	Velocity Unit	Command Position/sec	Command Position/sec
	Velocity Gain	1.000	1.000

- ent Fun197 parameters (S: EtherCAT axis 1, MD: relative position, PS: move to 10.000mm, V: speed 1.000mm/s, A: acceleration 100.000mm/s^2, D: Deceleration 100.000mm/s^2, SA: S acceleration curve 0.0%, SD: S deceleration curve 0.0%, DR: forward direction, BF: execute current command immediately) to execute position control.

7-22-22 Mapping Table Setting (MFMapTbPrm)

FUN198 MFMapTbPrm	Mapping Table Setting (MFMapTbPrm)	FUN198 MFMapTbPrm

Symbol



Gp: Mapping table group number
N: Mapping start table number
L: Mapping continuous length

Relay and Register

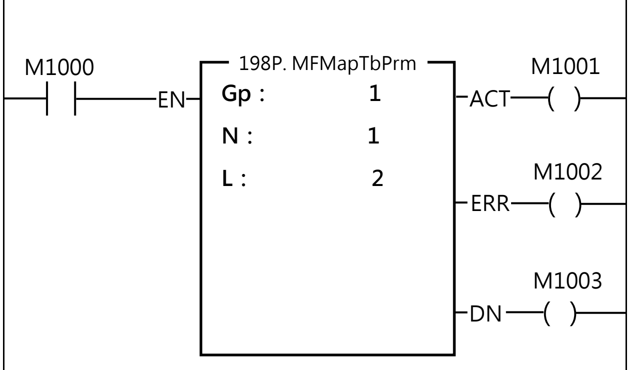
Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX1008	WY0 ∣ WY1008	WM0 ∣ WM9104	WS0 ∣ WS3088	T0 ∣ T1023	C0 ∣ C1279	R0 ∣ R34767	R34768 ∣ R35023	R35024 ∣ R35279	R35280 ∣ R43223	R43224 ∣ R47319	D0 ∣ D11999		V, Z P0 ~ P9
Gp													0~64	
N	○	○	○	○	○	○	○	○	○	○	○	○	0~1024	
L	○	○	○	○	○	○	○	○	○	○	○	○	1~1024	

Description

- [Fun198 Write Mapping Table] is used to change a single or a small number of motion control parameters. If you need to read or write a large number of motion control parameters, you can use [Fun188 Recipe Read] and [Fun189 Recipe Write].
- Operands
  Gp mapping table group number: group number 1~16, 0 means all groups.
  N Mapping table start table number: mapping table number 1~1024, 0 means the entire mapping table
  L Map Consecutive Length: Number of Consecutive Map Items
- When the execution control [EN] is triggered by the upper differential, Fun198 will map (write) the PLC temporary register to the motion control parameters.
  When the execution control [EN] is triggered by the lower differential, all output indications are reset.
- When the mapping is being written, the output indication [ACT] ON.
- When the mapping is being written, if an error occurs, the output indication [ERR] will be ON.
- When the mapping is written in, the output indication [DN] will be ON.

FUN198 MFMapTbPrm	Mapping Table Setting (MFMapTbPrm)	FUN198 MFMapTbPrm

Example

Ladder diagram	ST
	MFMapTbPrm (EN:= M1000, Gp:= 1, N:= 1, L:= 2, ACT=> M1001, ERR=> M1002, DN=> M1003);

Mapping Table

1:PM1

	Comment	Table	Index	Item	Address
1		Axis Table	1	19.Jogging Velocity	R9000
2		Axis Table	1	22.Inching Distance	R9002

Motion Axis Setting Table

Jogging	Jogging Base Velocity	0.100 mm/s
	Jogging Velocity	1.000 mm/s[2.000 mm/s]
	Jogging Acceleration	1000.000 mm/s^2
	Jogging Deceleration	1000.000 mm/s^2
	Inching Distance	5.000 mm[6.000 mm]

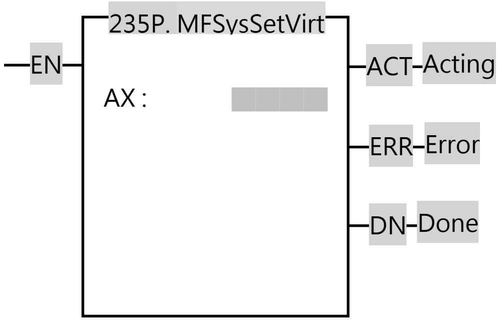
- When M1000 is from OFF to ON, according to the current Fun198 parameters (Gp 1: mapping table 1 (1: PM), N: starting from the first line of the mapping table (1: PM1), L: length 1) to execute mapping table writing, It can be seen from the motion axis setting table that the JOG speed has been modified to 2.000mm/s^2, and the inch movement distance has been modified to 6.000mm.

7-22-23 Real Axis to Virtual Axis (MFSysSetVirt)

FUN235 MFSysSetVirt	Real Axis to Virtual Axis	FUN235 MFSysSetVirt

Symbol

Ladder



Operand

AX: Axis number to be converted

EN: Trigger command

ACT: Acting

ERR: Conversion error

DN: Execution complete

Relay and Register

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX1008	WY0 | WY1008	WM0 | WM9104	WS0 | WS3088	T0 | T1023	C0 | C1279	R0 | R34767	R34768 | R35023	R35024 | R35279	R35280 | R43223	R43224 | R47319	D0 | D11999		V, Z P0 ~ P9
ID	○	○	○	○	○	○	○	○	○	○	○	○	1~16	○

Description

- This command is to convert real axis into virtual axis.
- Make sure the motion control system is in stop state before use, if it is in initialization state, ERR will output 1.
- If you need to stop the initialized system, you can refer to the instruction of FUN177 stop all motion flows.
- For details of this command, please refer to the instructions in the motion control manual.

Example

Ladder

To be supplemented

-

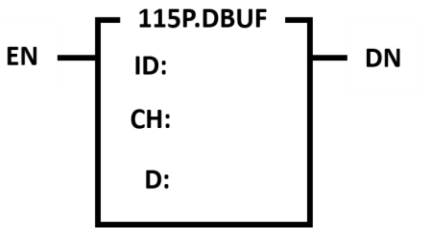
7-23 Other Instructions (FUN115, FUN258)

7-23-1 Data Buffering (DBUF)

FUN115P DBUF	Data Buffering	FUN115P DBUF

Symbol

Ladder Symbol



ID: Expansion module ID
CH: The channel designated for expansion module (0~3)
D: Starting position where the data will be saved.

Range / Operand	HR	IR	OR	SR	ROR	DR	K	XR
	R0 ~ R34767	R34768 ~ R34895	R35024 ~ R35151	R35280 ~ R43223	R43224 ~ R47319	D0 ~ D11999		V、Z ~ P0 – P9
ID							0-127	○
CH	○	○	○	○	○*	○*	0-63	○
D	○				○	○		

Description

It is used to obtain the data buffered on the module, which is suitable for expansion modules that have analog input and support data buffering function. The buffered data collected through this command will not be limited by the program scan cycle, but will be collected based on the sampling cycle set by the module.

FUN115P DBUF	Data Buffering	FUN115P DBUF
Example		

The data buffer function can be controlled through the relay, and the digital operation value can be stored in the data buffer area to observe the change of the digital operation value.

Use methods and instructions

Each buffer point updates the digital operation value to the data buffer area according to the processing time of the A/D conversion mode.

Each channel can store up to 600 points/ch.

Example:

When the cache points are set to 600 and the pre-trigger data points are set to 50:

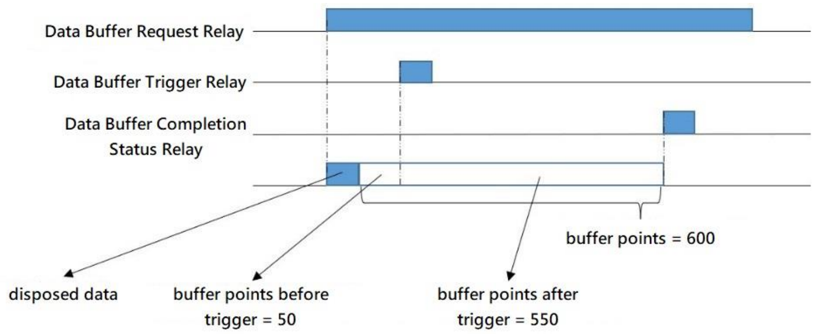


Fig. 137: Example diagram of the data buffering function

Setting and preset value

Setting	Preset Value
Buffer Points Before Trigger	200
Buffer Points	600

Table 65: Setting of data buffering function

FUN115P DBUF	Data Buffering	FUN115P DBUF

The following table shows how to use the data buffer function:

Run-time Relay Control

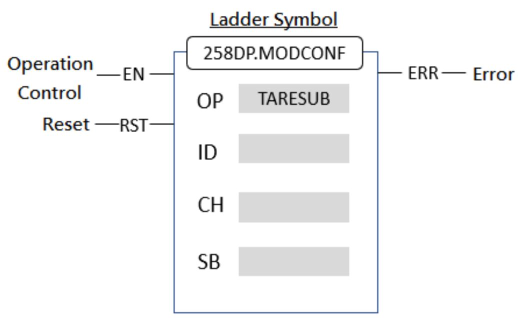
Data Buffer Relay	Description	Setting
Data Buffer Request Relay	Buffer Request	Off->On: Start buffering On->Off: Suspend buffering
Data Buffer Trigger Relay	Trigger	Off->On: Trigger data buffer relay
Data Buffer Completion Status Relay	Buffer Completion Data	Off->On: The specified cache points are completed, and the cache can be read through command 115 (DBUF function). On->Off: Data Buffer Request Relay: On -> Off, Off when the buffering is turned off. Data Buffer Completion Status Relay: On->Off->On, Off when retriggered, until Off->On after the buffer points are completed.

Table 66: Steps to use the data buffering function

After the data buffering is completed, use Fun115 DBUF to read the buffered data stored in the module to the address of the PLC designated register.

7-23-2 Tare Weight Deduction Command

FUN258P MODCONF	Tare Weight Deduction Command	FUN258P MODCONF
Symbol		



OP: TARESUB
ID: Expansion module ID number (0~N)
CH: The channel position to be deducted (0~N)
SB: The 32BIT value of the tare weight to be deducted

Operand \ Range	HR	IR	OR	SR	ROR	DR	K	XR
	R0 – R34767	R34768 – R34895	R35024 – R35151	R35280 – R43223	R43224 – R47319	D0 – D11999		V、Z P0~P9
FUN							1、2	○
ID	○	○	○	○	○*	○*	0~63	○
CH	○	○	○	○	○	○	○	○
SUB	○	○	○	○	○	○	○	○

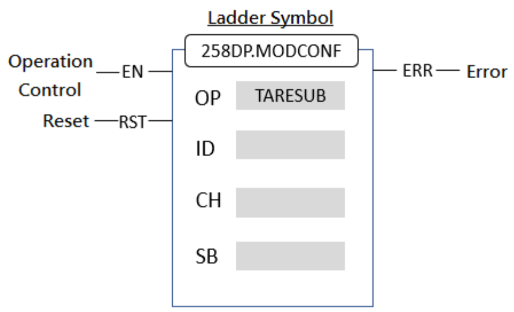
Description	

- To subtract the custom tare weight, you must change the config setting to “digital mode”. In the “light touch mode”, the current gross weight will be regarded as the tare weight directly deducted.
- Removing the fixed tare weight and recalibrating it may benefit from improved accuracy.
- When the Tare weight deduction command is enabled, if it is "light touch mode," it is the automatic parameter setting mode subtracting the current scale reading value.
- When the command of tare weight deducting is enabled, if the command mode is set to "digital", it is the manual parameter setting mode. At this time, the user can set the tare weight to be deducted by himself. When the command to enable tare weight deducting is sent, the command will subtract the corresponding weight according to the parameters set by the user.
- When RST OFF->ON, the setting before control will be restored.

7-23-3 Tare Weight Offset Command

FUN258P MODCONF	Tare Weight Offset Command	FUN258P MODCONF

Symbol



OP: TAREZROFFSET

ID: Expansion module ID number (0~N)

CH: The channel position to be deducted (0~N)

WR: Starting register for offset command setting

範圍 / 運算元	HR	IR	OR	SR	ROR	DR	K	XR
	R0 | R34767	R34768 | R34895	R35024 | R35151	R35280 | R43223	R43224 | R47319	D0 | D11999		V、Z P0~P9
OP							TAREZROFFSET	○
ID	○	○	○	○	○*	○*	0~63	○
CH	○	○	○	○	○	○	○	○
SB	○	○	○	○	○	○	○	○

Description

WR list:

WR+0	INA gain
WR+1	ADC gain
WR+2	digital value, 32 bits
WR+3	

FUN258P MODCONF	Tare Weight Offset Command	FUN258P MODCONF
Description		

- Remove the fixed tare weight. By setting the Instrumentation amplifier gain and ADC gain, it is possible to improve ADC conversion accuracy.
- Automatically parameter setting mode set the command mode to 0 and send the command to enable the tare zero function. The module will automatically calculate the appropriate Instrumentation amplifier gain, ADC gain, and Digital value and send it back to the PLC.
- User manual setting mode, MD=1, send a command to enable the tare zero function, and the command will be accompanied by the Instrumentation amplifier gain, ADC gain and Digital value set by the user. The Instrumentation amplifier gain is 433.92, 216.96 and 108.48, the ADC gain is 1, 2, 4, 8 and 16, and the Digital value setting range is 1~ 56874 (2.1696 V).
- The formula for calculating fixed weight，

$$\text{Fixed tare weight}= \left(\frac{DAC\ digital\ value}{65535} \times 2.5\right) \times \frac{Rated\ capacity \times Number\ of\ LC\ sensor}{INA\ gain \times Excitation\ voltage \times Rated\ ou}$$

- The suggested formula for ADC/INA Gain setting，

$$\frac{Max\ weighting\ capacity}{Rated\ capacity \times Number\ of\ LC\ sensors} \times INA\ gain \times \mathrm{ADC}\ gain \leq 500 \quad = \left(\frac{DAC\ digital\ value}{65535} \times 2.5\right) \times$$

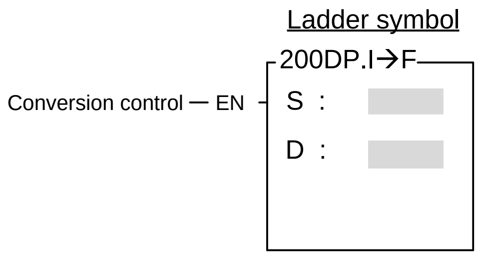
$$\frac{Rated\ capacity \times Number\ of\ LC\ sensors}{INA\ gain \times Excitation\ voltage \times Rated\ output}$$

*The tare weight offset command is only supported by the LCR module, not by the LC.

7-24 Floating Point Instructions (FUN200~220)

7-24-1 CONVERSION OF INTEGER TO FLOATING POINT NUMBER

FUN200 D P I→F	CONVERSION OF INTEGER TO FLOATING POINT NUMBER	FUN200 D P I→F
Symbol	※Because floating-point numbers occupy two registers, when using indirect addressing, it should be noted that odd-numbered registers cannot be used.	



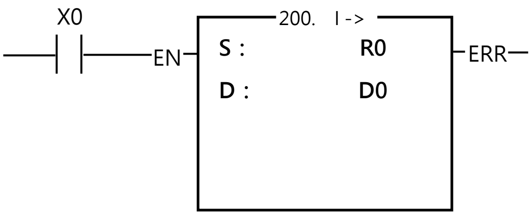
S: Starting register of Integer to be converted
D: Starting register to store the result of conversion
The register used by the operand must be an even address. For example, R8 is legal, but R7 is not.
S and D operands can be combined with V, Z, P0~P9 indicators for indirect addressing applications

Range / Operand	HR	ROR	DR	K	XR
	R0 | R34767	R43224 | R47319	D0 | D11999	16/32-bit +- numbers	V,Z P0-P9
S	○	○	○	○	○
D	○	○*	○*		○

Description

- The format of floating-point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System).
- When the execution control "EN"=1 or from 0→1 (P instruction), the integer value data in the S register is converted into floating-point format data, and then stored in the D register.

FUN200D P I→F	CONVERSION OF INTEGER TO FLOATING POINT NUMBER	FUN200D P I→F
Example		

Ladder diagram	ST
	Generate corresponding tags in the label first -> `IF X0 Then` `ItoF( S:= R0, D=> Tag_FlD0);` `END_IF`

- R0 = 200 → X0=
- After I > F Conversion → DD0 = 4348000H

I→F

R0 :

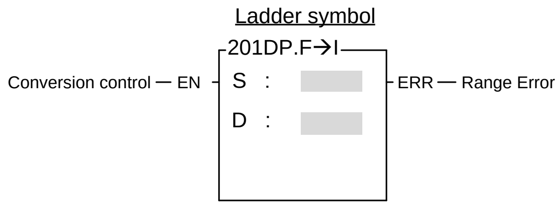
0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

DD0 :

0	1	0	0	0	0	1	1	0	1	0	0	1	0	0	0	0	00···0	0
b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16	b15	b14 ~ b1	b0
s	e	e	e	e	e	e	e	e	m	m	m	m	m	m	m	m	mm···m	m

7-24-2 CONVERSION OF FLOATING POINT NUMBER TO INTEGER

FUN201D P F→I	CONVERSION OF FLOATING POINT NUMBER TO INTEGER	FUN201D P F→I
Symbol	※Because floating-point numbers occupy two registers, when using indirect addressing, it should be noted that odd-numbered registers cannot be used.	



S: Starting register of Integer to be converted
D: Starting register to store the result of conversion
The register used by the operand must be an even address. For example, R8 is legal, but R7 is not.
S and D operands can be combined with V, Z, P0~P9 indicators for indirect addressing applications

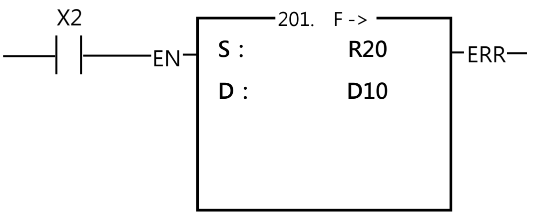
Range / Operand	HR	ROR	DR	XR
	R0 | R34767	R43224 | R47319	D0 | D11998	V,Z P0-P9
S	○	○	○	○
D	○	○*	○*	○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System)
- When the execution control "EN"=1 or from 0→1 (P command), the floating-point data in the S register is converted into integer format data and stored in the D register.
- If the value exceeds the valid range of destination, then do not carry out this instruction, and set the range-error flag “ERR” as 1 and the D register will be intact.

FUN201D P F→I	CONVERSION OF FLOATING POINT NUMBER TO INTEGER	FUN201D P F→I

Example

Ladder diagram	ST
	Generate corresponding tag in the label first -> `IF X2 Then` `FtoI( S:= Tag_FlR20, D=> D10);` `END_IF`


```
IF X2 Then
FtoI( S:= Tag_FlR20, D=> D10);
END_IF
```

- DR20 = 123.45 → X2=
- After F > I Conversion → D10 = 007BH

DR20:

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
0	1	0	0	0	0	1	0	1	1	1	1	0	1	1	0
s	e	e	e	e	e	e	e	e	m	m	m	m	m	m	m

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m

F → I

D10:

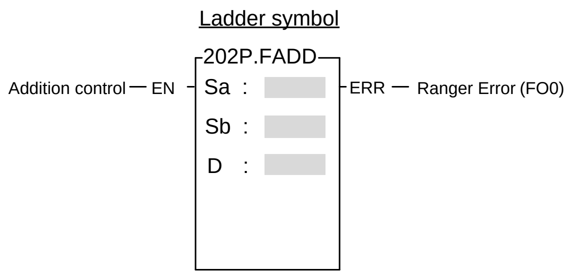
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1

7-24-3 FLOATING POINT NUMBER ADDITION

FUN202 P FADD	FLOATING POINT NUMBER ADDITION	FUN202 P FADD

Symbol

※Because floating-point numbers occupy two registers, when using indirect addressing, it should be noted that odd-numbered registers cannot be used.



Sa: Augend
Sb: Addend
D: Destination register to store the results of the addition
The register used by the operand must be an even address. For example, R8 is legal, but R7 is not.
Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect addressing

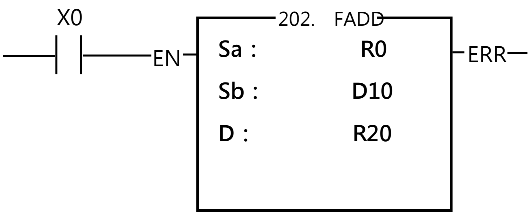
Range / Ope-rand	HR	ROR	DR	K	XR
	R0 | R34767	R43224 | R47319	D0 | D11999	floating point number	V,Z P0-P9
Sa	○	○	○	○	○
Sb	○	○	○	○	○
D	○	○*	○*		○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3-P. 118 (Numbering System).
- When addition control "EN"=1 or from 0→1 (P instruction), perform floating-point addition operation on Sa and Sb and write the result into D. If the execution result exceeds the expressible range of floating point numbers (+-3.4*10 38)", the error flag "ERR" is set to 1, and the value of the D register is an invalid value, which should be ignored.

FUN202 P FADD	FLOATING POINT NUMBER ADDITION	FUN202 P FADD
Example		

Ladder diagram	ST

	Generate corresponding tag in the label first
	->
	`IF X0 Then`
	`Tag_FlR20 := Tag_FlD10 + Tag_FlR0 ;`
	`END_IF`

- When X0=ON, performs the addition of the data specified at Sa and Sb:

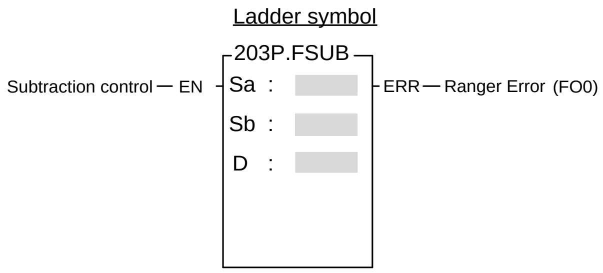
DR0	2 0 0	⇨ Floating Point Number :	DR0	4 3 4 8 0 0 0 0 H
DR10	1 5 0	⇨ Floating Point Number :	DR10	4 3 1 6 0 0 0 0 H
		+		
			DR20	4 3 A F 0 0 0 0 H

7-24-4 FLOATING POINT NUMBER SUBTRACTION

FUN 203 P FSUB	FLOATING POINT NUMBER SUBTRACTION	FUN 203 P FSUB

Symbol

※Because floating-point numbers occupy two registers, when using indirect addressing, it should be noted that odd-numbered registers cannot be used.



Sa: Minuend
Sb: Subtrahend
D: Destination register to store the results of the subtraction
The register used by the operand must be an even address. For example, R8 is legal, but R7 is not.
Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect addressing.

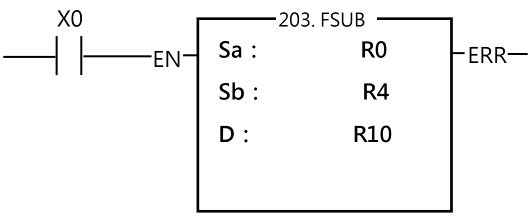
Range / Operand	HR R0 | R34767	ROR R43224 | R47319	DR D0 | D11999	K floating point number	XR V,Z P0-P9
Sa	○	○	○	○	○
Sb	○	○	○	○	○
D	○	○*	○*		○

Description

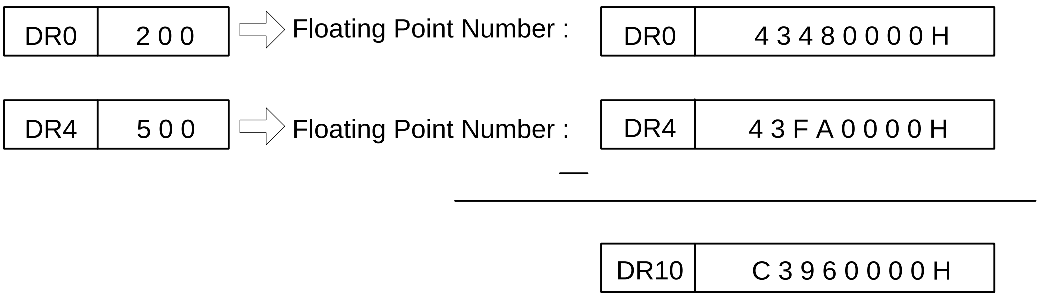
- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System).
- When addition control "EN"=1 or from 0→1 (P instruction), perform floating-point addition operation on Sa and Sb and write the result into D. If the execution result exceeds the expressible range of floating point numbers (+-3.4*10 38)", the error flag "ERR" is set to 1, and the value of the D register is an invalid value, which should be ignored.

FUN 203 P FSUB	FLOATING POINT NUMBER SUBTRACTION	FUN 203 P FSUB
Example		

Ladder diagram	ST

	Generate corresponding tag in the label first
	->
	`IF X0 Then`
	`Tag_FlR10 := Tag_FlR0 - Tag_FlR4 ;`
	`END_IF`

- When X0=ON, performs the subtraction of the data specified at Sa and Sb

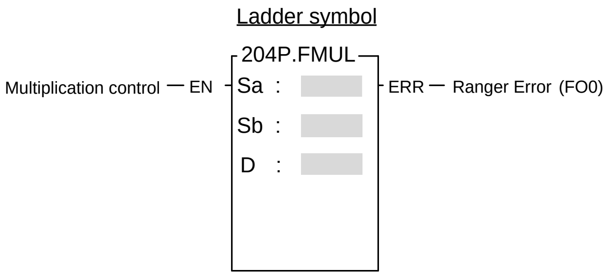


7-24-5 FLOATING POINT NUMBER MULTIPLICATION

FUN 204 P FMUL	FLOATING POINT NUMBER MULTIPLICATION	FUN 204 P FMUL

Symbol

※Because floating-point numbers occupy two registers, when using indirect addressing, it should be noted that odd-numbered registers cannot be used.



Sa: Multiplicand
Sb: Multiplier
D: Destination register to store the results of the multiplication
The register used by the operand must be an even address. For example, R8 is legal, but R7 is not.
Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect addressing

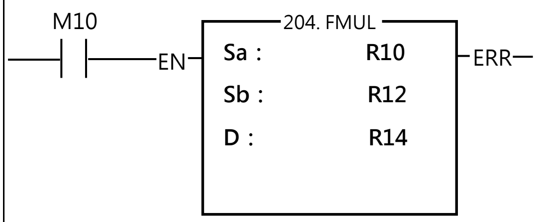
Range / Operand	HR	ROR	DR	K	XR
	R0 | R34767	R43224 | R47319	D0 | D11999	floating point number	V,Z P0-P9
Sa	○	○	○	○	○
Sb	○	○	○	○	○
D	○	○*	○*		○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System)-P.118.
- When addition control "EN"=1 or from 0→1 (P instruction), perform floating-point addition operation on Sa and Sb and write the result into D. If the execution result exceeds the expressible range of floating point numbers (+-3.4*10 38)", the error flag "ERR" is set to 1, and the value of the D register is an invalid value, which should be ignored.

FUN 204 P FMUL	FLOATING POINT NUMBER MULTIPLICATION	FUN 204 P FMUL
Example		

Ladder diagram



ST

Generate corresponding tag in the label first

->

```
IF M10 Then
Tag_FlR14 := Tag_FlR10 * Tag_FlR12 ;
END_IF
```

OR

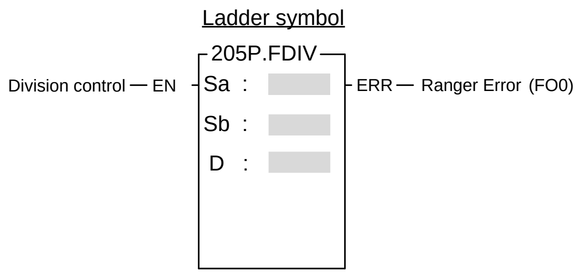
```
IF M10 Then
FMUL( Sa:= Tag_FlR10, Sb:=
Tag_FlR12, D:= Tag_FlR14);
END_IF
```

When M10= Performs the multiplication of the data specified at Sa and Sb :

DR10	1 2 3 . 4 5	⇨ Floating Point Number :	DR10	4 2 F 6 E 6 6 6 H
DR12	6 7 8 . 5 4	⇨ Floating Point Number :	DR12	4 4 2 9 A 2 8 F H
		×		
			DR14	4 7 A 3 9 A E 2 H

7-24-6 FLOATING POINT NUMBER DIVIDION

FUN 205 P FDIV	FLOATING POINT NUMBER DIVIDION	FUN 205 P FDIV
Symbol	※Because floating-point numbers occupy two registers, when using indirect addressing, it should be noted that odd-numbered registers cannot be used.	



Sa: Dividend
Sb: Divisor
D: Destination register to store the results of the division
The register used by the operand must be an even address. For example, R8 is legal, but R7 is not.
Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect addressing

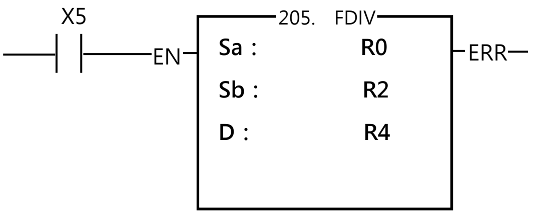
Range / Ope-rand	HR	ROR	DR	K	XR
	R0 | R34767	R43224 | R47319	D0 | D11999	floating point number	V,Z P0-P9
Sa	○	○	○	○	○
Sb	○	○	○	○	○
D	○	○*	○*		○

Description	

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System)-P.118.
- When addition control "EN"=1 or from 0→1 (P instruction), perform floating-point addition operation on Sa and Sb and write the result into D. If the execution result exceeds the expressible range of floating point numbers (+-3.4*10 38)", the error flag "ERR" is set to 1, and the value of the D register is an invalid value, which should be ignored.

FUN 205 P FDIV	FLOATING POINT NUMBER DIVIDION	FUN 205 P FDIV
Example		

Ladder diagram	ST

	Generate corresponding tag in the label first
	->
	`IF X5 Then`
	`Tag_FlR4 := Tag_FlR0 / Tag_FlR2 ;`
	`END_IF`
	
	OR
	
	`IF X5 Then`
	`FDIV( Sa:= Tag_FlR0, Sb:= Tag_FlR2,`
	`D:= Tag_FlR4);`
	`END_IF`

- When X5=ON, performs the division of the data specified at Sa and Sb :

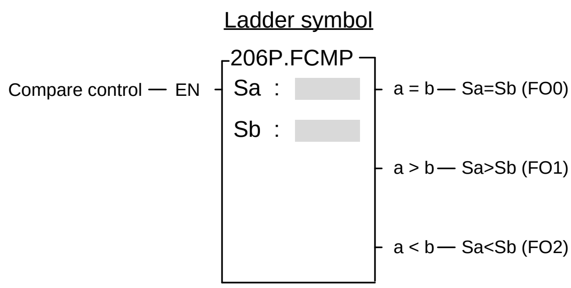
DR0	1 2 5 . 2 5	⇨ Floating Point Number :	DR0	4 2 F A 8 0 0 0 H
DR2	5	⇨ Floating Point Number :	DR2	4 0 A 0 0 0 0 0 H

÷

DR4	4 1 C 8 6 6 6 6 H

7-24-7 FLOATING POINT NUMBER COMPARE

FUN 206 P FCMP	FLOATING POINT NUMBER COMPARE	FUN 206 P FCMP
Symbol	※Because floating-point numbers occupy two registers, when using indirect addressing, it should be noted that odd-numbered registers cannot be used.	



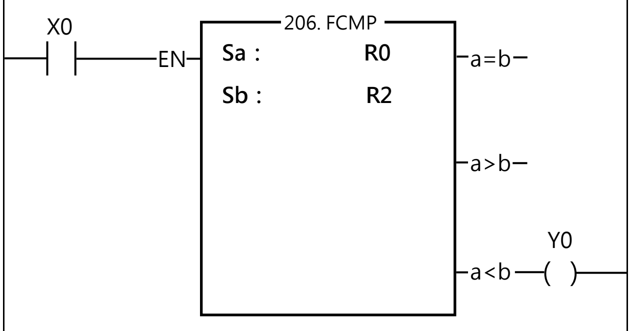
Sa: The register to be compared
Sb: The register to be compared
The register used by the operand must be an even address. For example, R8 is legal, but R7 is not.
Sa, Sb may combine with V, Z, P0~P9 to serve indirect addressing.

Range / Ope-rand	HR	ROR	DR	K	XR
	R0 | R34767	R43224 | R47319	D0 | D11999	floating point number	V,Z P0-P9
Sa	○	○	○	○	○
Sb	○	○	○	○	○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System)-P.118.
- Compares the data of Sa and Sb when the compare control input "EN" =1 or from 0 to 1 (P instruction). If the data of Sa is equal to Sb, then set FO0 to 1. If the data of Sa>Sb, then set FO1 to 1. If the data of Sa<Sb, then set FO2 to 1. If the data of Sa < Sb, then set the FO2 to 1.

FUN 206 P FCMP	FLOATING POINT NUMBER COMPARE	FUN 206 P FCMP
Example		

Ladder diagram	ST
	

- When X0=ON, compares the data of Sa and Sb:

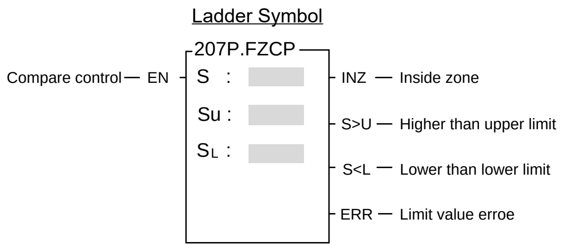
DR0	2 0 0 . 1	⇨ Floating Point Number :	DR0	4 3 4 8 1 9 9 A H
DR2	2 0 0 . 2	⇨ Floating Point Number :	DR2	4 3 4 8 3 3 3 3 H

- From the above example, we first assume the data of DR0 is 200.1 and DR2 is 200.2, compare the data when X0 =1 by executing the CMP instruction. The FO0 and FO1 are set to 0 and FO2 (a<b) is set to 1 since a<b.
- If you want to have the compound results, such as ≧、≦、< > etc., please send =、< and > results to relay first and then combine the result from the relays.

7-24-8 FLOATING POINT NUMBER ZONE COMPARE

FUN 207 P FZCP	FLOATING POINT NUMBER ZONE COMPARE	FUN 207 P FZCP

Symbol	※Because floating-point numbers occupy two registers, when using indirect addressing, it should be noted that odd-numbered registers cannot be used.



S: Register for zone comparison
SU: The upper limit value
SL: The lower limit value
The register used by the operand must be an even address. For example, R8 is legal, but R7 is not.
S, SU, SL may combine with V, Z, P0~P9 to serve indirect address application

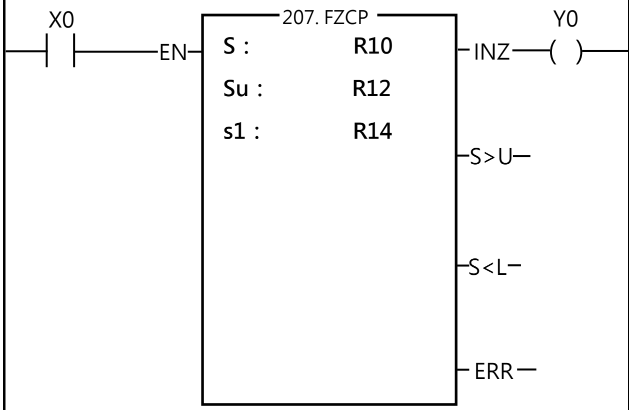
Range / Ope-rand	HR R0 | R34767	ROR R43224 | R47319	DR D0 | D11999	K floating point number	XR V,Z P0-P9
S	○	○	○		○
SU	○	○	○	○	○
SL	○	○	○	○	○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System)-P.118.
- When compare control "EN" = 1 or changes from 0 to 1 (P instruction), compares S with upper limit SU and lower limit SL. If S is between the upper limit and the lower limit (SL≦S≦SU), then set the inside zone flag "INZ" to 1. If the value of S is greater than the upper limit SU, then set the higher than upper limit flag "S>U" to 1. If the value of S is smaller then the lower limit SL, then set the lower than lower limit flag "S<L" as 1.
- The upper limit SU should be greater than the lower limit SL. If SU<SL, then the limit value error flag "ERR" will set to 1, and this instruction will not carry out.

FUN 207 P FZCP	FLOATING POINT NUMBER ZONE COMPARE	FUN 207 P FZCP
Example		

Ladder diagram	ST




```
IF X0 THEN
FZCP( S:= R10, Su:= R12, Sl:= R14,
INZ=> Y0, S_Gt_U=> M0, S_Less_L=> M1,
ERR=> M2);
END_IF
```

- The instruction compares the value of DR10 with the upper and lower limit zones formed by DR12 and DR14. If the values of DR10~DR14 are as shown in the diagram at bottom left, then the result can then be obtained as at the right of this diagram.
- If want to get the status of out side the zone, then OUT NOT Y0 may be used, or an OR operation between the two outputs S>U and S<L may be carried out, and move the result to Y0.

S	DR10	2 0 0 0 . 2	⇨ Floating Point Number :	DR10	4 4 F A 0 6 6 6 H	
Su	DR12	3 0 0 0 . 3	⇨ Floating Point Number :	DR12	4 5 3 B 8 4 C D H	(Upper limit value)
SL	DR14	1 0 0 0 . 1	⇨ Floating Point Number :	DR14	4 4 7 A 0 6 6 6 H	(Lower limit value)

Before-execution

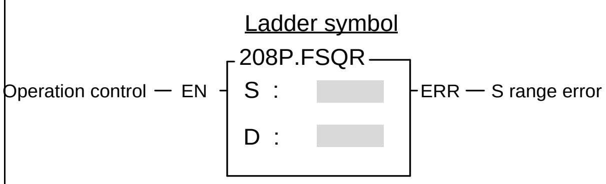
X0 = ↑ → FLOATING ZONE COMPARE → Y0 = $\boxed{1}$

Results of execution

7-24-9 FLOATING POINT NUMBER SQUARE ROOT

FUN 208 P FSQR	FLOATING POINT NUMBER SQUARE ROOT	FUN 208 P FSQR

Symbol	※Because floating-point numbers occupy two registers, when using indirect addressing, it should be noted that odd-numbered registers cannot be used.



S: Source register to be taken square root
D: Register for storing result
(Square root value)
The register used by the operand must be an even address. For example, R8 is legal, but R7 is not.
S, D may combine with V, Z, P0~P9 to serve indirect address application

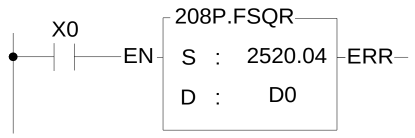
Range / Operand	HR	ROR	DR	K	XR
	R0 | R34767	R43224 | R47319	D0 | D11999	floating point number	V,Z P0-P9
S	○	○	○	○	○
D	○	○*	○*		○

Description

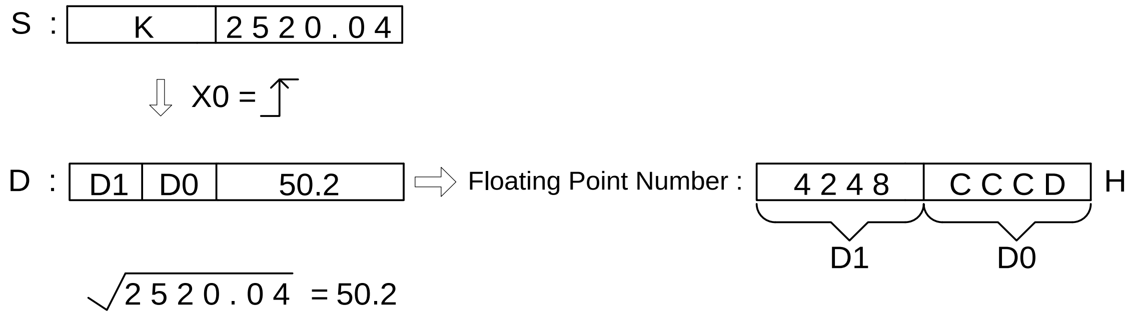
- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System)-P.118.
- When the operation control "EN"=1 or from 0→1 (P instruction), take the square root value of the S value or the content value of the temporary register designated by S and store it in the temporary register designated by D.
- If the value of S is negative, then the error flag "ERR" will be set to 1, and do not execute the operation.

FUN 208 P FSQR	FLOATING POINT NUMBER SQUARE ROOT	FUN 208 P FSQR

Example

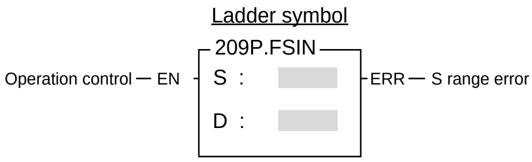
<table>
<tr><th>Ladder diagram</th><th>ST</th></tr>
<tr><td>
</td><td>Generate corresponding tag in the label first
->
<code>IF R_TRIG(S:= X0) Then</code>
<code>FSQR(S:= 2520.04, D:=Tag_FlD0);</code>
<code>END_IF</code></td></tr>
</table>

When X0=↑, take the square root of S :



7-24-10 SIN TRIGONOMETRIC INSTRUCTION

FUN 209 P FSIN	SIN TRIGONOMETRIC INSTRUCTION	FUN 209 P FSIN
Symbol	※Because floating-point numbers occupy two registers, when using indirect addressing, it should be noted that odd-numbered registers cannot be used.	



S: Source register to be taken SIN
D: Register for storing result
(SIN value)
The register used by the operand must be an even address. For example, R8 is legal, but R7 is not.
S, D may combine with V, Z, P0~P9 to serve indirect address application.

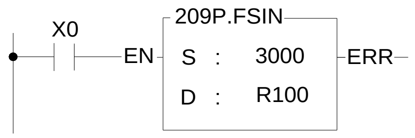
Range / Operand	HR R0 | R34767	ROR R43224 | R47319	DR D0 | D11999	K 16 – bit Integer	XR V,Z P0-P9
S	○	○	○	○	○
D	○	○*	○*		○

Description

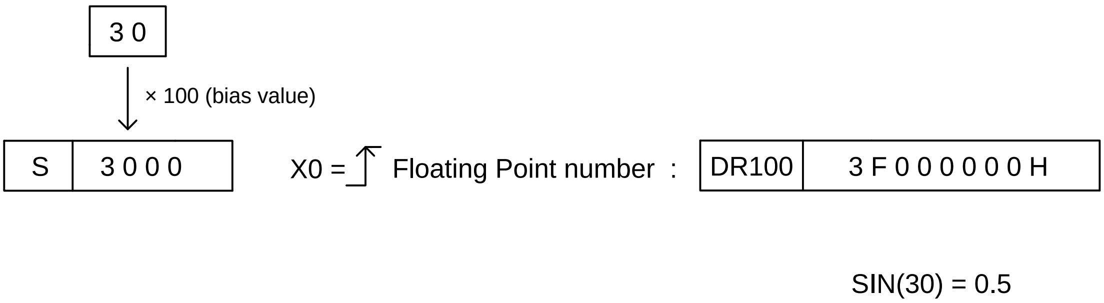
- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System)-P.118.
- When operation control "EN" = 1 or from 0 to 1 (P instruction), take the SIN value of the angle data specified by the S register and store the result into the register D~D+1 in floating point number format. The valid range of the angle is from –18000 to +18000, unit in 0.01 degree.
- If the S value is not within the valid range, then the S value error flag "ERR" will be set to 1, and do not execute the operation.

FUN 209 P FSIN	SIN TRIGONOMETRIC INSTRUCTION	FUN 209 P FSIN
Example		

Ladder diagram	ST

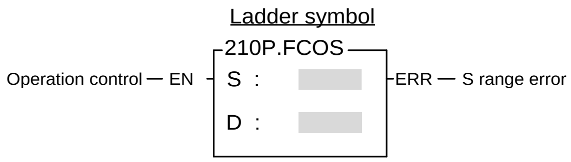
	Generate corresponding tag in the label first
	->
	`IF R_TRIG( S:= X0 ) Then`
	`FSIN( S:= 3000, D:= Tag_FlR100);`
	`END_IF`

- In the example above, when X0=ON, store the value of $SIN\angle 30$ in DR100.



7-24-11 COS TRIGONOMETRIC INSTRUCTION

FUN 210 P FCOS	COS TRIGONOMETRIC INSTRUCTION	FUN 210 P FCOS
Symbol	※Because floating-point numbers occupy two registers, when using indirect addressing, it should be noted that odd-numbered registers cannot be used.	



S: Source register to be taken COS
D: Register for storing result (COS value)
The register used by the operand must be an even address. For example, R8 is legal, but R7 is not.
S, D may combine with V, Z, P0~P9 to serve indirect address application

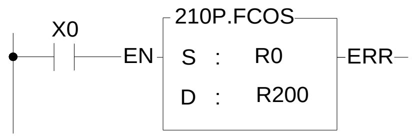
Range / Operand	HR	ROR	DR	K	XR
	R0 | R34767	R43224 | R47319	D0 | D11999	16 – bit Integer	V,Z P0-P9
S	○	○	○	○	○
D	○	○*	○*		○

Description

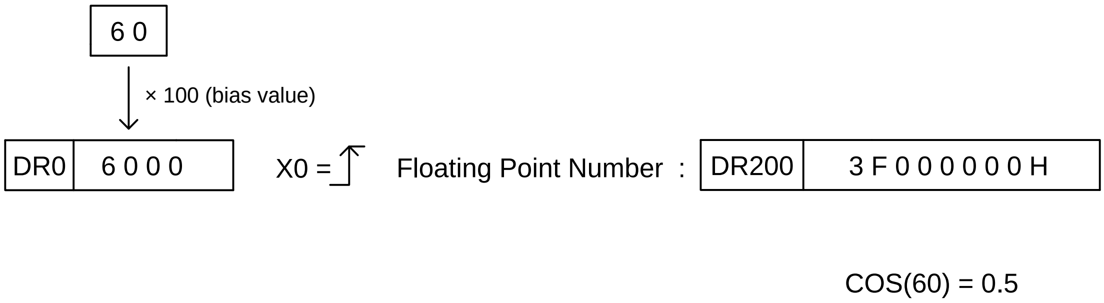
- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System).
- When operation control "EN" = 1 or from 0 to 1 (P instruction), take the COS value of the angle data specified by the S register and store the result into the register D~D+1 in floating point number format. The valid range of the angle is from –18000 to +18000, unit in 0.01 degree.
- If the S value is not within the valid range, then the S value error flag "ERR" will be set to 1, and do not execute the operation.

FUN 210 P FCOS	COS TRIGONOMETRIC INSTRUCTION	FUN 210 P FCOS
Example		

Ladder diagram	ST

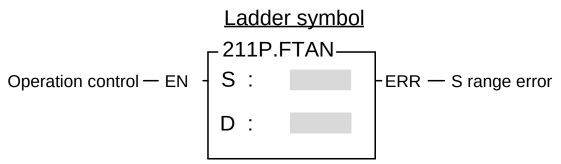
	Generate corresponding tag in the label first
	->
	`IF R_TRIG( S:= X0 ) Then`
	`FCOS( S:= R0, D:= Tag_FlR200);`
	`END_IF`

- In the example above, when X0=ON, store the value of $\mathrm{COS}\angle 60$ in DR200.



7-24-12 TAN TRIGONOMETRIC INSTRUCTION

FUN 211 P FTAN	TAN TRIGONOMETRIC INSTRUCTION	FUN 211 P FTAN
Symbol	※Because floating-point numbers occupy two registers, when using indirect addressing, it should be noted that odd-numbered registers cannot be used.	



S: Source register to be taken TAN
D: Register for storing result (TAN value)
The register used by the operand must be an even address. For example, R8 is legal, but R7 is not.
S, D may combine with V, Z, P0~P9 to serve indirect address application

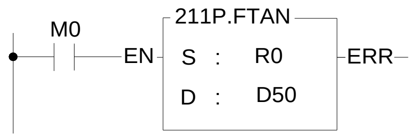
Range / Operand	HR R0 | R34767	ROR R43224 | R47319	DR D0 | D11999	K 16 – bit Integer	XR V,Z P0-P9
S	○	○	○	○	○
D	○	○*	○*		○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System).
- When the operation control "EN"=1 or from 0→1 (P instruction), the S value or the content value of the temporary register designated by S is taken from the TAN function and stored in the temporary register designated by D. The effective range of S is -18000 ~ +18000, the unit is 0.01 degree.
- If the S value is not within the valid range, then the S value error flag "ERR" will be set to 1, and do not execute the operation.

FUN 211 P FTAN	TAN TRIGONOMETRIC INSTRUCTION	FUN 211 P FTAN
Example		

Ladder diagram	ST

	Generate corresponding tag in the label first
	->
	`IF R_TRIG( S:= M0 ) Then`
	`FTAN( S:= R0, D:= Tag_FlD50);`
	`END_IF`

- In the example above, when M0= store the value of TAN∠45 into DD50.

4 5

× 100 (bias value)

DR0	4 5 0 0

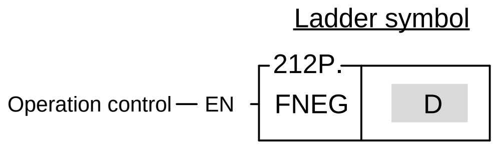
M0 = ↑ Floating Point Number ：

DD50	3 F 8 0 0 0 0 0 H

TAN(45) = 1

7-24-13 CHANGE SIGN OF THE FLOATING POINT NUMBER

FUN 212 P FNEG	CHANGE SIGN OF THE FLOATING POINT NUMBER	FUN 212 P FNEG
Symbol	※Because floating-point numbers occupy two registers, when using indirect addressing, it should be noted that odd-numbered registers cannot be used.	



D: Register to be changed sign
The register used by the operand must be an even address. For example, R8 is legal, but R7 is not.
D may combine with V, Z, P0~P9 to serve indirect address application

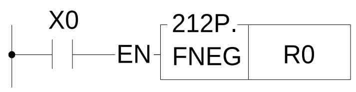
Range / Operand	HR	ROR	DR	XR
	R0 | R34767	R43224 | R47319	D0 | D11999	V,Z P0-P9
D	○	○	○	○

Description

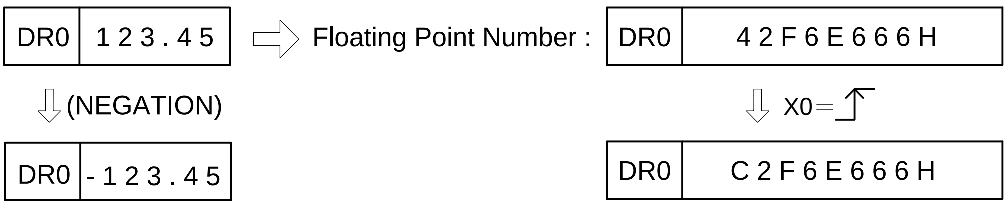
- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System). 。
- When operation control "EN" = 1 or from 0 to 1 (P instruction), the sign of the floating point number register specified by D will be toogled.
- If value of D was originally negative, the result of taking a negative number will become a positive number.

FUN 212 P FNEG	CHANGE SIGN OF THE FLOATING POINT NUMBER	FUN 212 P FNEG
Example		

Ladder diagram	ST

	Generate corresponding tag in the label first
	->
	`IF R_TRIG( S:= X0 ) Then`
	`Tag_FlR0 := - Tag_FlR0;`
	`END_IF`

The instruction at left negates the value of the DR0 register, and stores it back to DR0.

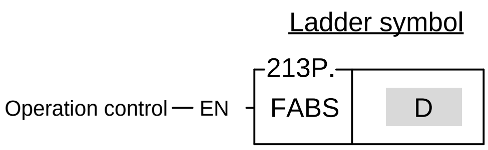


7-24-14 FLOATING POINT NUMBER ABSOLUTE VALUE

FUN 213 P FABS	FLOATING POINT NUMBER ABSOLUTE VALUE	FUN 213 P FABS

Symbol

※Because floating-point numbers occupy two registers, when using indirect addressing, it should be noted that odd-numbered registers cannot be used.



D: Register to be taken absolute value
The register used by the operand must be an even address. For example, R8 is legal, but R7 is not.
D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	HR	ROR	DR	XR
	R0 | R34767	R43224 | R47319	D0 | D11999	V,Z P0-P9
D	○	○	○	○

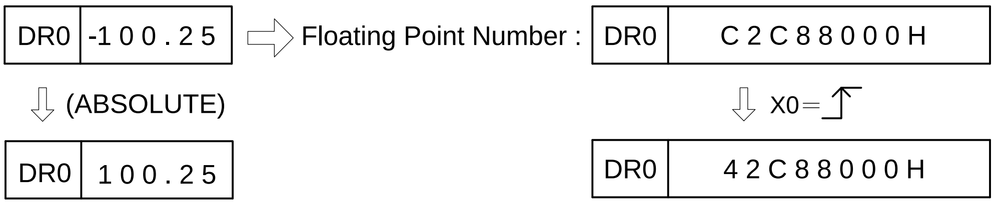
Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System).
- When operation control "EN" = 1 or from 0 to 1 (P instruction), calculate the absolute value of the floating point number register specified by D, and write it back into the original D register.

FUN 213 P FABS	FLOATING POINT NUMBER ABSOLUTE VALUE	FUN 213 P FABS
Example		

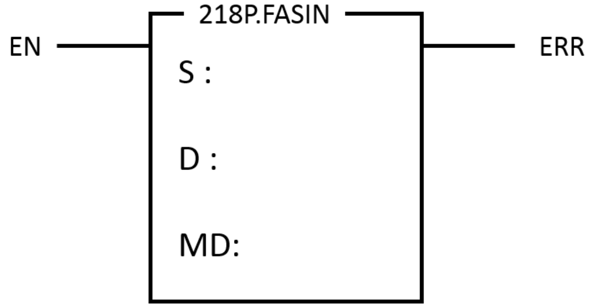
Ladder diagram	ST
 	

- This instruction calculates the absolute value of the DR0 register, and stores it back in DR0.



7-24-15 FLOATING POINT ARC SINE FUNCTION

FUN 218 P FASIN	FLOATING POINT ARC SINE FUNCTION	FUN 218 P FASIN
Symbol	※Because floating-point numbers occupy two registers, when using indirect addressing, it should be noted that odd-numbered registers cannot be used.	



S: Source data or register to be calculated the arc sine value.
The register used by the operand must be an even address. For example, R8 is legal, but R7 is not.
D: Register for storing the result.
S, D may combine with V, Z, P0~P9 to serve indirect address application.
MD: In order to make the user more intuitive in use, MD can choose the output mode:
MD is 0: the output value is the radius, and the output is a floating point number (32bit).
MD is 1: the output value is an angle, and the output is a positive integer (16bit).

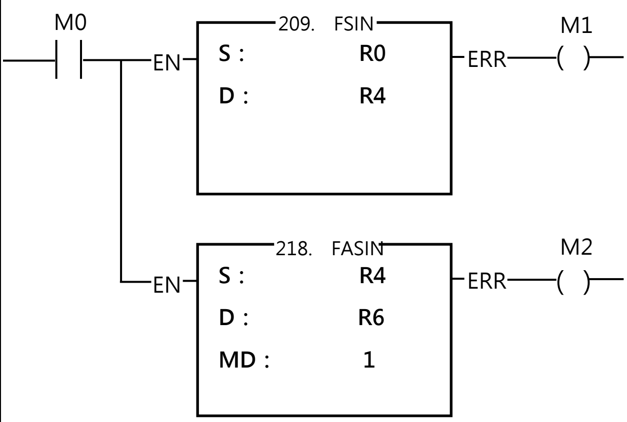
Range / Operand	HR	ROR	DR	K	XR
	R0 | R34767	R43224 | R47319	D0 | D11999	floating point number	V,Z P0-P9
S	○	○	○	○	○
D	○	○*	○*		○
MD				0,1	

Description	

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System).
- When the operation control "EN"=1 or from 0→1 (P instruction), the S value or the temporary register content value designated by S takes the arc sine function value (unit is Radian) and stores it in D specified register.
- Range of S data: -1~ +1; range of D value: -π/2 ~ π/2 (Unit in radian)
- If the value of S exceeds the valid range, or the indirect addressing is wrong, the error flag "ERR" is set to 1, and the contents of the register designated by D will not be updated.
- All floating point instructions can’t be executed in interrupt service routine.

FUN 218 P FASIN	FLOATING POINT ARC SINE FUNCTION, $\sin^{-1}$	FUN 218 P FASIN
Example		

Ladder diagram



ST

Generate corresponding tag in the label first ->

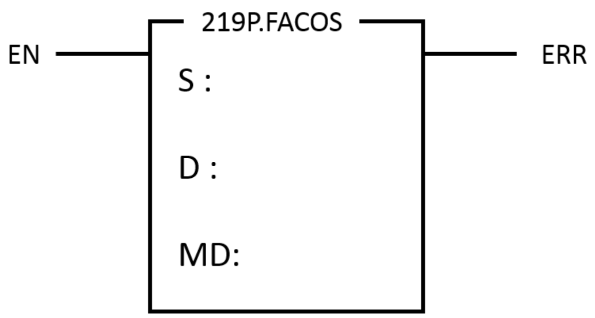
```
IF M0 Then
FSIN( S:= R0, D:= Tag_FlR4);
FASIN_deg( S:= Tag_FlR4, D:= R6);
END_IF
```

- When M0 = 1, calculate the arc sine value of DR4, then store the degree (MD=1) to DR6.

Name	Status	Data	Comment
DR0	DEC	30	[R0]
DR4	FLOAT	0.005235963	[R4]
DR6	DEC	30	[R6]

7-24-16 FLOATING POINT ARC COSINE FUNCTION

FUN 219 P FACOS	FLOATING POINT ARC COSINE FUNTION	FUN 219 P FACOS
Symbol	※Because floating-point numbers occupy two temporary registers, when using indirect addressing, it should be noted that odd-numbered temporary registers cannot be used.	



S: Source data or register to be calculated the arc cosine value.
The register used by the operand must be an even address. For example, R8 is legal, but R7 is not.
D: Register for storing the result.
S, D may combine with V, Z, P0~P9 to serve indirect address application.
MD: In order to make the user more intuitive in use, MD can choose the output mode:
MD is 0: the output value is the radius, and the output is a floating point number (32bit).
MD is 1: the output value is an angle, and the output is a positive integer (16bit).

Range / Ope-rand	HR	ROR	DR	K	XR
	R0 | R34767	R43224 | R47319	D0 | D11999	floating point number	V,Z P0-P9
S	○	○	○	○	○
D	○	○*	○*		○
MD				0,1	

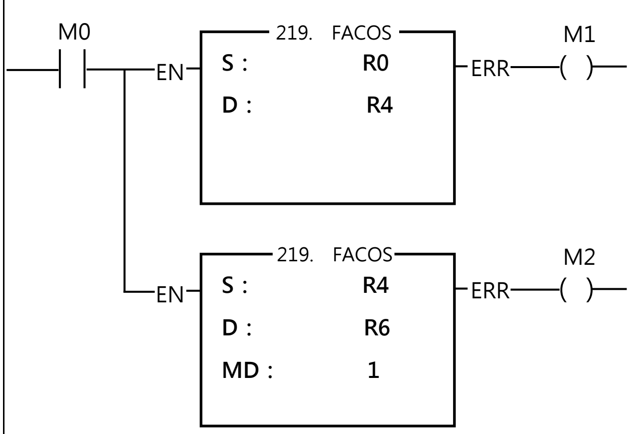
Description	

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System).
- When the operation control "EN"=1 or from 0→1 (P instruction), the S value or the temporary register content value designated by S takes the arc cosine function value (unit is Radian) and stores it in D specified register.
- Range of S data: -1~ +1; range of D value: -π/2 ~ π/2 (Unit in radian)
- If the value of S exceeds the valid range, or the indirect addressing is wrong, the error flag "ERR" is set to 1, and the contents of the register designated by D will not be updated.
- All floating point instructions can't be executed in interrupt service routine.

FUN 219 P FACOS	FLOATING POINT ARC COSINE VALUE	FUN 219 P FACOS
Example		

- When M0 = 1, calculate the arc cosine value of DR4, then store the degree (MD=1) to DR6.

Ladder diagram	ST

	Generate corresponding tag in the label first
	->
	`IF M0 Then`
	`FCOS( S:= R0, D:= Tag_FlR4);`
	`FACOS_deg( S:= Tag_FlR4, D:= R6);`
	`END_IF`

Name	Status	Data	Comment
DR0	DEC	30	[R0]
DR4	FLOAT	0.99998629	[R4]
DR6	DEC	30	[R6]

8

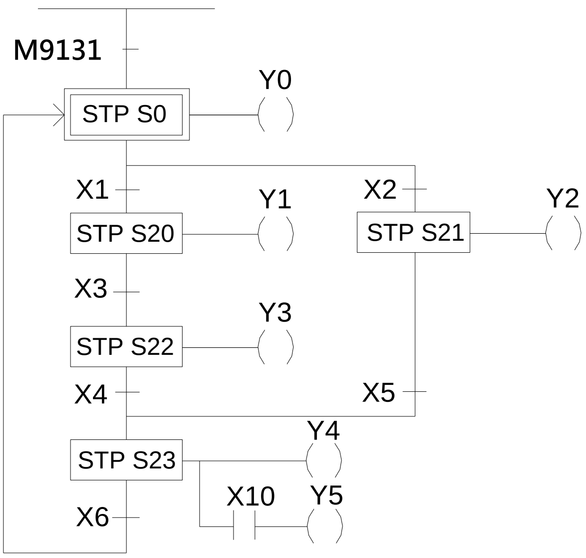
Step Instruction Description

Structured programming design is a major trend in software design. The benefits are high readability, easy maintenance, convenient updating and high quality and reliability. For the control applications, consisted of many sequential tasks, designed by conventional ladder program design methodology usually makes others hard to maintain. Therefore, it is necessary to combine the current widely used ladder diagrams with the sequential controls made especially for machine working flow. With help from step instructions, the design work will become more efficient, time saving and controlled. This kind of design method that combines process control and ladder diagram together is called the step ladder language.

The basic unit of step ladder diagram is a step. A step is equivalent to a movement (step) in the machine operation where each movement has an output. The complete machine or the overall sequential control process is the combination of steps in serial or parallel. Its step-by-step sequential execution procedure allows others to be able to understand the machine operations thoroughly, so that design, operation, and maintenance will become more effective and simpler.

8-1 The Operation Principle of Step Ladder Diagram

【Example】



【Description】

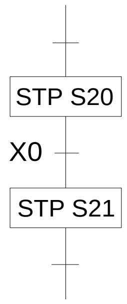
1. STP Sxxxx is the symbol representing a step Sxxxx that can be one of S0 ~ S3103. When executing the step (status ON), the ladder diagram on the right will be executed and the previous step and output will become OFF.
2. M9131 is on for a scan time after program start. Hence, as soon as ON, the stop of the initial step S0 is entered (S0 ON) while the other steps are kept inactive, i.e. Y1 ~ Y5 are all OFF. This means M9131 ON→S0 ON→Y0 ON and Y0 will remain ON until one of the contacts X1 or X2 is ON.
3. Assume that X2 is ON first; the path to S21 will then be executed.
   X2 ON⇒ $\begin{cases}\text{S21 ON}\\ \text{S0 OFF}\end{cases}$ ⇒ $\begin{cases}\text{Y2 ON}\\ \text{Y0 OFF}\end{cases}$
   Y2 will remain ON until X5 is ON.
4. Assume that X5 is ON, the process will move forward to step S23.
   i.e. X5 ON⇒ $\begin{cases}\text{S23 ON}\\ \text{S21 OFF}\end{cases}$ ⇒ $\begin{cases}\text{Y4 ON}\\ \text{Y2 OFF}\end{cases}$
   Y4 and Y5 will remain ON until X6 is ON.

   ※If X10 is ON, then Y5 will be ON.。

5. Assume that X6 is ON, the process will move forward to S0.
   i.e. X6 ON⇒ $\begin{cases}\text{S0 ON}\\ \text{S23 OFF}\end{cases}$ ⇒ $\begin{cases}\text{Y0 ON}\\ \text{Y4,Y5 OFF}\end{cases}$
   Then, a control process cycle is completed and the next control process cycle is entered.

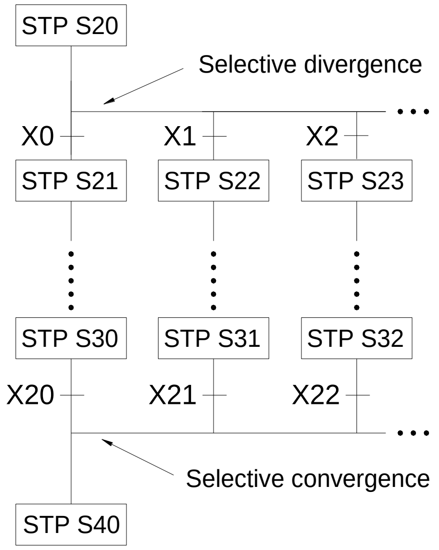
8-2 Basic Formation of Step Ladder Diagram

①Single path



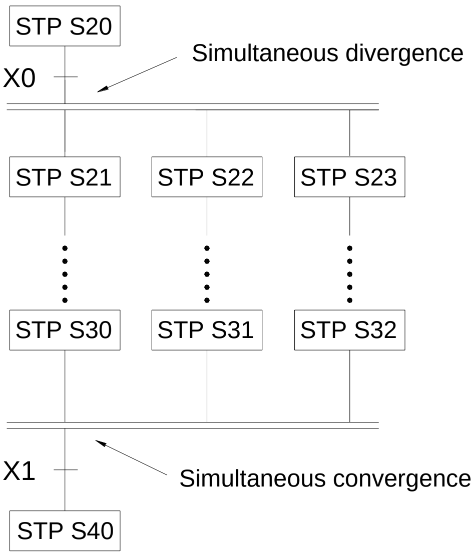
- Step S20 alone moves to step S21 through X0.
- X0 can be changed to other serial or parallel combination of contacts.

②Selective divergence/convergence



- Step S20 selects an only one path which divergent condition first met. E.g. X2 is ON first, then only the path of step S23 will be executed.
- A divergence may have up to 8 paths maximum.
- X1, X2, ….., X22 can all be replaced by the serial or parallel combination of other contacts.

③Simultaneous divergence/convergence

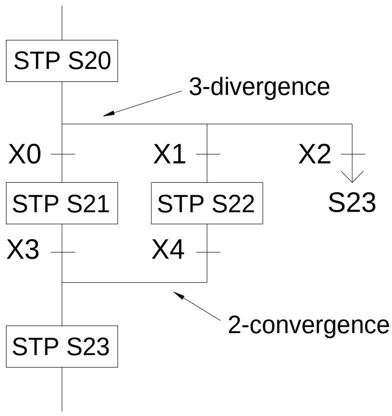


After X0 is ON, step S20 will simultaneously execute all paths below it, i.e. all S21, S22, S23, and so on, are in action.

- All divergent paths at a convergent point will be executed to the last step (e.g. S30, S31 and S32). When X1 is ON, they can then transfer to S40 for execution.
- The number of divergent paths must be the same as the number of convergent paths. The maximum number of divergence/convergence path is 8.

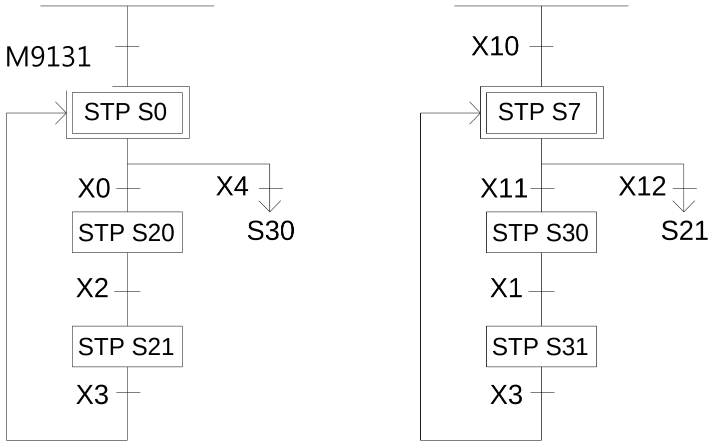
④Jump

a. The same step loop



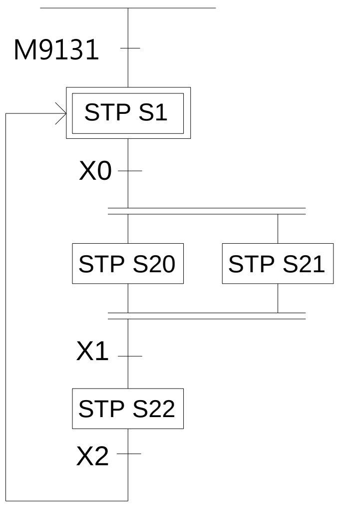
- There are 3 paths below step S20 as shown on the left. Assume that X2 is ON, then the process can jump directly to step S23 to execute without going through the process of selective convergence. 。
- The execution of simultaneous divergent paths can not be skipped.

b．Different step loop

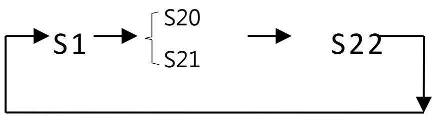


⑤Close Loop and Single Cycle

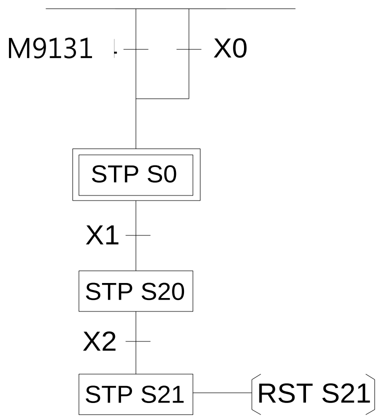
a．Close Loop



- The initial step S1 is ON, endless cycle will be continued afterwards.

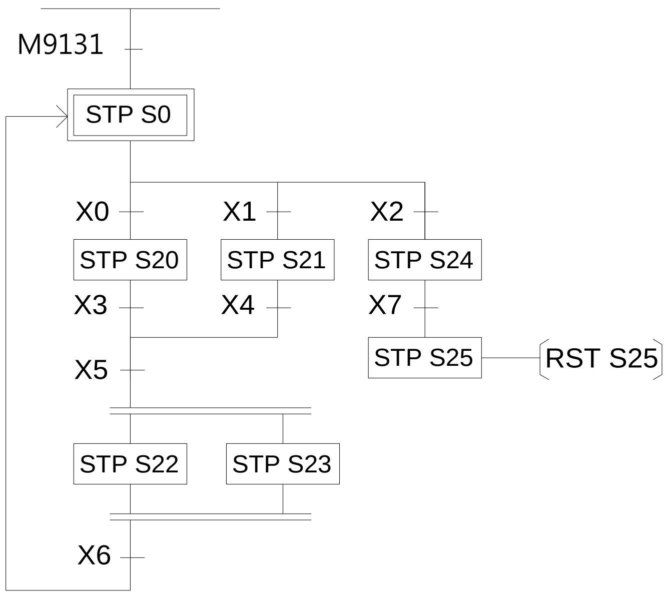


b. Single Cycle

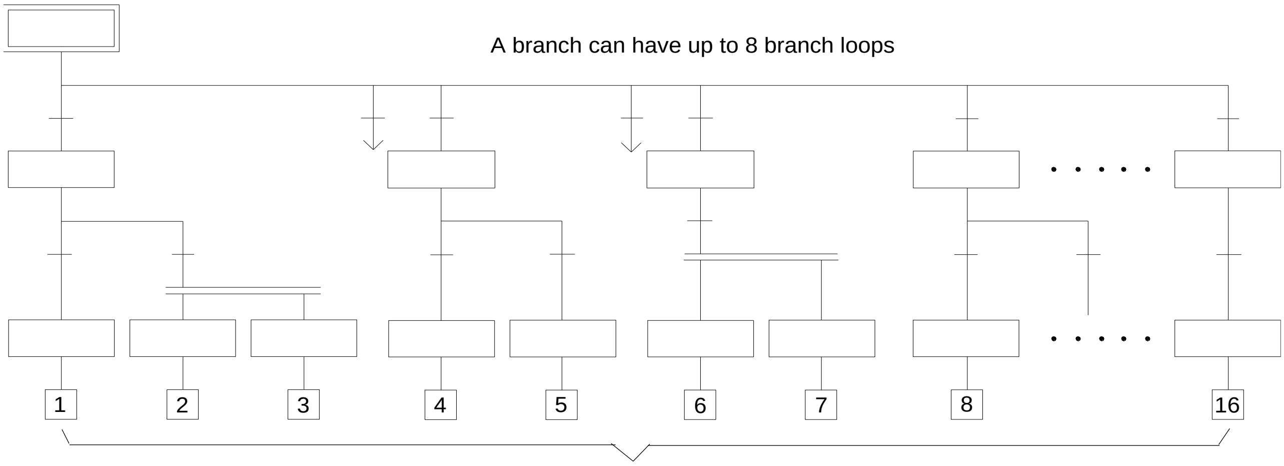


- When step S20 is ON, if X2 is also ON, then“RST S21”instruction will let S21 OFF which will stop the whole step process.

c. Mixed Process



⑥Combined Application



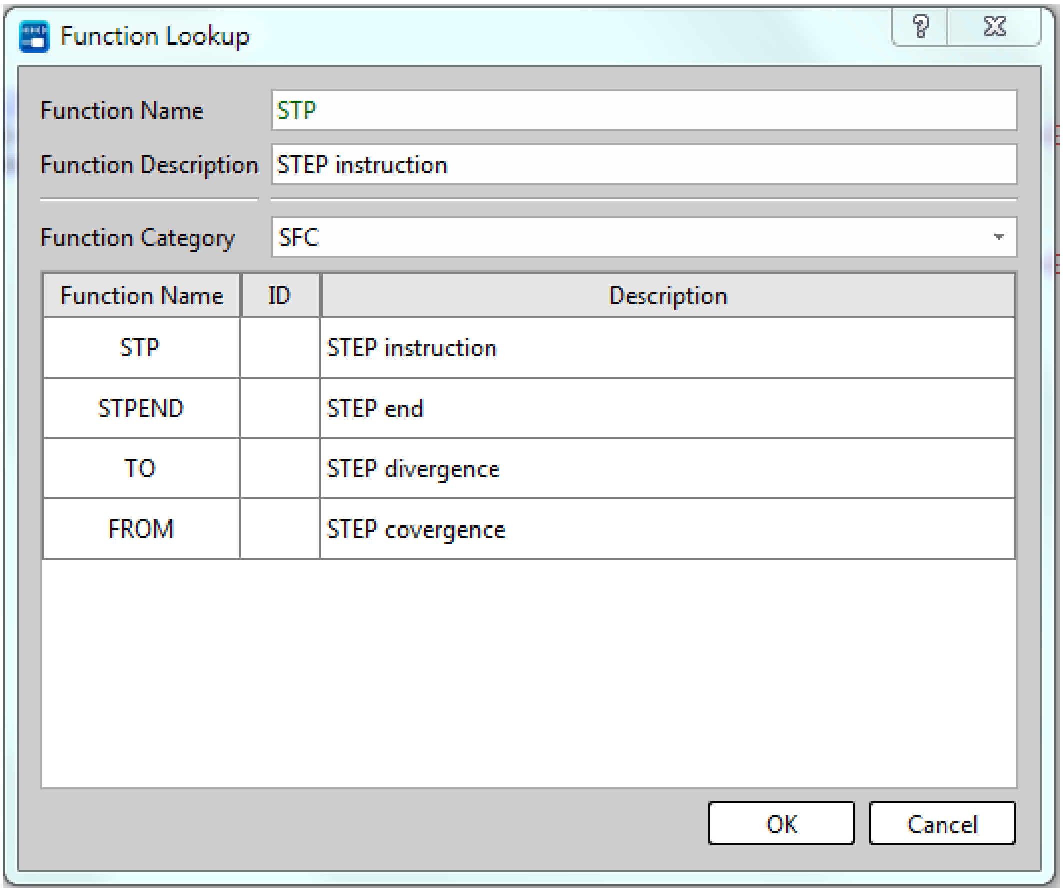
The maximum number of downward horizontal branch loops of an initial step is 16

8-3 Introduction of Step Instruction：STP、FROM、TO、STPEND

This section will introduce step instructions, and how to call instructions in UpperLogic, and how to use them.

Step instructions can be called by:

Select the function bar Ladder → Function Instruction；Or click the component panel icon; or right-click in the ladder diagram program area to display a pop-up menu, Function Instruction → Function Instruction, click on the position where you want to input the step command in the ladder diagram program area, All categories of function instructions will appear, select[SFC instruction], there are four step instructions [STP], [FROM], [TO], and [STPEND] on the right of the instruction name, as shown in the figure below:：



8-3-1 STP

● STP Sx ：S0 ≦ Sx ≦ S7（Displayed in UperLogic）or STP Sx：S0 ≦ Sx ≦ S7

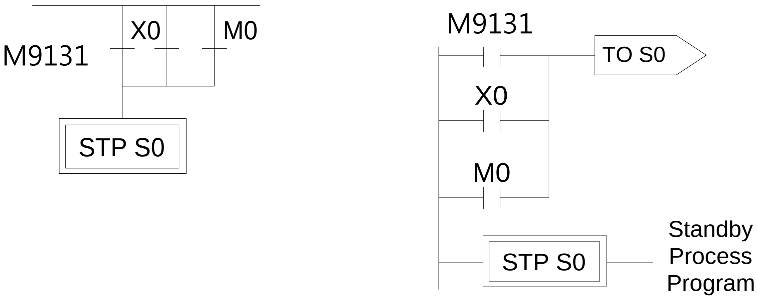
This instruction is the initial step instruction, from which the step control of each mechanical process can be derived. M-Series can provide up to 8 initial step points, that is to say, a PLC can control up to 8 processes at the same time. each step process can operate independently or generate operation results for reference by other processes.

【Example 1】Start the initial step point S0 every time when turn on PLC

UperLogic

【Example 2】Every time turn on PLC or press the button, or an abnormality in automatic production occurs and there is no personnel to deal with it within a certain period of time, it will automatically enter the initial step point S0 and stand by.

UperLogic



【Description】X0：button；M0：Abnormal Contact

● STP Sxxxx ：S20≦Sxxxx≦S3103（Displayed in UperLogic）

or

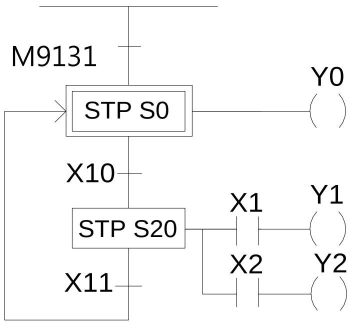
STP Sxxxx ： S20≦Sxxxx≦S3103

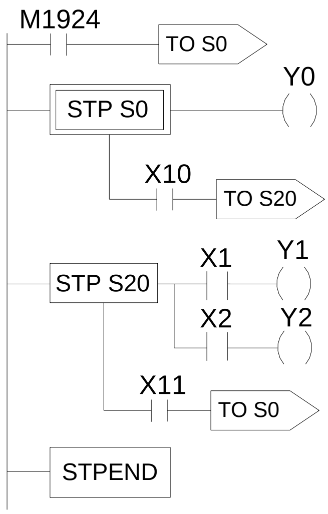
This instruction is a step instruction, each step in a process represents a step of sequence. If the status of step is ON then the step is active and will execute the ladder program associate to the step.

【Example】

UperLogic

M9131





【Description】

1. When ON, the initial step S0 is ON and Y0 is ON.

2. When transfer condition X10 is ON (in actual application, the transferring condition may be formed by the serial or parallel combination of the contacts X, Y, M, T and C), the step S20 is activated. The system will automatically turn S0 OFF in the current scan cycle and Y0 will be reset automatically to OFF.

i.e.X10 ON⇒ { S20 ON, S0 OFF } ⇒ { X1 ON →Y1 ON, X2 ON →Y2 ON, Y0 OFF }

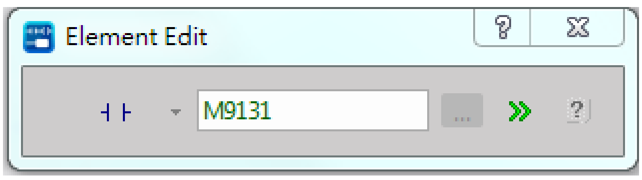
3. When the transfer condition X11 is ON, the step S0 is ON, Y0 is ON and S20, Y1 and Y2 will turn OFF at the same time.

i.e.X11 ON⇒ { S0 ON, S20 OFF } ⇒ { Y0 ON, Y1 OFF, Y2 OFF }

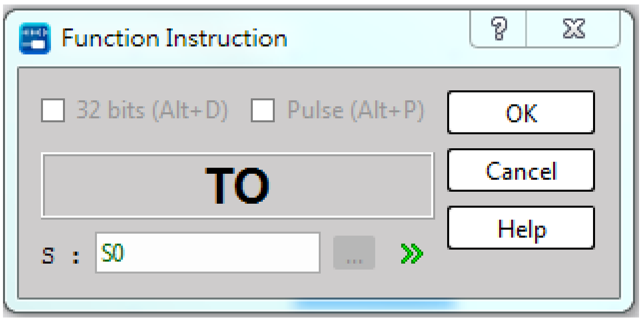
- Enter step point (STP Insruction)

If we want to set an initial step point S0 for each boot, the method is as follows:

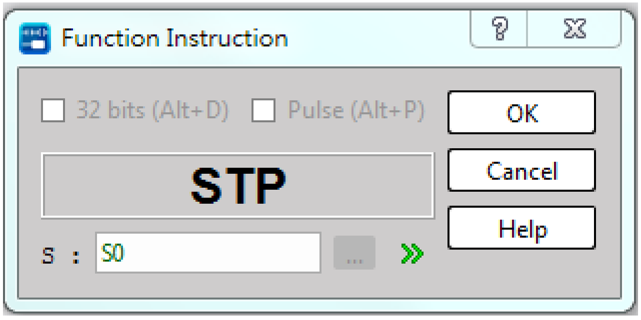
Select the A contact component on the component tray, click on the ladder diagram network, and enter "M9131" in the number input box :



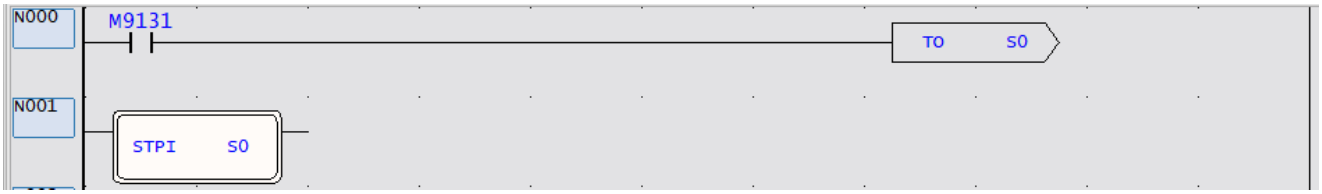
Click on the component panel icon, click after the "M9131" contact, the [Application Command] window will appear, select "SFC Instruction" under [Type], select "TO" for [Instruction Name], and press the "OK" button , the following window appears :



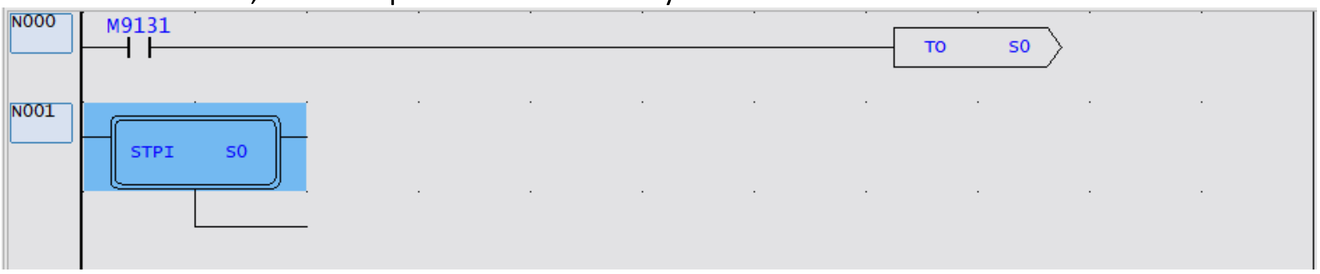
Enter "S0", press the "OK" button, and repeat the "SF instruction", this time select "STP" for [instruction name], and the following figure will appear :



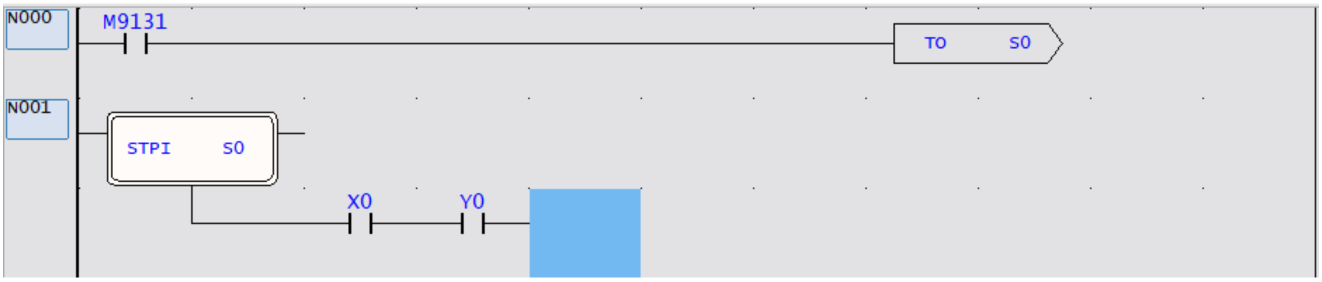
Input "S0" and press the "OK" button to complete the operation of setting an initial step point S0 for each boot.



You can also add state transition conditions for the initial step point. First, place the cursor on the component panel to select the [vertical line] component, and then click on "STPI S0"; or stop the cursor on "STPI S0", and then press the shortcut key "V" works too.



After the divergence line appears, add transition conditions, for example, we add two transition conditions "X0" and "Y0".



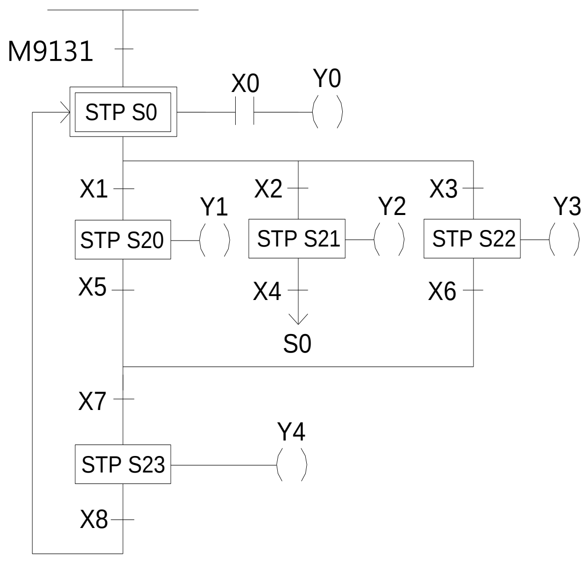
After adding the state point to be transferred, we assume that when the two transfer conditions of "X0" and "Y0" are satisfied (ON), it will transfer to the state point "S21". Call out the [SFC function instruction] category, select [TO] for the instruction name; or press the shortcut key “>”, after a dialog box appears, enter “S21” to complete the following example :

8-3-2 FROM

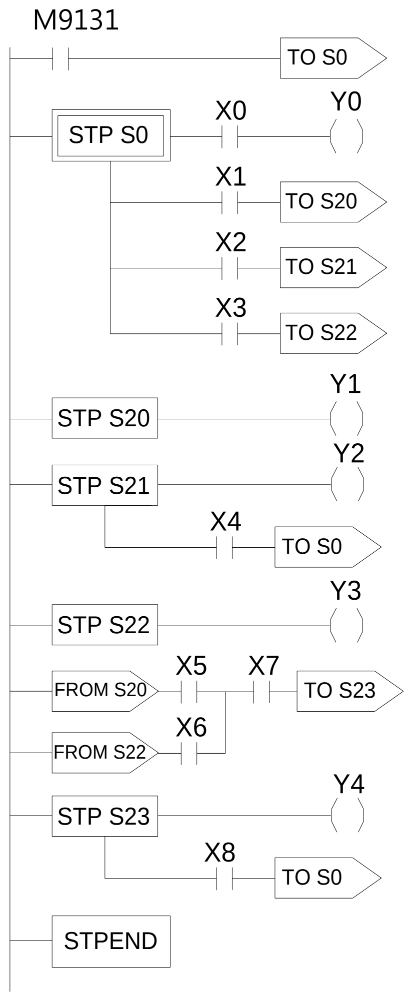
● FROM Sxxxx ： S0 ≦ Sxxxx ≦ S3103（Displayed in UperLogic）

The instruction describes the source step of the transfer, i.e. moving from step Sxxxx to the next step in coordination with transfer condition.

【Example】



UperLogic

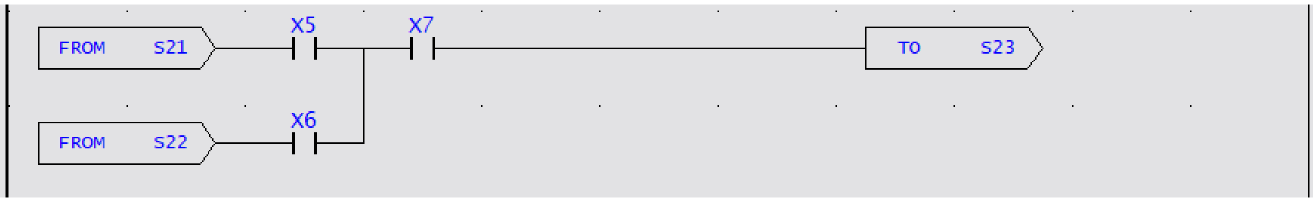


【Description】：

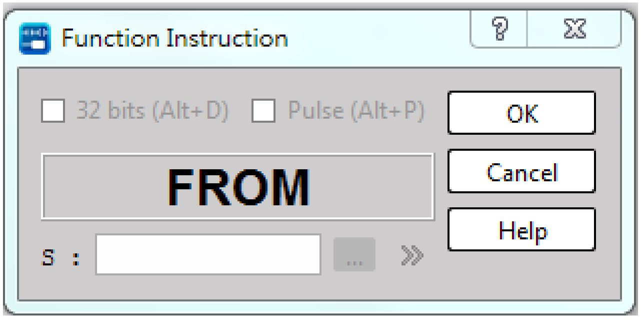
1. When ON, the initial step S0 is ON. If X0 is ON, then Y0 will be ON.
2. When S0 is ON: a. if X1 is ON, then step S20 will be ON and Y1 will be ON.
   b. if X2 is ON, then step S21 will be ON and Y2 will be ON.
   c. if X3 is ON, then step S22 will be ON and Y3 will be ON.
   d. if X1, X2 and X3 are all ON simultaneous, then step S20 will have the priority to be ON first and either S21 or S22 will not be ON.
   e. if X2 and X3 are ON at the same time, then step S21 will have the priority to be ON first and S22 will not be ON.。
3. When S20 is ON, if X5 and X7 are ON at the same time, then step S23 will be ON, Y4 will be ON and S20 and Y1 will be OFF.
4. When S21 is ON, if X4 is ON, then step S0 will be ON and S21 and Y2 will be OFF.
5. When S22 is ON, if X6 and X7 are ON at the same time, then step S23 will be ON, Y4 will be ON and S22 and Y3 will be OFF.
6. When S23 is ON, if X8 is ON, then step S0 will be ON and S23 and Y4 will be OFF.

- Enter convergence point (FROM)

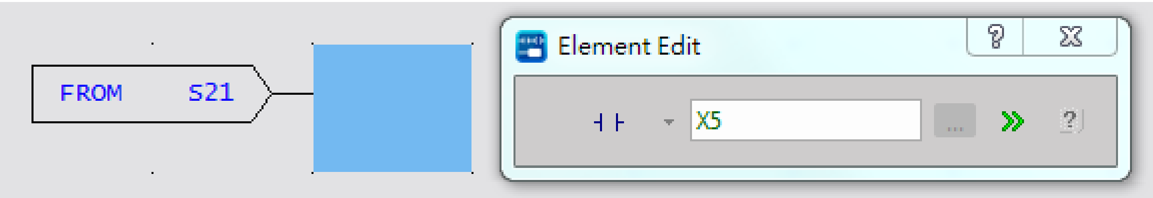
1. selective convergence



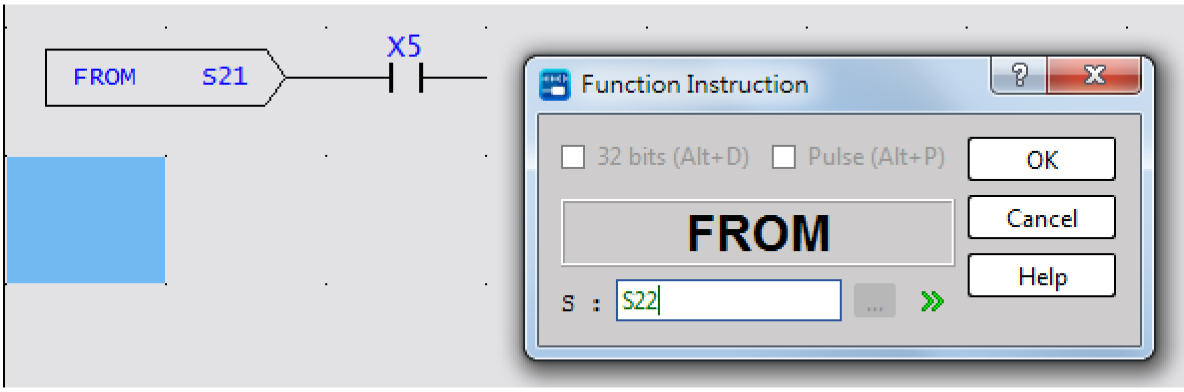
If we want to make the above results, we will do the following: We first call the [SFC function instruction] category by referring to the operation method in section 7.4.2, select [FROM] for the instruction name, and press“OK”, and the following window will appear.

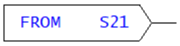


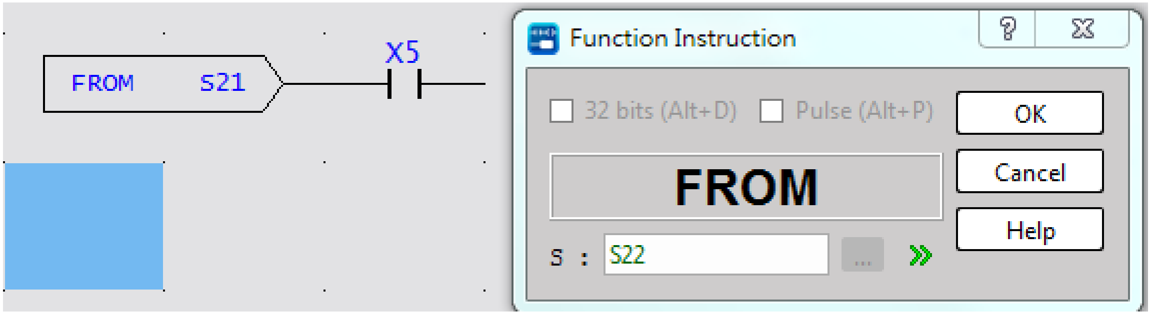
Input "S21", press the "OK" button, move the cursor on the component panel to select the [A contact] component and click it, the following window will appear:



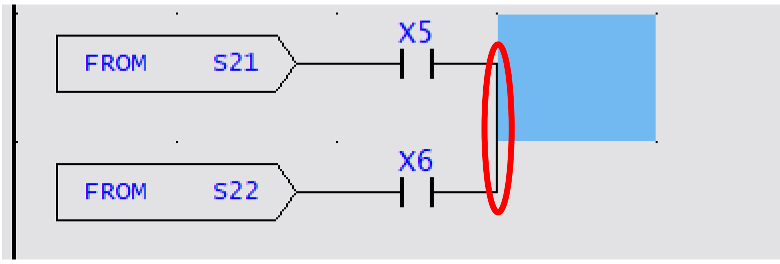
Input "X5", press "ENTER", use the function instruction again, call out the [SFC function instruction] category, select [FROM] for the instruction name, and press "OK".



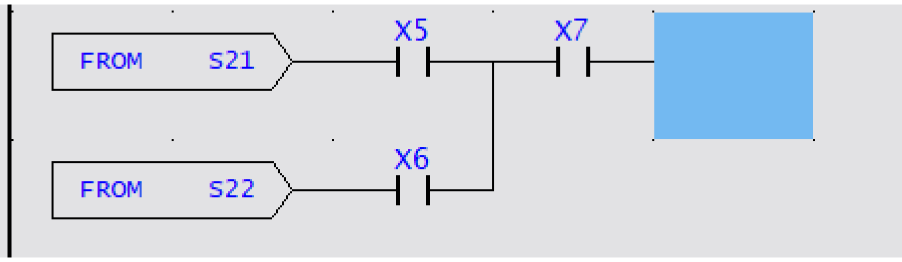
Input "S22", press the "OK" button, move the cursor on the component panel to select the [A contact] component and click it  , the following window will appear :



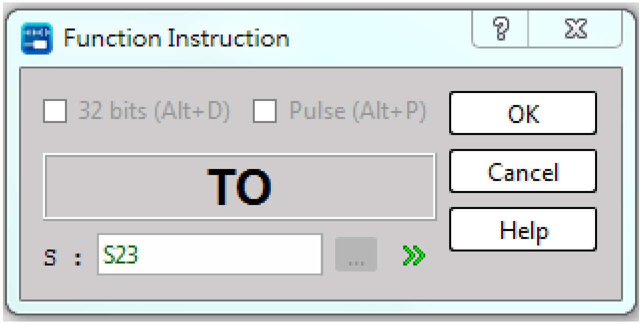
Input "X6", press "ENTER" , the cursor will select the [vertical line] component in the component panel, click it immediately after the X5 contact; or press the shortcut key "V" after the cursor is placed in X5, a vertical line will appear. line, as shown in the following figure :



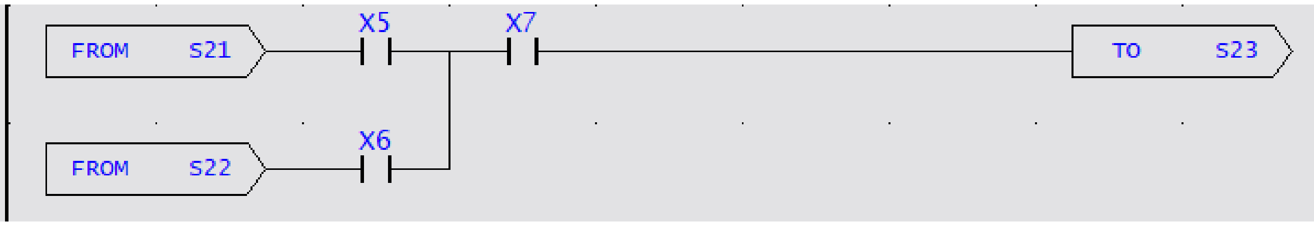
Enter "X7", as shown in the following image：



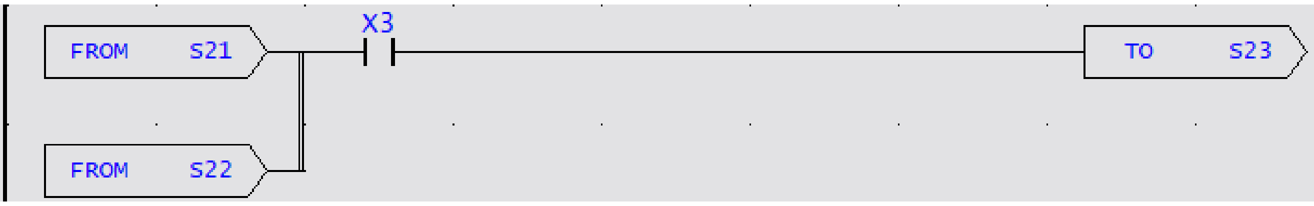
Use the function command again, call out the [SFC function instruction] category, select [TO] for the instruction name, and then press "OK" to appear.



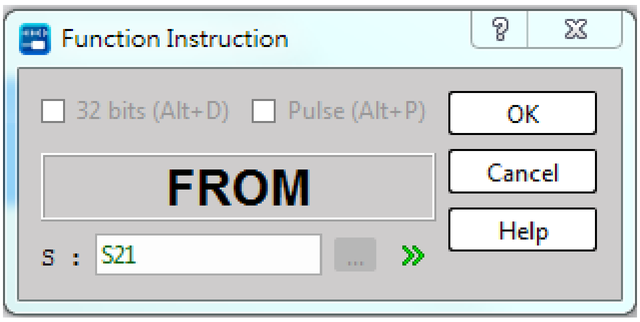
Input "S23" and press "OK" to complete an example of selective convergence. As shown below.



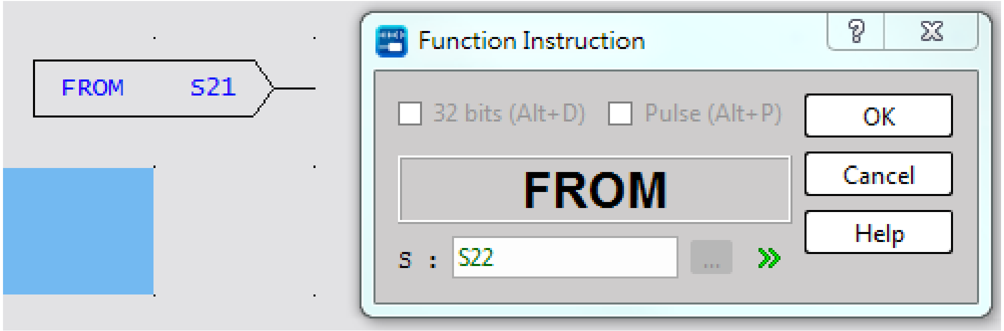
2. Simultaneous convergence



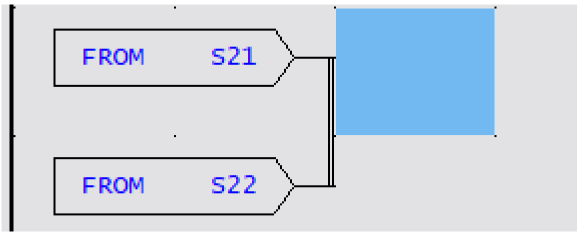
If we want to make the above result, the method is as follows: We first call the [SFC function instruction] category by referring to the operation method in section 7.4.2, select [FROM] for the instruction name, and press "OK", and the following window will appear:

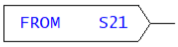


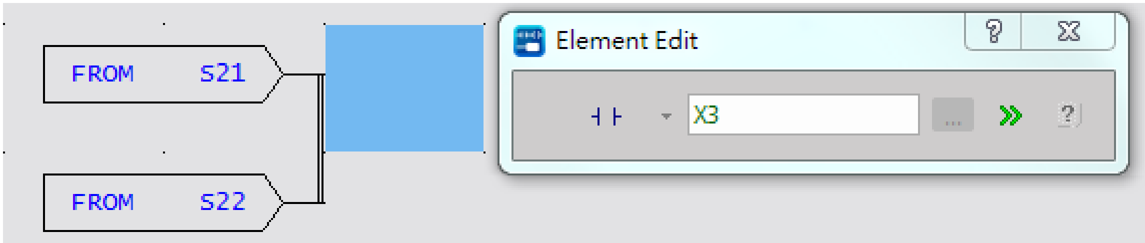
Input "S21", press "OK", call out the [SFC function instruction] category again, select [FROM] for the instruction name, and press "OK", the following window will appear :



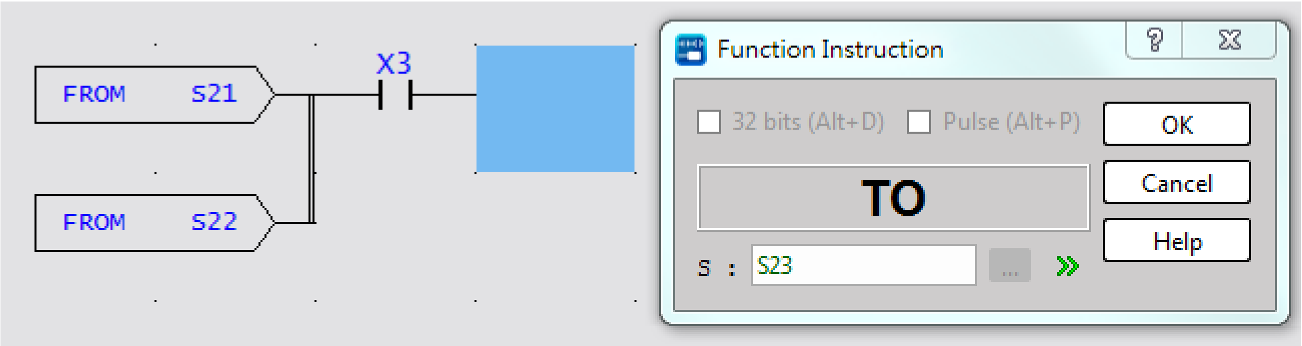
Enter "S22", press "OK" , select the [vertical line] component with the cursor on the component panel, and then click it; or press the shortcut key "V", that is, to complete the expression of the parallel and confluent ladder diagram program.



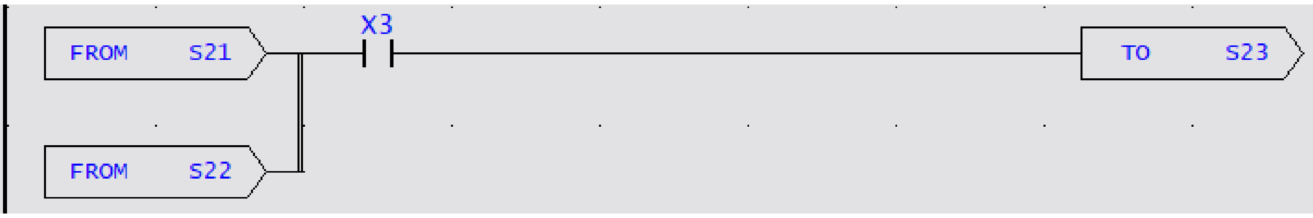
Select the [A Contact] component with the cursor on the component panel, and then click  :

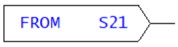


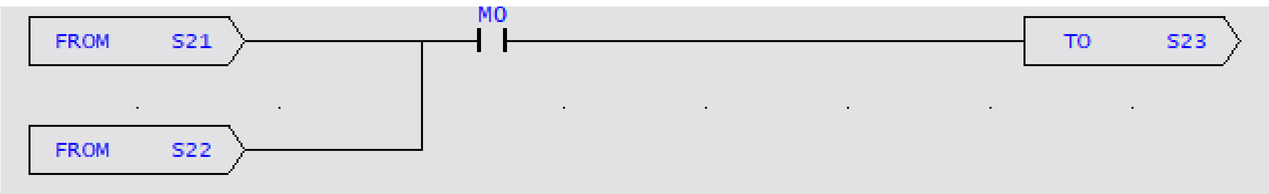
Enter "X3" and press "ENTER". Use the function command again, call out the [SFC function instruction] category, select [TO] for the instruction name and press "OK", and the following window will appear：



Input "S23" and press "OK" to complete the example of simultaneous convergence. As shown below：



Special attention should be paid to the [vertical line] element in order to complete the simultaneous convergence. It must be next to  . Once there is a space in the middle, it will become a selective convergence, as shown below:



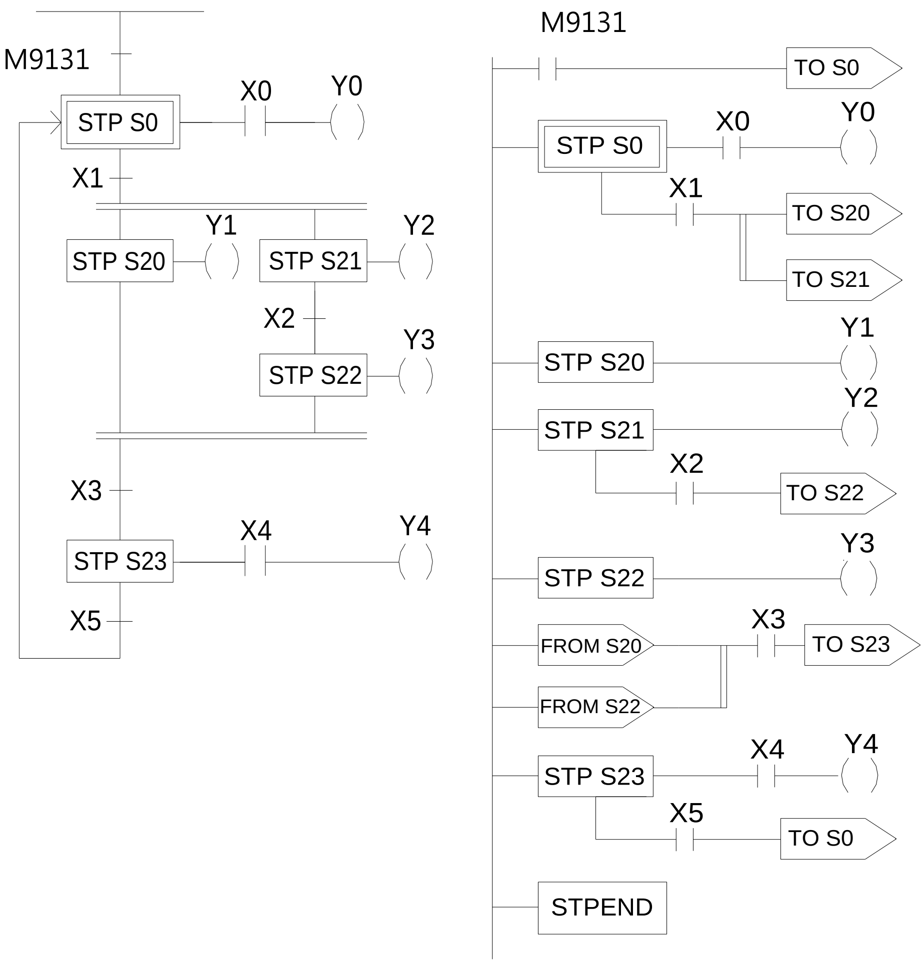
8-3-3 TO

● TO Sxxxx ： S0≦Sxxxx≦S3103（Displayed in UperLogic）

This instruction describes the step to be transferred to.

【Example】

UperLogic



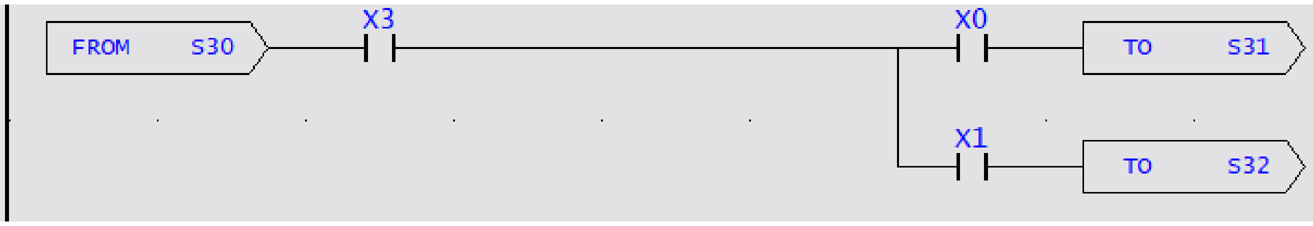
【Description】：

1. When ON, the initial step S0 is ON. If X0 is ON, then Y0 will be ON.
2. When S0 is ON: if X1 is ON, then steps S20 and S21 will be ON simultaneously and Y1 and Y2 will also be ON.
3. When S21 is ON: if X2 is ON, then step S22 will be ON, Y3 will be ON and S21 and Y2 will be OFF.
4. When S20 and S22 are ON at the same time and the transferring condition X3 is ON, then step S23 will be ON (if X4 is ON, then Y4 will be ON) and S20 and S22 will automatically turn OFF and Y1 and Y3 will also turn OFF.
5. When S23 is ON: if X5 is ON, then the process will transfer back to the initial step, i.e. So will be ON and S23 and Y4 will be OFF.

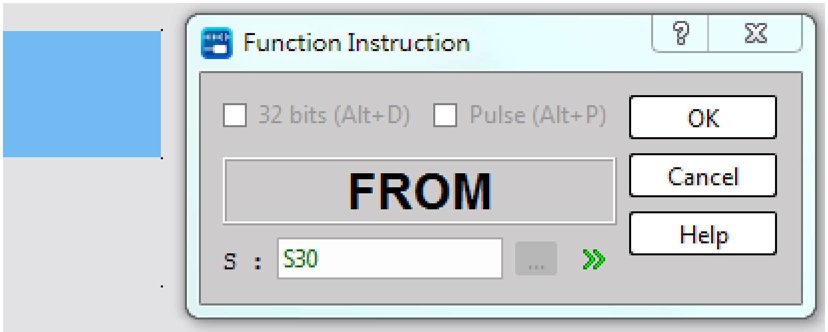
- Enter divergence point (TO Instruction)

Using the UperLogic ladder diagram program are as follows：

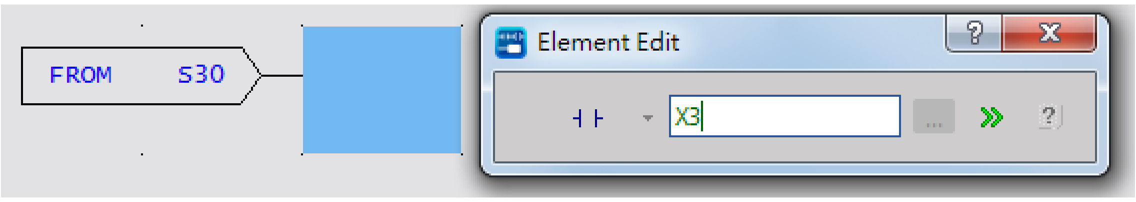
1. Selective Divergence



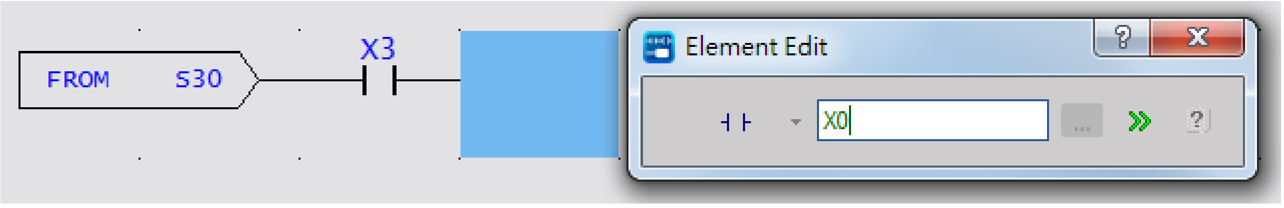
If we wanted to make the above result :
Place the cursor at the desired input position in the program area, call out the [SFC function instruction] category, and select the instruction name [FROM] :



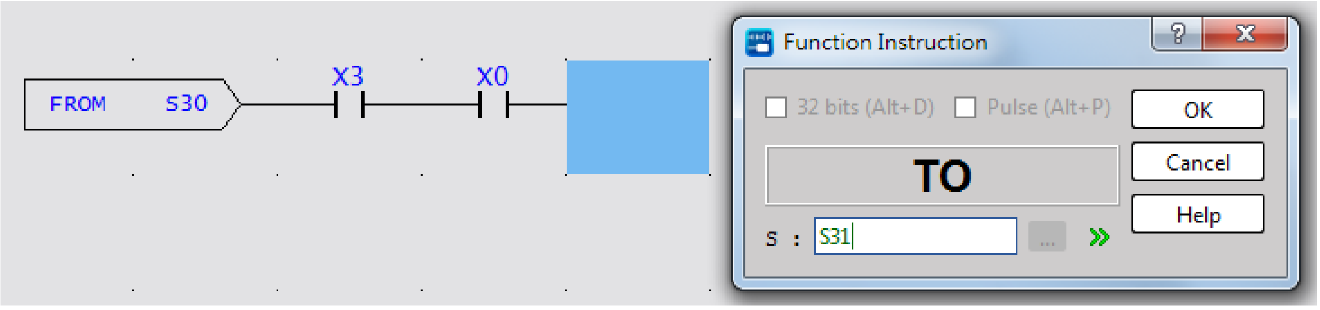
Input "S30" and press "OK", the FROM instruction S30 element will appear in the program area. Cursor to select the A contact element and click on it, and enter the "X3" number; or directly enter "AX3" directly after it, as shown in the following window：



Type X0 followed by it,

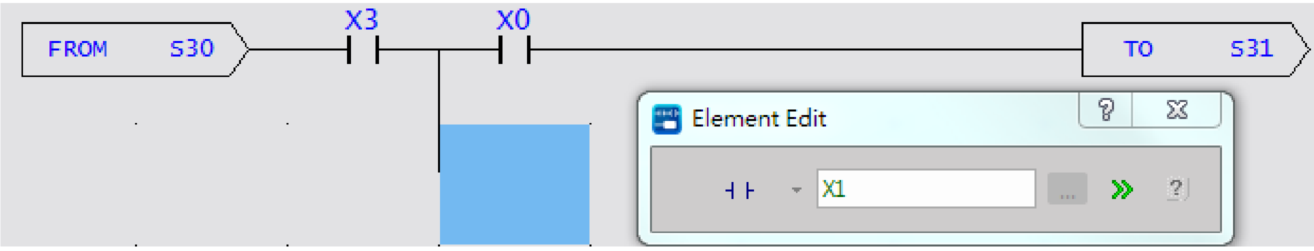


Place the cursor at the desired input position in the program area, then call the [SFC function instruction] category, and select [TO] ;

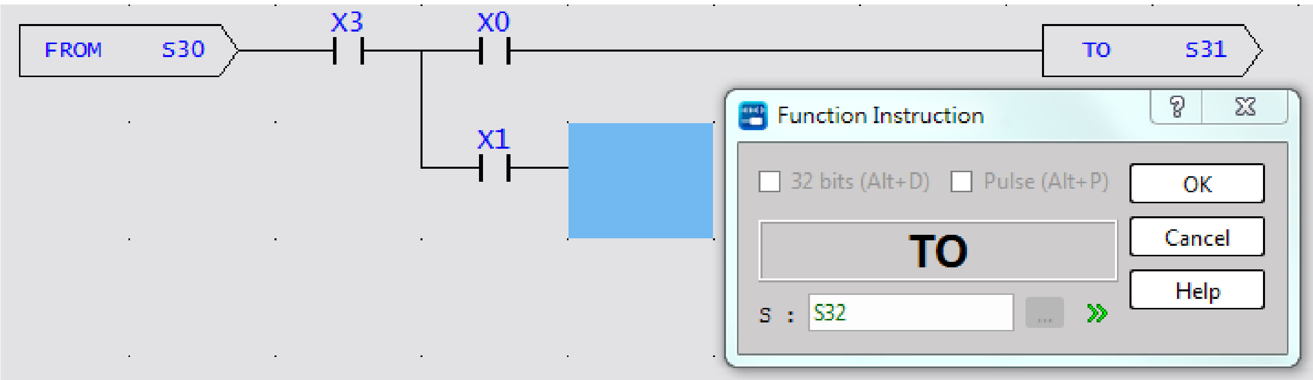


Enter "S31", press "OK", the cursor is placed at the X0 position, enter "V", and add a vertical line, as shown in the following figure：

Place the cursor below X0 and enter "X1" or "X1A"：

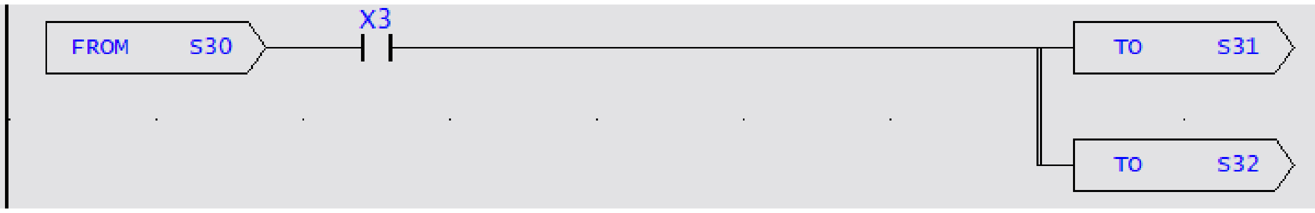


Call the [SFC function instruction] category, and select [TO]：



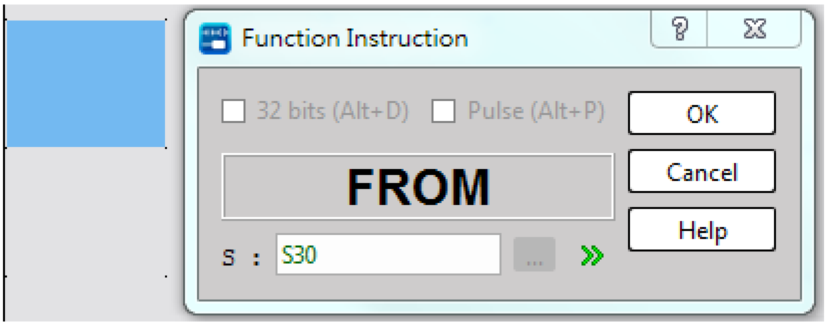
Input "S32" and press "OK", an example of selective divergence is completed. As shown below：

2. Simultaneous Divergence

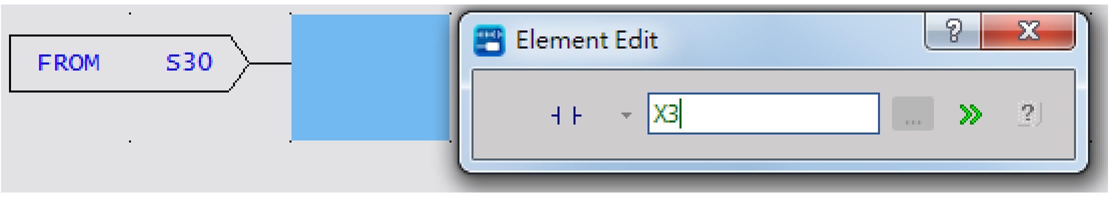


If we want to make the above result, the method is as follows :

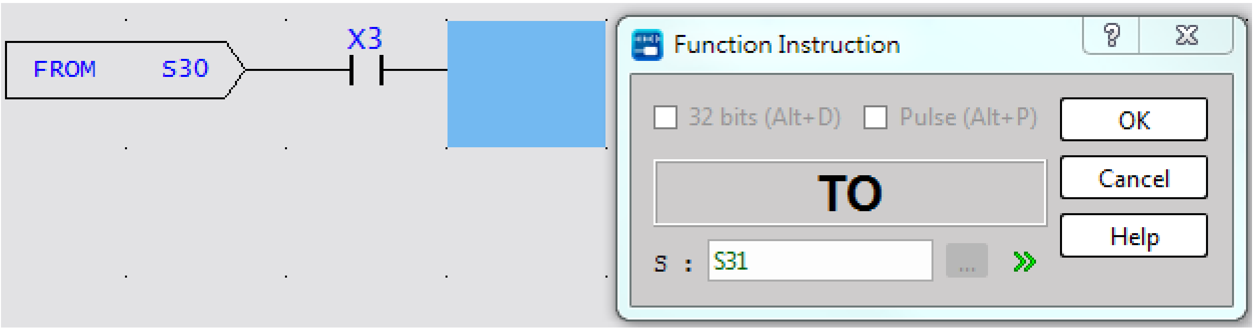
Place the cursor at the desired input position in the program area, call the [SFC function instruction] category, and select [FROM] ：



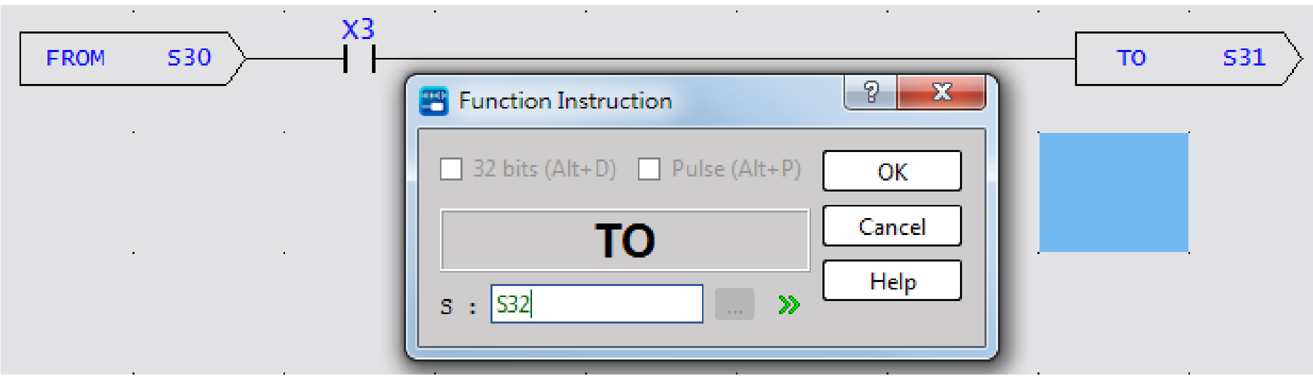
Input "S30" and press "OK", the FROM instruction will appear. Cursor to select the A-contact component, click and select it, and enter "X3" or "AX3", as shown in the following window ：

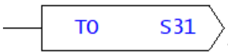


Place the cursor at the desired input position in the program area, call the [SFC function instruction] category, and select [TO] ：

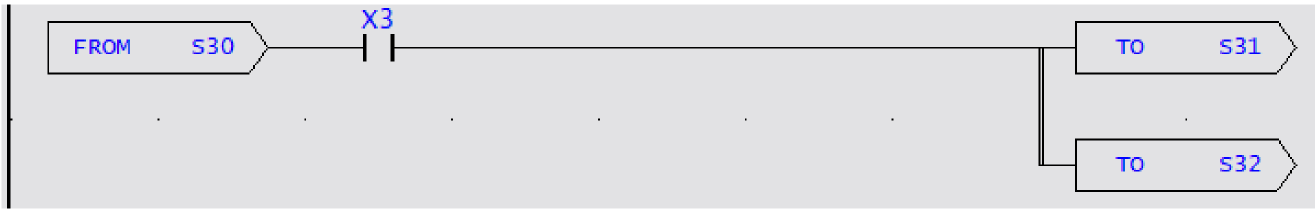


Input "S31" and press "OK", the TO instruction will appear. At the position below the instruction TO command S31, call the [SFC function instruction] category, and select [TO] ：



Enter "S32" and press "OK". Select the vertical line component, click the icon in the program area 

; or press the shortcut key "V", the following figure will appear:



That is, to complete the example of Simultaneous divergence.

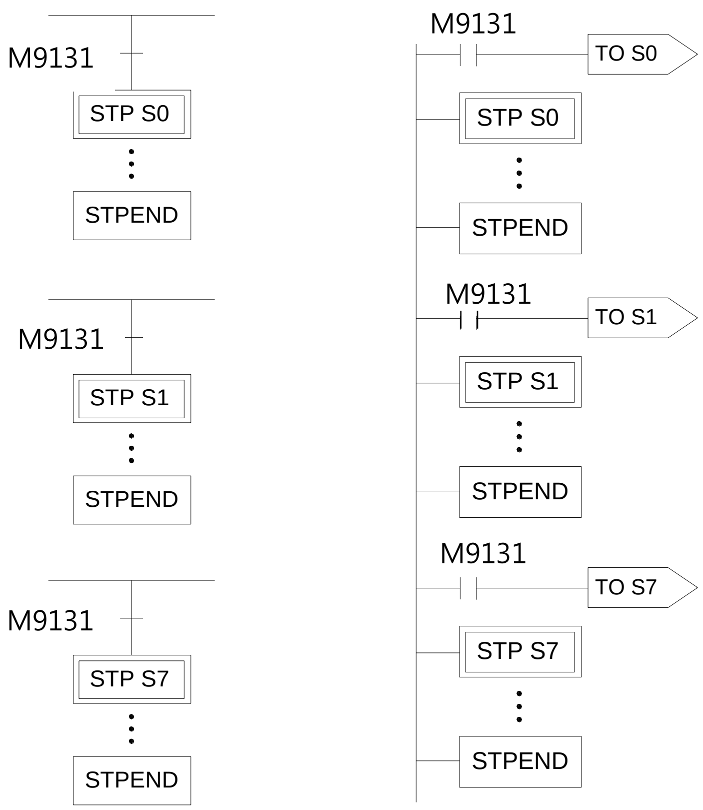
8-3-4 STPEND

● STPEND ： （Displayed in UperLogic）

This instruction represents the end of a process, which is required for all processes to work correctly. PLC has at most 8 step processes (S0~S7) that can be controlled at the same time, so there are at most 8 STPEND instructions.

【Example】

UperLogic



【Description】8 step processes are activated at the same time when PLC boot.

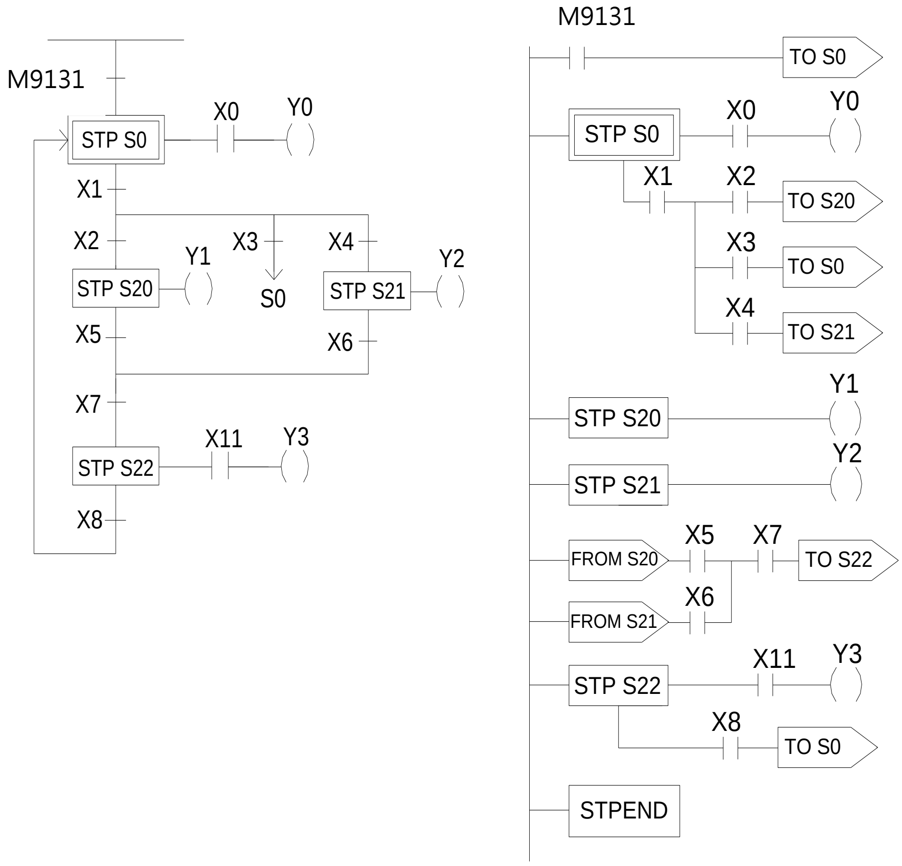
8-4 Notes for Writing a Step Ladder Diagram

【Notes】

- In actual applications, the ladder diagram can be used together with the step ladder.
- There are 8 steps, S0 ~ S7, that can be used as the starting point and are called the “initial steps”.
- When PLC starts operating, it is necessary to activate the initial step. The M9131 (the first scan ON signal) provided by the system may be used to activate the initial step.
- Except the initial step, the start of any other steps must be driven by other step.
- It is necessary to have an initial step and the final STPEND instruction in a step ladder diagram to complete a step process program.
- There are 3085 steps, S20 ~ S3103, available that can be used freely. However, used numbers cannot be repeated. S2064 ~ S3103 are retentive(The range can be modified by users), can be used if it is required to continue the machine process after power is off.
- Basically, a step must consist of three parts which are control output, transition conditions and transition targets.
- MC and SKP instructions cannot be used in a step program and the sub-programs. It’s recommended that JMP instruction should be avoided as much as possible.
- If the output point is required to stay ON after the step is divergent to other step, it is necessary to use the SET instruction to control the output point and use RST instruction to clear the output point to OFF.
- Looking down from an initial step, the maximum number of horizontal paths is 16. However, a step is only allowed to have up to 8 branch paths.

【Example 1】

UpreLogic

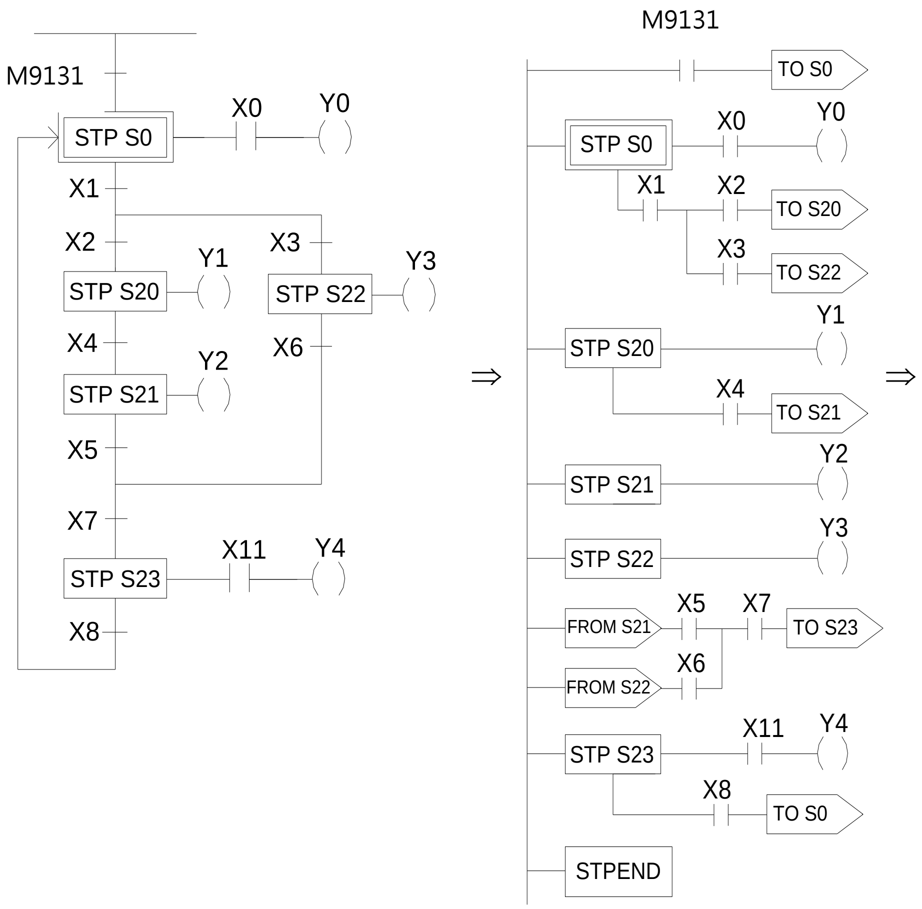


【Description】：1. Input the condition to initial step S0

2. Input the S0 and the divergent conditions of S20, S0 and S21
3. Input the S20
4. Input the S21
5. Input the convergence of S20 and S21
6. Input the S22

【 Example 2】

UperLogic

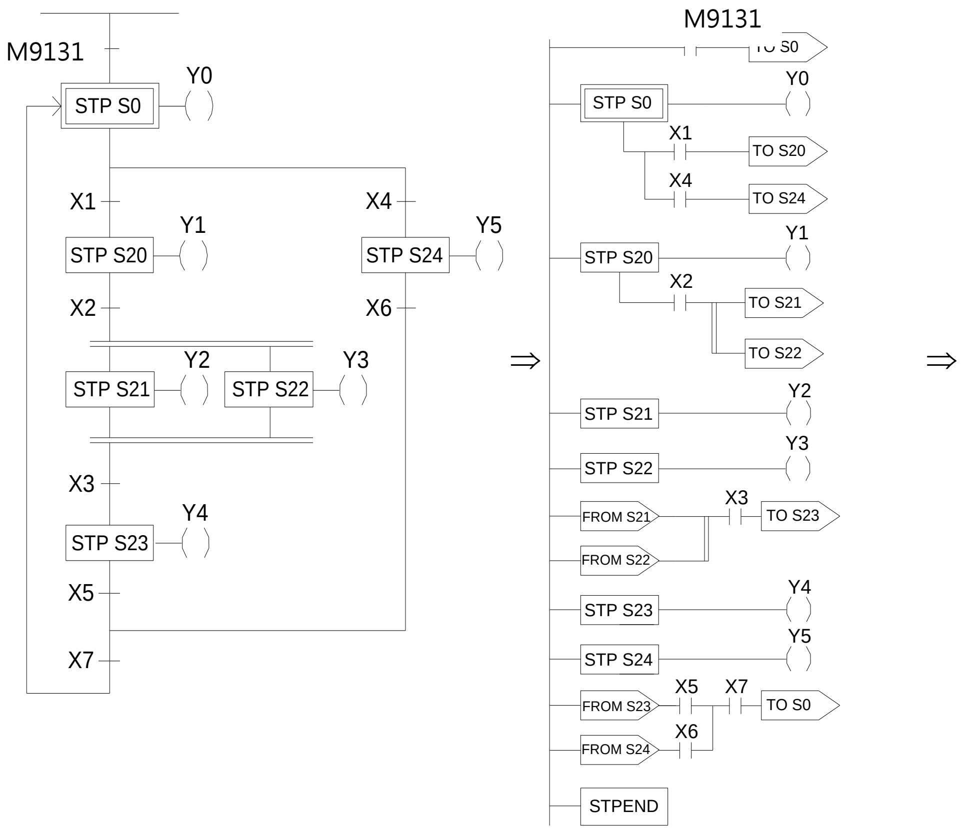


【Description】：1. Input the condition to initial step S0

2. Input the S0 and the divergent condition of S20 and S22
3. Input the S20
4. Input the S21
5. Input the S22
6. Input the convergence of S21 and S22
7. Input the S23

【 Example 3】

UperLogic

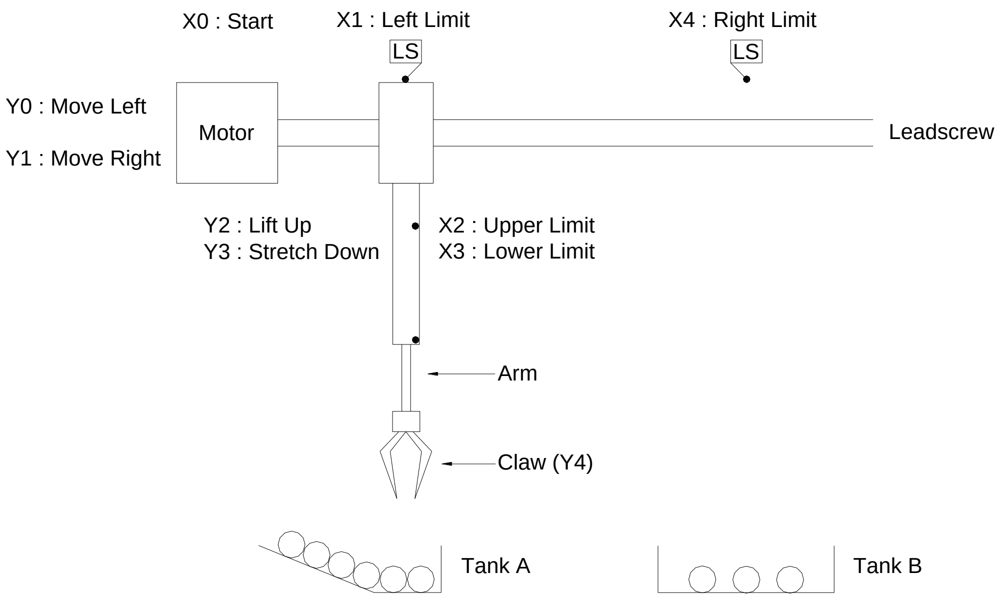


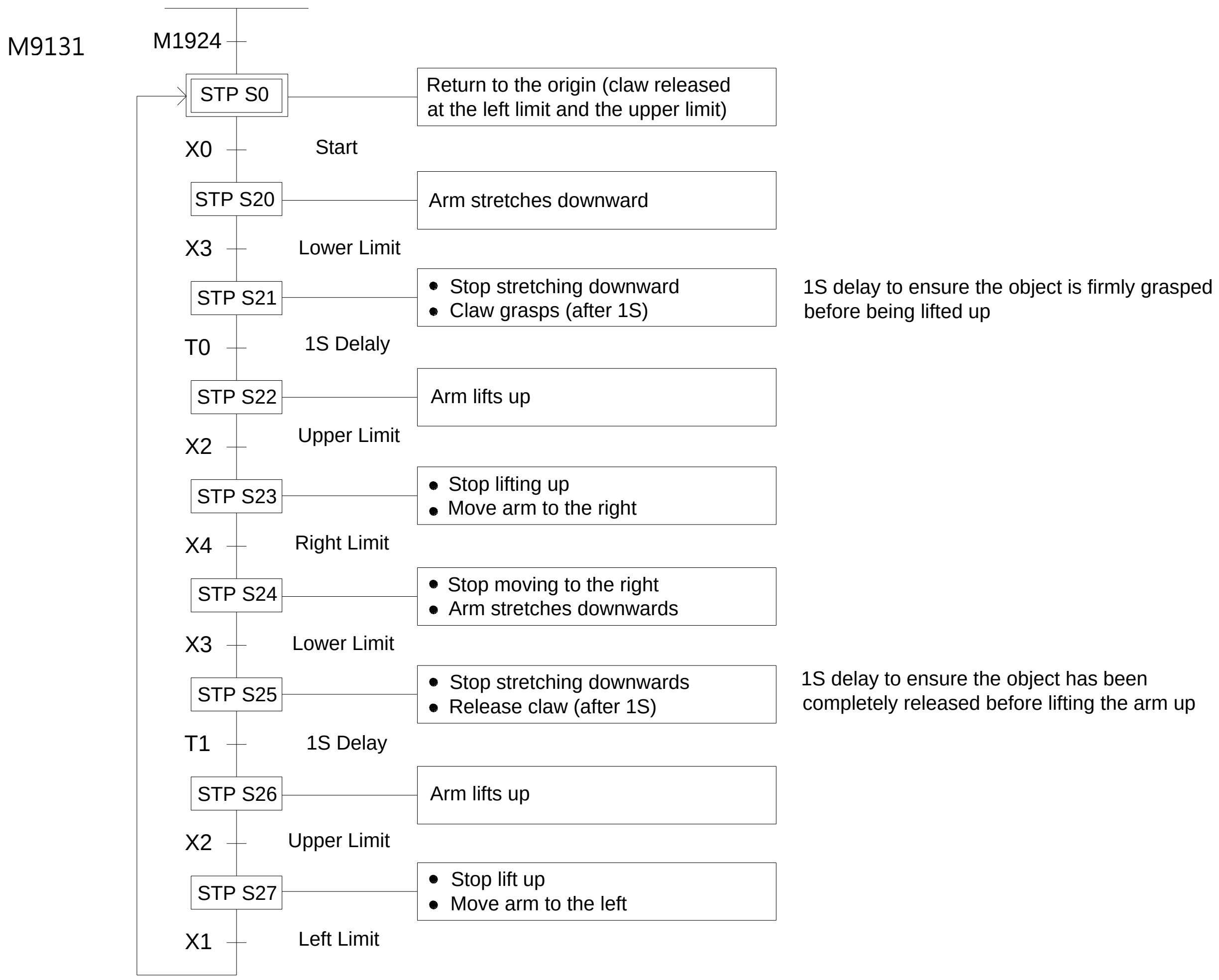
【Description】：1. Input the condition to initial step S0

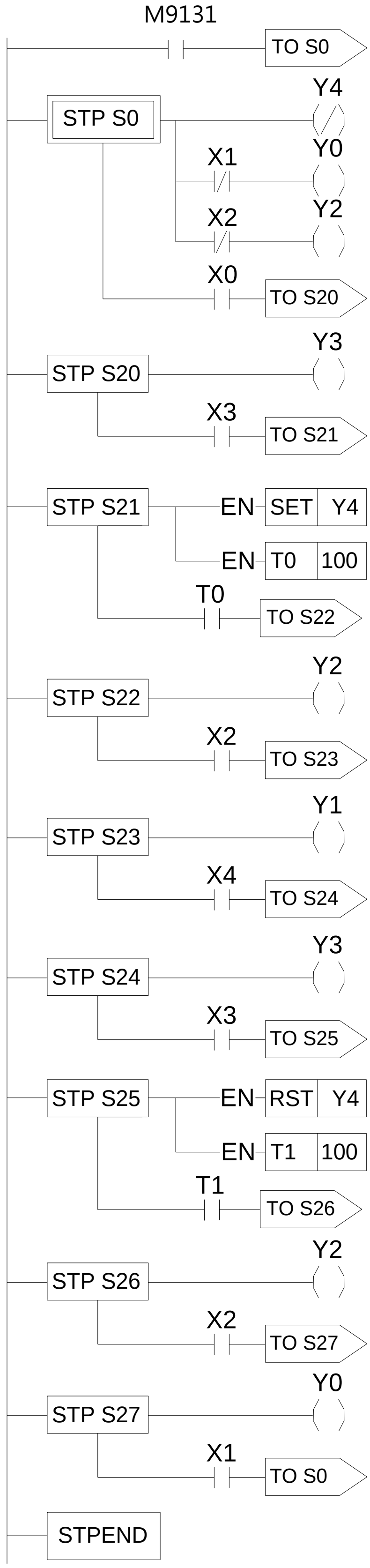
2. Input the S0 and the divergences of S20 and S24
3. Input the S20
4. Input the S20 and the divergences of S21 and S22
5. Input the S21
6. Input the S22
7. Input the convergences of S21 and S22
8. Input the S23
9. Input the S24
10. Input the convergences of S23 and S24

8-5 Application Examples

【 Example 1】 Grasp an object from tank A and put it in Tank B

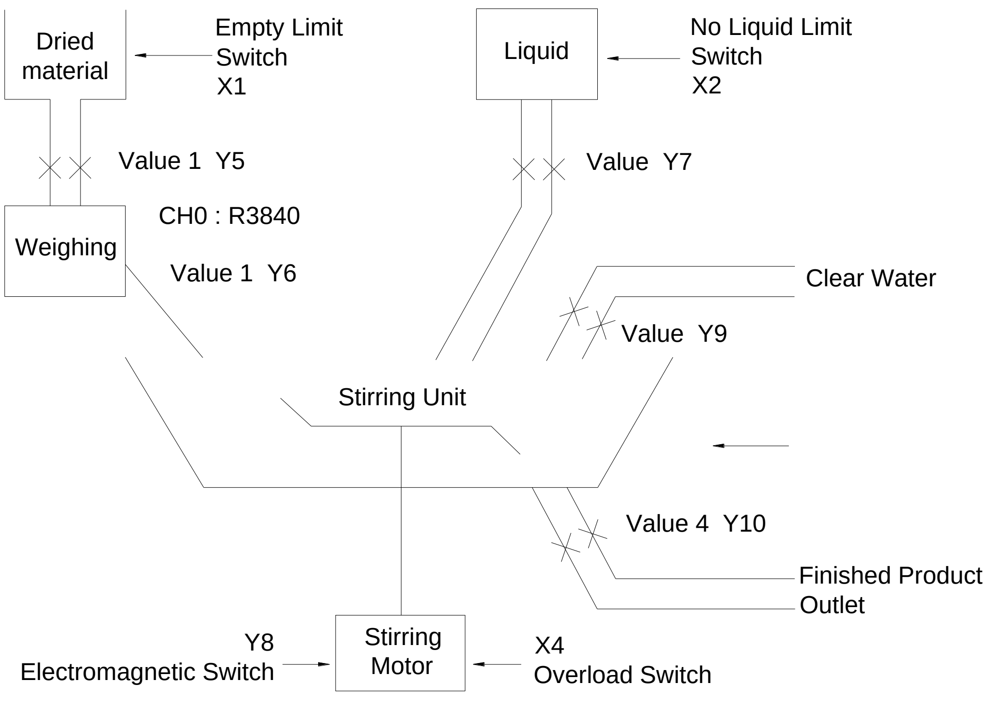






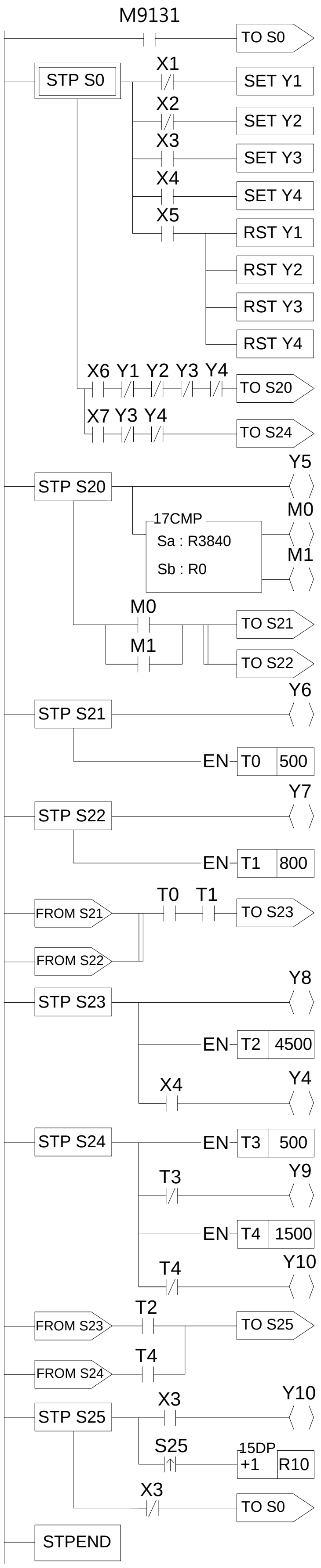
Release claw
Return to the left limit
Return to the upper limit
Turn the switch ON before moving to S20
Stretch arm downward
Move to S21 after stretching to the lower limit
Claw grasps (since the SET instruction is used, Y4 should remain ON after departing from STP S21)
Divergent into S22 after 1S
Lift the arm up
Divergent into S23 after reaching the upper limit
Move arm to the right
Divergent into S24 after moving to the right limit
Stretch the arm downward
Divergent into S25 after stretching to the lower limit
Release claw
Delay for 1S
Transfer into S26 after 1S
Lift the arm up
Divergent into S27 after reaching the upper limit
Move the arm to the left
Divergent into S0 after moving to the left limit (a complete cycle)

【Example 2】Liquid Stirring Process



- Input Points：Empty limit switch X1
  Noliquid linit switch X2
  Empty limit switch X3
  Over-load switch X4
  Warning clear button X5
  Start button X6
  Water washing button X7
- Warning Indicators: Empty dried material Y1
  Insufficient liquid Y2
  Empty stirring unit Y3
  Motor over-load Y4
- Output point: dry material feeding valve Y5
  Dry feed valve Y6
  Liquid feed valve Y7
  Start motor solenoid valve Y8
  Fresh water inlet valve Y9
  Finished product feed valve Y100

- Weighing Output：CH0（R3840）



UperLogic

Warning indicators

Reset warning

Production start

Water washing start

Input weighing

Status after weighing

Divergent into S21 and S22

Add liquid to stirring unit

Complete dried material and liquid input, transfer the status to S23

Stirring timer

Wash stirring unit

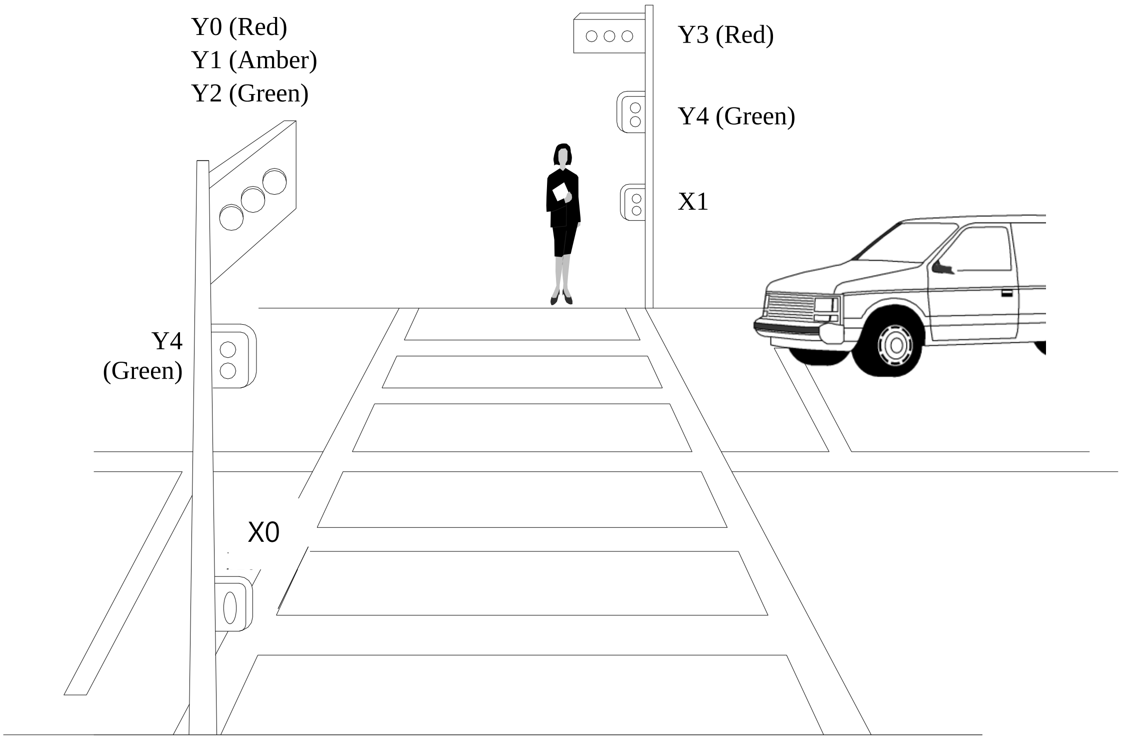
Input clean water

Drain water out

Output finished product

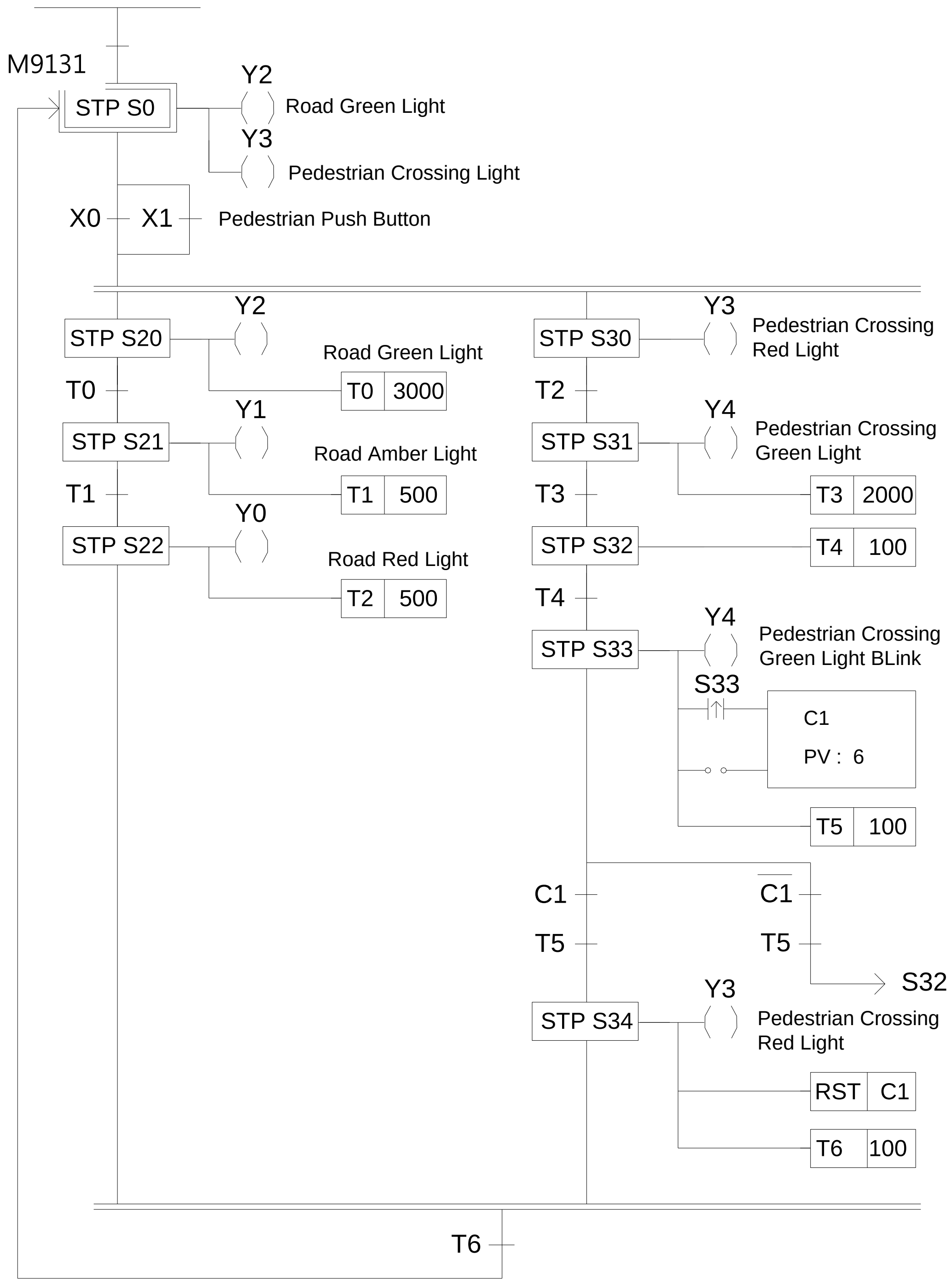
and accumulate the cycle

【Example 3】Pedestrian Crossing Lights

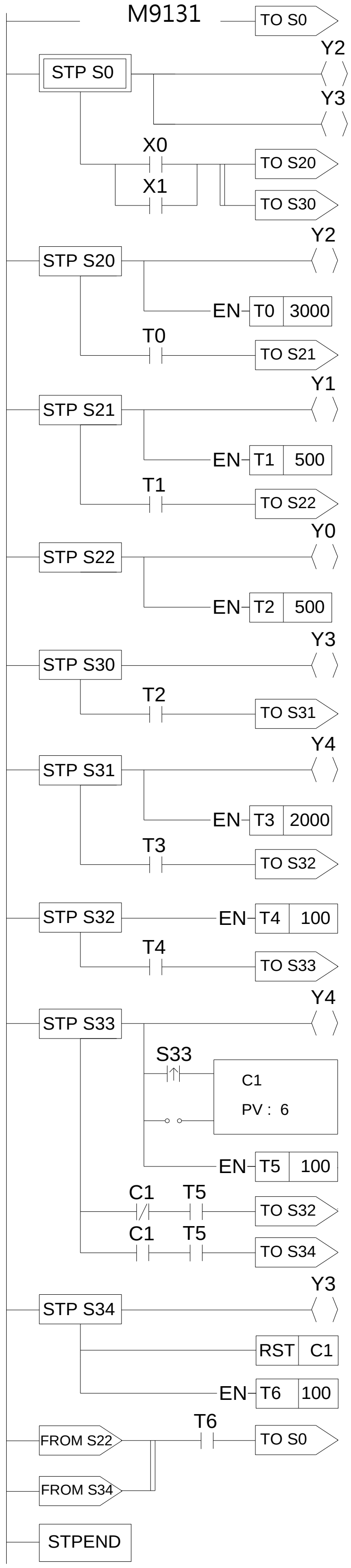


- Input Point： Pedestrian Push Button X0
  Pedestrian Push Button X1
- Output Points： Road Red Light Y0
  Road Amber light Y1
  Road Green Light Y2
  Pedestrian Crossing Red Light Y3
  Pedestrian Crossing Green Light Y4

- Pedestrian Crossing Lights Control Process Diagram



● Pedestrian Crossing Lights Control Program
UperLogic



8-6 Syntax Check Error Codes for Step Instruction

The error codes for the usage of step instruction are as follows:

E51：TO(S0-S7) must begin with ORG instruction.

E52：TO(S20-S3103) can't begin with ORG instruction.

E53：TO instruction without matched FROM instruction.

E54：To instruction must comes after TO, AND, OR, ANDLD or ORLD instruction.

E56：The instructions before FROM must be AND, OR, ANDLD or ORLD

E57：The instruction after FROM can’t be a coil or a function

E58：Coil or function must before FROM while in STEP network

E59：More than 8 TO# at same network.

E60：More than 8 FROM# at same network.

E61：TO(S0-S19) must locate at first row of the network.

E62：A contact occupies the location for TO instruction.

E71：Incomplete connection (should not happen)

E72：Duplicated TO Sxxxx instruction.

E73：Duplicated STP sxxxx instruction.

E74：Duplicated FROM sxxxx instruction.

E76：TP(S0~S19) without a matched STPEND or STPEND without a matched STP(S0~S19).

E77 ：The previous network of STP(S0~S19) is not the only ORG~S19(S0~S19)

E78：TO(S20~S3103), STP (S20~S3103) or FROM instructions comes before or without STP(S0~S19).

E79：STP Sxxxx or FROM Sxxxx instructions comes before or without TO Sxxxx.

E80：FROM Sxxxx instruction comes before or without STP Sxxxx.

E81：The max level of branches must <=16.

E82：The max number of branches with same level must <=16.

E83：Not place the step instruction with TO->STP->FROM sequence.

E84：The definition of STP# sequence not follow the TO# sequence.

E85：Convergence do not match the corresponding divergence.

E86：Illegal usage of STP or FROM before convergent with TO instruction.

E87：STP# or FROM# comes before corresponding TO#.

E88：During this branch, STP# or FROM# comes before the corresponding TO#.

E89：FROM# comes before corresponding TO# or STP#.

E90：Invalid To# usage in the simultaneous branch.

E91：Last STP (S0~S19) has not been processed completely, use ORG, LBL, RTS, RTI, MCE, SKPE, FOR, NEXT, ENDD.

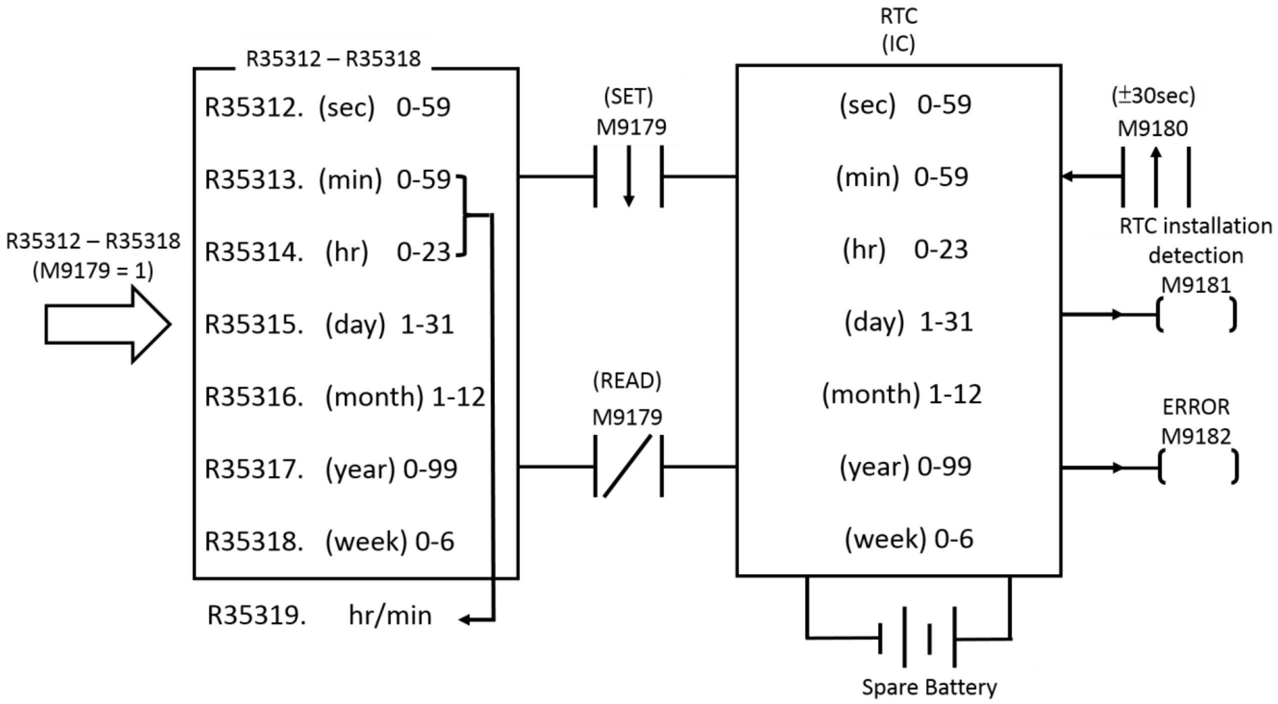
9

Real Time Clock (RTC)

A real time clock (RTC) has been built in the M-Sreies PLC's MC/MN main unit. No matter whether the PLC is switched on or off, the RTC will always keep accurate time. It provides 7 kinds of time value data-week, year, month, day, hour, minute and second. Users can take advantage of the real time clock to do 24 hour controls throughout the year (for example, businesses or factories can switch lights on and off at set times each day, control gate access, and do pre-cooling and pre-heating before business or operations begin). It can enable your control system to automatically coordinate with people's living schedules, and not only will it raise the level of automatic control, it will improve efficiency.

9-1 Correspondence Between RTC and the RTCR Within PLC

Within PLC, there are special purpose registers (RTCR) for storing the time values of the RTC. There are 8 RTCR registers in all, going from R35312 to R35319. R35312 to R35318 are used to store the 7 kinds of time values mentioned above, from weeks to seconds. Because in practical daily application, certain hour and minute time data is often used, we have specially merged the time values of the hour register (R35314) and minute register (R35313) within RTCR, and put them in R35319 high byte and low byte, so they can be accessed by the user. The diagram below shows the correspondence between RTC and the RTCR within PLC, as well as the control switch and status flag (M9179-M9182) related to RTC accessing.



9-2 RTC Access Control and Settings

Within PLC, R35312~R35318 registers have been allocated to store the time values of RTC, and this is of great convenience to the user. However, if you want to load the set values of R35312~R35318 into RTC or read out what is in RTC onto R35312~R35318, and tune the time value etc, then the setting must be done using the special relays (M9179 and M9180) for RTC access. Below is an explanation of the access and adjustment procedures, and the status flag relays.

- RTC setting (R35312 ~ R35318→RTC):

  The setting action is only executed once at the moment that relay M1952 goes from 1→0 (falling edge).

Note: If you want to load the set values into RTC, you must first make M9179 as 1 and then load the set values into R35312~R35318. The loading of the set values into R35312~R35318 can be done via MOVE instruction. However, you must first halt the RTC read out (make M9179 as 1), otherwise the data that you just wrote into R35312~R35318 will immediately be overridden by the time data being read back from RTC in the opposite direction.

- RTC readout (RTC→R35312~R35319) :

  Whenever the M9179 relay is 0 (RTC timing active). With every scan, CPU will take the time value data within RTC and move it to R35312~R35319. When it is 1, it will not read out. In this case R35312~R35318 can load in the set values and they won't be overridden.

- ±30 second adjustment :

  At the moment that the status of relay M9180 goes 1, CPU will check the value of the second register (R35312) within RTC. If its value is between 0 and 29 seconds then it will be cleared to 0. If its value is between 30 and 59 seconds then besides being cleared to 0, the minute register (R35313) will be increased by 1 (ie, one minute will be added). This can be used to adjust your RTC time value.

- M1981 RTC installation detecting flag：

  When RTC is fitted to the PLC, relay M9181 will be set as 1; otherwise it will be 0.

- M9182 set value error flag：

  When the time value which is set to RTC's IC is illegal, then the error flag relay M9182 will be set as 1, and the setting action will not be executed.

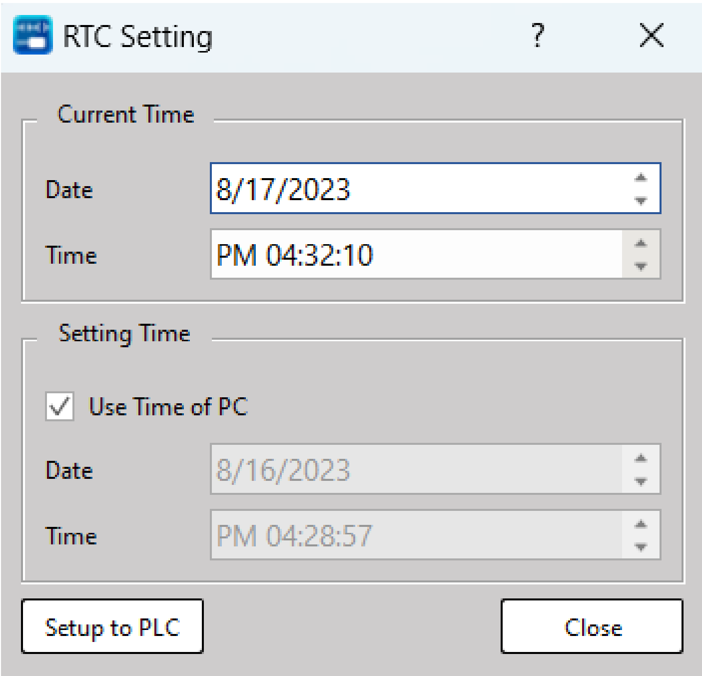
Note: M-Series PLC's Real Time Clock has already set the time, so customer don't need to set it again when using it. However, if you need to reset by yourself, in addition to using your ladder diagram program or using FP-07C and using the control of M9179 as described in item 1 RTC setting method to make settings, on the UperLogic package software, we provide more convenient setting function. As long as you enter the time you want to set, press the set button to complete the setting, and you don't need to deal with the control of M9179, please refer to the instructions of the Ladder Master package software.

Setting the calendar with UpperLogic

Click the "calendar" Item which in Tool bar : PLC

calendar

→Click right button and select "New Table"



● 〔PLC current time〕

It is means current time of PLC in on-line situation. In the "Setup" frame, if "Apply PC time" item is chosen then current time of PC will display below, press "Update PLC time" button to write PC's current time into PLC. But if "Apply PC time" item isn't chosen you can modify the Date and Time by yourself. After you change the Date and Time, press "Update PLC time" button to write the Date and time into PLC's calendar.

Amendment Record

Version	Date	Description	Page	Author
V0.0.01	2020/11/02	Version 1		
V0.0.02	2020/12/07	Version 2		
V0.0.02	2021/01/05	Version 3		
V1.0	**2021/04/26**	Version 4	502	