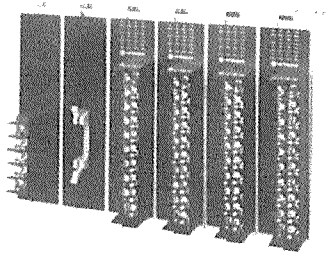
HITACHI

PROGRAMMING MANUAL

EM-II SERIES



NJI 098(X)

USING THIS MANUAL

Introduction

This manual describes the EM-II Series Programmable Controller. This manual tells how to install, program, operate, and maintain your programmable controller.

Formore information on the HITACHI product line refer to the publications listed under additional information.

Manual Contents

- Chapter 1 - Principle of PC
- Chapter 2 - Input/Output and Numbers
- Chapter 3 - Programming

PROGRAMMING

MANUAL

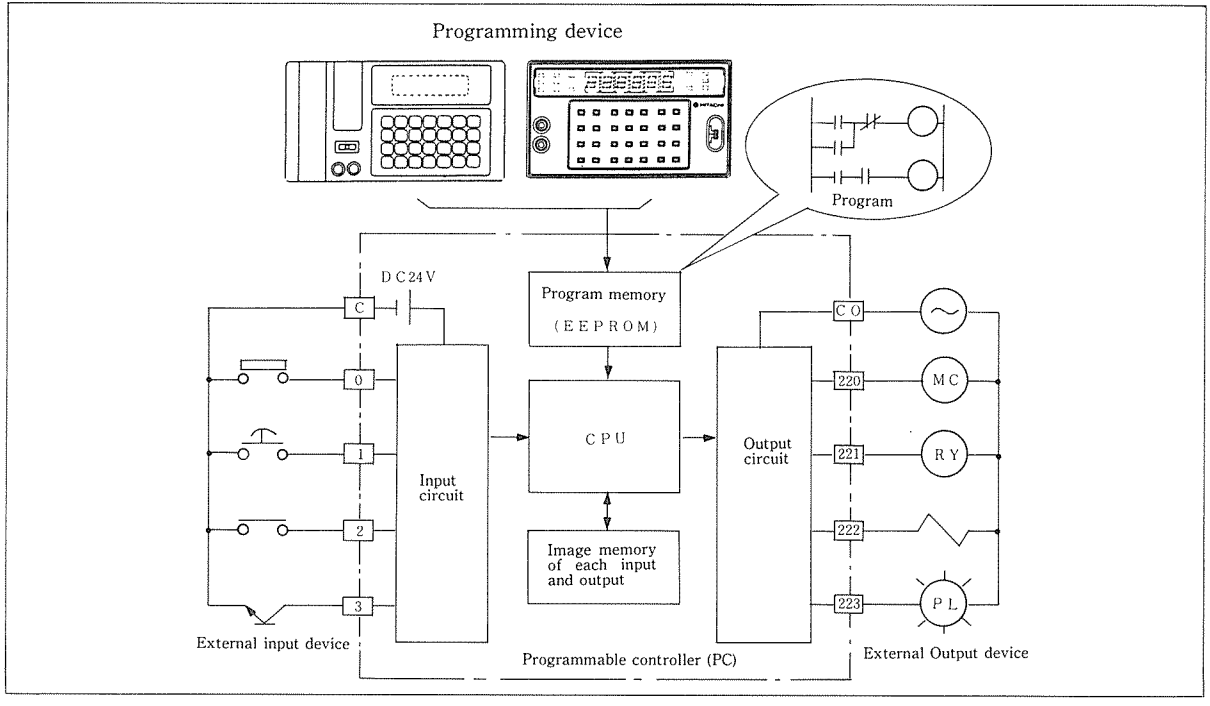
EM-II SERIES

TABLE OF CONTENTS

1 PRINCIPLE OF PC

2 INPUT/OUTPUT AND NUMBERS

3 PROGRAMMING


Programming device
Program
DC24V
C
0
1
2
3
Input circuit
Program memory
(EEPROM)
CPU
Image memory of each input and output
Output circuit
CO
220
221
222
223
MC
RY
PL
External input device
Programmable controller (PC)
External Output device

[Explanation]

1. **The programmable controller (PC) consists of CPU, program memory, image memory of each input and output, input circuit, output circuit and power supply.**

(1) The CPU is composed of a microprocessor which excutes logic and arithmetic operations, and the system software which controls PC itself.

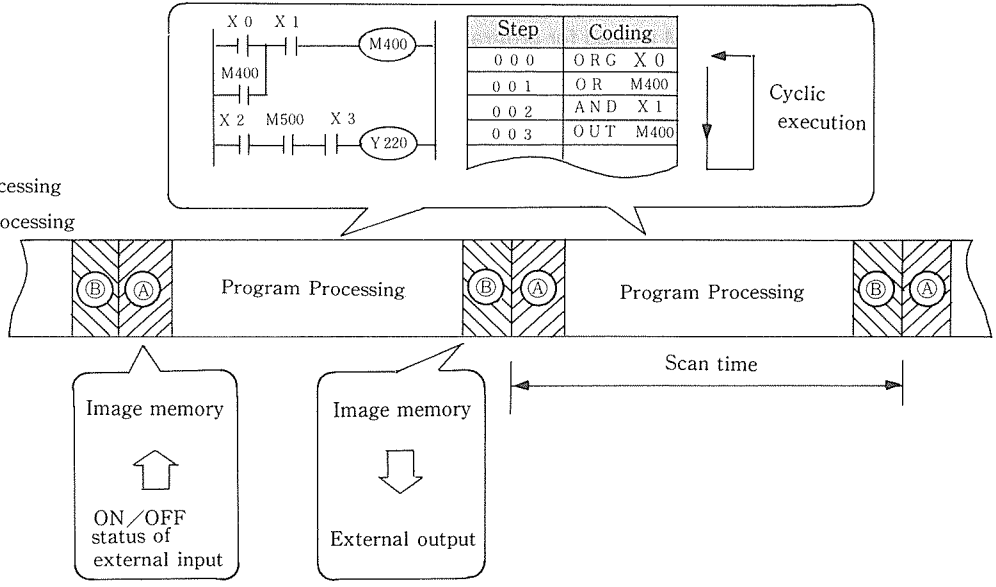
(2) The program memory is used to store a user-defined sequence program (ladder diagram). Program is to be generated by using the exclusive programming device or personnal computer.
In the E/EM series, EEPROM is used for program, so a stored program will not be lost after the PC power supply is turned off. The program can be modified easily if necessary.

(3) The image memory of each component contains data including ON/OFF status of input/output and current value of timer/counter. These data change along with program execution.

(4) The input circuit composes an interface to the external input devices (such as pushbutton switches, limit switches and proximity switches). It is electrically isolated by photocouplers.

(5) The output circuit composes an interface to the external output devices (such as electromagnetic contactors, valves and lamps).

Explanation of processing system



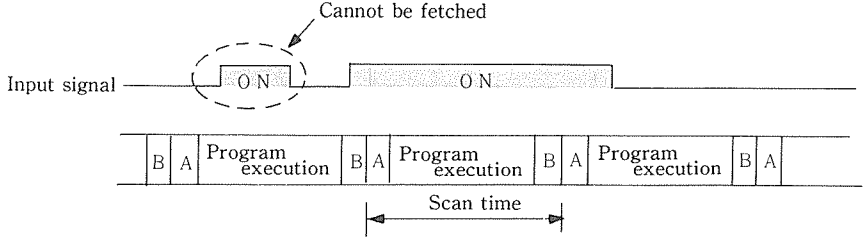
[Explanation]

1. **Scan time**

   The PC sequentially executes the written program (stored 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.

2. **Input operation**

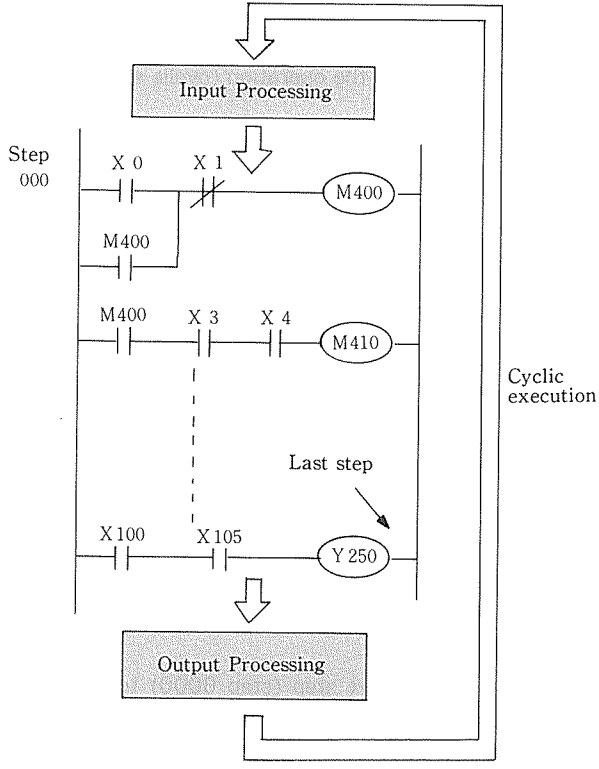
   The ON/OFF status of external input is fetched in the image memory. Even if the ON/OFF status of external input changes during program execution, the input status in the image memory remains unchanged. The status change can be read only during input processing for the next scan. So an input signal can be fetched only when its duration is longer than the time for a single scan. For fetching an input signal with a shorter duration than above, external interruption input or refresh instruction is usable.



3. **Program execution**

   A program runs sequentially from its start step (step 0000) to the last step according to the written instructions. The status of external output, internal output, etc. changes sequentially on the image memory along with program processing.

[Example]


Input Processing
Step
000
X 0
X 1
M400
M400
M400
X 3
X 4
M410
Last step
X100
X105
Y 250
Output Processing
Cyclic
execution

4. Output processing

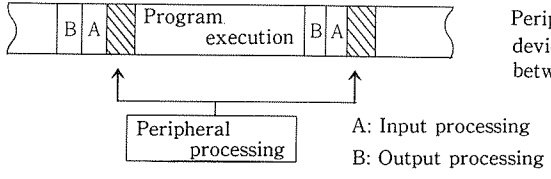
The ON/OFF status of external output on the image memory is sent to the output circuit.

5. I/O batch processing

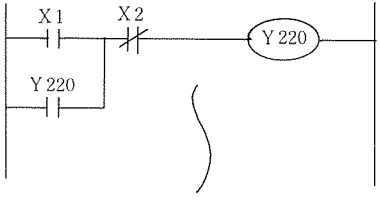
Reading the status of all external input signals at the beginning of a scan and outputting the resulting signals to an external device at the end of this scan is called I/O batch precessing. (Some PC·s use direct processing in which the external inputs are read sequentially and the result is output to the external device also sequentially.)

I/O batch processing does not cause a change in the ON/OFF status of external input and output during a scan. This makes the timing check on a program easy. Therefore, this system is widely used on small-scale PC·s. The EM-II employs this system.

6. Peripheral processing



Peripheral processing (communication) with programming device, etc.) is to be made for only 1 ms during the time between program execution and input processing.



Code list

Step	Instruction	
	Instruction word	I／O No.
0 0 0	O R G	X1
0 0 1	O R	Y220
0 0 2	AND NOT	X2
0 0 3	O U T	Y220

X, Y and M representing I/O classification cannot be keyed in. Key in numeral (s) alone.

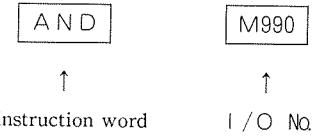
Instruction = Instruction word + I／O No.

Instruction
- 1-word instruction = 1 step ···Basic instruction, etc.
- 2-word instruction, etc. = 2 steps ···Arithmetic instruction, etc.

[Explanation]

1. Instruction

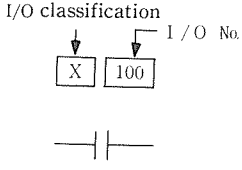
(1) An instruction is a combination of an instruction word (basic instruction, application instruction or arithmetic instruction) and the I/O number (external input, external output, internal output, timer, counter, constant or the like). Some instruction words do not require an I/O number.



(2) A single word occupies one step.
There are two kinds of instructions; one-word (16-bit) instruction and two-word (32-bit) instruction. Because the capacity of standard EM-II program memory is 3,997 words,up to 3,997 one-word instructions are programmable.

2. I/O number

A code representing the I/O classification is prefixed to each I/O number.



[I/O classification code]

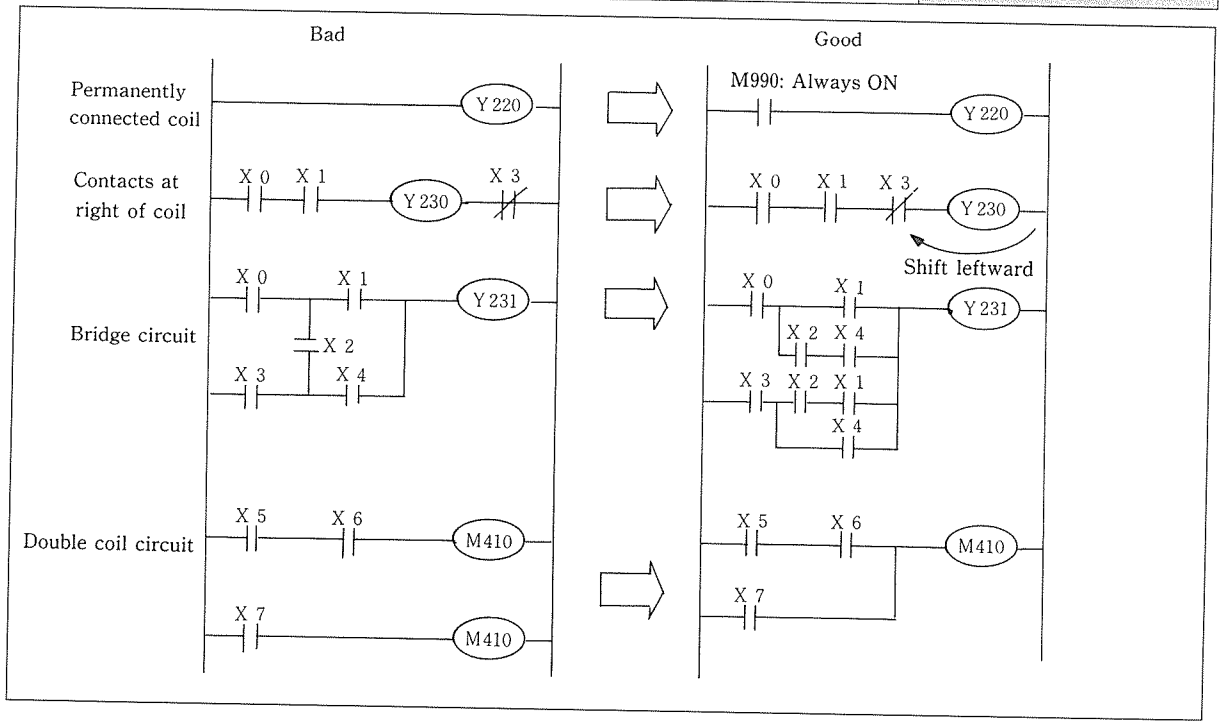
X: External input
Y: External output
M: Internal output

These can be identified by I/O number, so they need not be keyed in when using a programmer.

T/C: Timer, counter

No code: Constant, number of jump, etc.

The I/O number is determined by the assignment table (described later) so that numbers used for X, Y and M are not used twice, and the I/O classification (X, Y and M) need not be keyed in when using a programmer. However, X, Y and M are written in this manual so that the reader can easily recognize the I/O classification in the sequence program.


Bad
Good
Permanently connected coil
Y 220
M990: Always ON
Y 220
Contacts at right of coil
X 0
X 1
Y 230
X 3
X 0
X 1
X 3
Y 230
Shift leftward
Bridge circuit
X 0
X 1
Y 231
X 2
X 3
X 4
X 0
X 1
Y 231
X 2
X 4
X 3
X 2
X 1
X 4
Double coil circuit
X 5
X 6
M410
X 7
M410
X 5
X 6
M410
X 7

[Explanation]

1. **Permanently connected coil**

   An output coil cannot be connected directly to the left bus. It must be connected via the contacts of special internal output (M990) which are always closed.

2. **Contacts at right of coil**

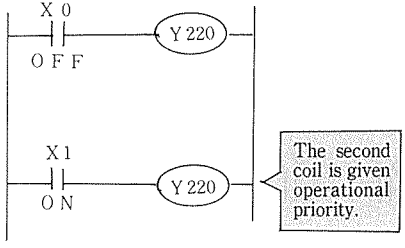
   Although the contacts of thermal relay are connected at the right of output coil in the relay sequence, it is unallowable in the PC sequence. In case such a connection is required, the contacts must be connected at the left of the coil.

3. **Bridge circuit**

   Vertical disposition of any contacts cannot be programmed. So connect the contacts in the horizontal direction.

4. **Double coil**

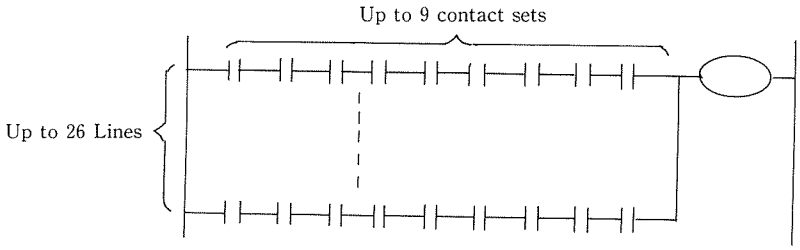
   Do not use the same output coil more than once, otherwise a double coil error (E. display) will be detected during the syntax check. However, operation will continue even if a double coil error occurs, and the output signal of the second coil will be used in the subsequent steps.



NOTE

Coil following the FUN02 (IF) or FUN;3 (IFR) is not treated as a double coil error.

5. Restrictions on number of serial and parallel contacts



(1) For entering a program with the portable graphic programmer (PGM-GPE2), the number of contact sets is restricted to 9 on each of 26 lines at maximum.

(2) For printing out data using the universal programmer (PRGMJ-R2), the number of contact sets is restricted to 8 on each of 26 lines at maximum.

(3) Although there is no restriction imposed in either vertical or horizontal direction when using the standard programmer (PGMJ) or universal programmer (PGMJ-R2), it is recommended to avoid using contacts beyond 8 sets on each line and beyond 26 lines in consideration of (1) and (2) above.

Q What is the difference between the PC and relay panel?

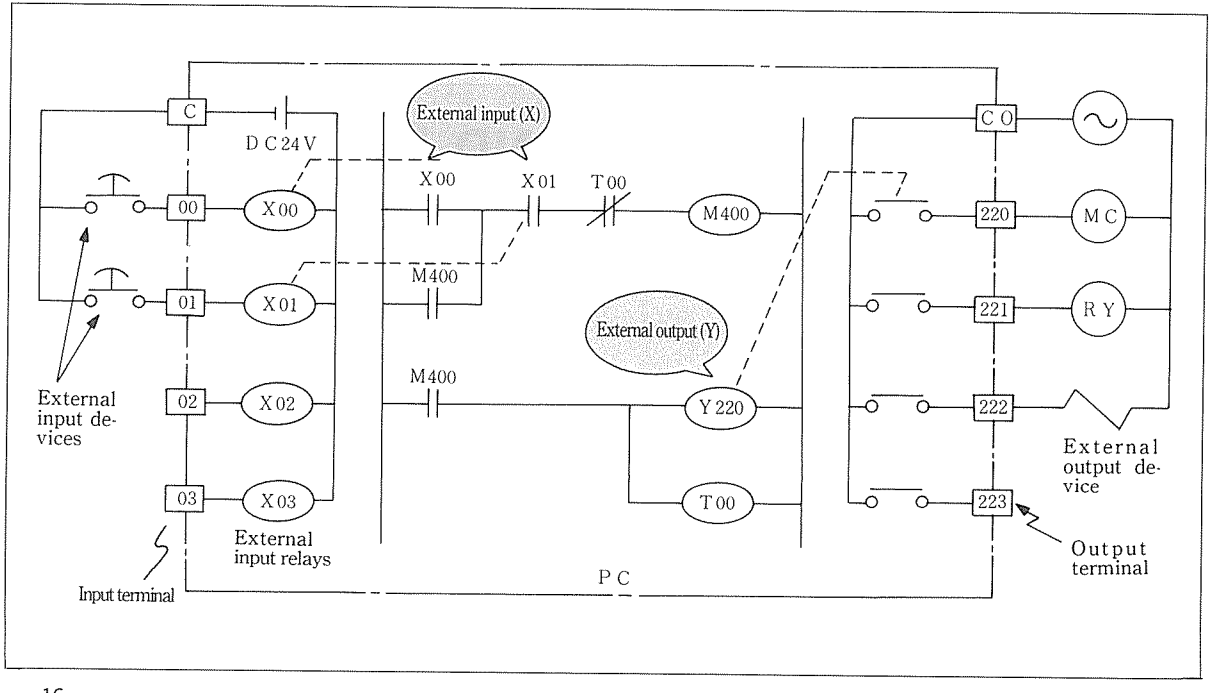
A The PC is more compact, has higher performance and more flexibility and is easier to operate than a relay panel.

Item \ System	Relay system		PC control	
Function	△	Complicated control is enabled by using many relays.	○	Control can respond to any complication through programming.
Modification of control data	×	Impossible except by rewiring.	○	Possible freely through program modification.
Reliability	△	No problem in normal use. However, poor contact may occur and the servise life is limited.	○	Highly reliable because semiconductors are used in key components.
Universality	×	Complete device serves for only one purpose.	○	Usable for any control through programming.
System expandability	△	Difficult because modification is required.	○	Freely expandable within capacity.
Ease of maintenance	△	Periodic maintenance and replacement of service parts are required.	○	Repair is possible inside each unit.
Necessary technical understanding	○	Popular, widely known, simple and easy to understand	△	Programming software rules must be learned.
Equipment size	△	Usually large	○	Remains compact for even complicated and sophisticated control
Design and manufacturing periods	×	Many drawings must be prepared, and a long time is needed for arranging parts and performing assembly test.	○	Design is easy even for complicated control. Manufacture can be completed in a shorter time period. Hardware is usable for general purposes (ready-made products).

Sign: ○ Very good △ Good × Poor

1	PRINCIPLE OF PC	
2	INPUT/OUTPUT AND NUMBERS	
3	PROGRAMMING	3.1 Basic Instructions
		3.2 Application Instructions (I)
		3.3 Arithmetic Instructions
		3.4 Application Instructions (II)

External inputs(X), external outputs(Y)	Internal outputs(M)	Timer(T)	Counter(C)	Instruction words and I/O numbers	Arithmetic register
16	18	27	36	40	47



[Explanation]

1. **External input (X)**

   Input sensors, such as limit switches, pushbutton switches, proximity switches and photoelectric switches are external input devices of the PC. They are connected to the input terminals of the PC and drive the external input relay (X) in the PC.

   (The ON/OFF status of each external input device is fetched in the image memory.)

   External input relay is referred to as an external input (X) hereinafter. The external input (X) has many normally open contacts ("a" contacts) and normally closed contacts ("b" contacts). They are used for generating the sequence in the PC.

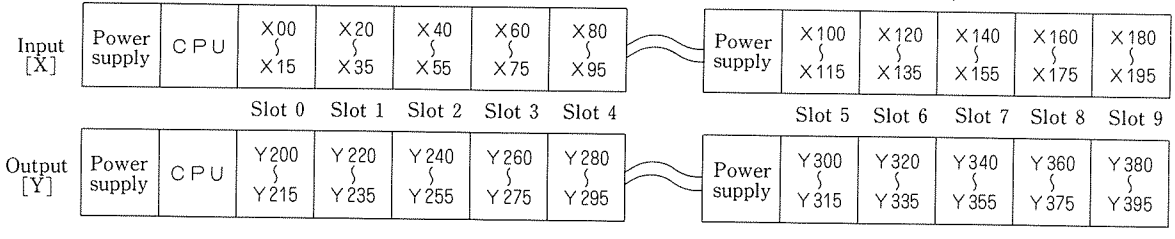
2. **External output (Y)**

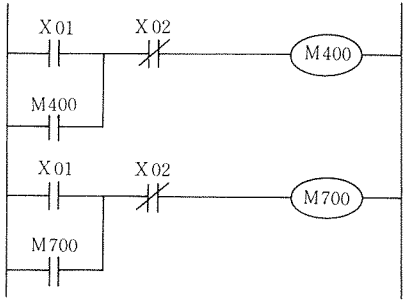
   Electromagnetic contactors, valves, indicator lamps, etc. , are external output devices of the PC. These devices are connected to the PC output terminals and driven via the contacts of external output relays in the PC. External output relay is referred to as an external output (Y) hereinafter.

   The external output (Y) also has many normally open contacts ("a" contacts)and normally closed contacts ("b") contacts). They are used for generating the sequence in the PC.

3. **I/O number assignment**

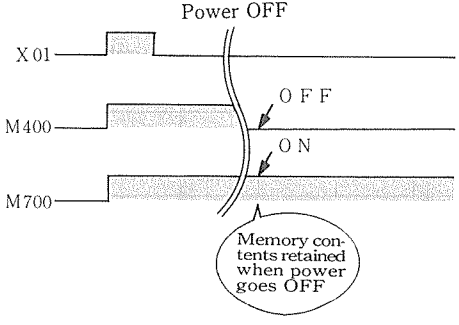
   I/O numbers are assigned according to the slot position of base (BSM-3 to 9). When mounting an input module in slot 0 in the example below, input numbers X00 to X15 are assigned, and output numbers Y220 to 235 are assigned when mounting an output module in slot 1 in the same example.





Internal output (M) [non-retentive] ····· M400 to M655 (256 points)
Contents are lost when power goes OFF.

Internal output (M) [retentive] ····· M700 to M955 (256 points)
Contents are retained when power goes OFF.



When composing a self-holding circuit for retentive internal output (M700 to M955), it is usable as a keep relay.

[**Explanation**]

1. The internal output (M) is equivalent to an auxiliary relay in relay sequence. It has many normally open contacts ("a" contacts) and norlally closed contacts ("b" contacts). They are used for generating an internal PC sequence.
2. There are two kinds of internal outputs (M) ;
   non-retentive (memory cleared to zero because of status change from power OFF to ON, stop to run or run to stop) and retentive (memory not cleared to zero regardless of status change from power OFF to ON, stop to run or run to stop). This is discriminated in I/O number.
3. When composing a self-holding circuit for retentive internal output (M700 to M955), it is usable as a keep relay.
4. Internal outputs with special function (M960 through M991)
   There are special internal outputs which function as a clock or a flag for a failure. Table 2-3 details the functions of each special intemnal output.

Table 2-1 shows how the external input (X), external output (Y), internal output (M) and timer/counter (T/C)) are assigned.

Table 2-1 Assignment of I/O Numbers (1/2)

Classification		Slot	Number: Input module	Number: Output module	Remarks
External input (160 points) or external output (160 points)		0	X 0~X 15	Y 200~Y 215	○ Decimal numbers ○ Numbers 0 to 15 are assigned when mounting a 16-point input module in slot 0. ○ Numbers 200 to 215 are assigned when mounting a 16-point output module in slot 0. ○ Numbers 8 to 15 are omitted when mounting an 8-point module. ○ For assignment of special modules, refer to Section 8.
		1	X 20~X 35	Y 220~Y 235	
		2	X 40~X 55	Y 240~Y 255	
		3	X 60~X 75	Y 260~Y 275	
		4	X 80~X 95	Y 280~Y 295	
		5	X 100~X 115	Y 300~Y 315	
		6	X 120~X 135	Y 320~Y 335	
		7	X 140~X 155	Y 340~Y 355	
		8	X 160~X 175	Y 360~Y 375	
		9	X 180~X 195	Y 380~Y 395	
Internal output	Non-retentive memory at power failure (256 points)		M400~M655		○ Decimal numbers ○ Each number has a data capacity of 8 bits. M400 b7 b6 b5 b4 b3 b2 b1 b0 M401 b7 b6 b5 b4 b3 b2 b1 b0 ○ The bit handling instruction determines ON/OFF status of b7. ○ The word handling instruction handles 8-bit data of M400 and that of M401, 16 bits in total, when No. 400 is designated.
	Retentive memory at power failure (256 points)		M700~M955		
	Special function (32 points)		M960~M991		○ All bit data ○ Detailed in Table 3-3.

Table 2-1 Assignment of I/O Numbers (2/2)

Classification	Slot	Number		Remarks
		Input module	Output module	
Timer and counter (96 points in total)	Coil contacts	T/C 0 ~ T/C 95		○ Decimal numbers ○ Timer and counter share the same number. ○ Up-timer and up-counter, respectively ○ 100 is added to timer/counter number (2-digit) for representing a current value, and 200 is added for indicating preset value. ○ States of coil and contacts are shown by bit data. ○ Current value and preset value are of 16 bit data.
	Current value	T/C 100 ~ T/C 195		
	Preset value	T/C 200 ~ T/C 295		

Table 2-2 lists each range of constent and argument used in instrucctions such as AJMP and MODE.

Table 2-2 Each Range of Constant and Argument

Classification		Range	Remarks
Constant	Word constant	0000 H ~ 9999 H	The hexadecimal code H is not suffixed at the time of program entry. (Example) FUNO. 1234 (1234H→AR)
		0 ~ F F F F	This constant is designated in a decimal number because the programmer does not have keys A to F which are indispensable for hexadecimal designation. Entry is possible in up to 3 digits. Effective range of decimal constant: 0 to 999 (Example) FUN51 427 (AR+1ABH → AR) (Decimal 427=hexadecimal 1ABH)
	Byte constant	00 ~ F F	This constant is also designated in a decimal number because the programmer does not have keys A to F which are indispensable for hexadecimal designation. Effective range of decimal constant: 0 to 255 (Example) FUN50 255 (FFH → ARL) (Decimal 255=hexadecimal FFH)
	No. of bits	0 ~ 255	Used for FUN72 and FUN73. (Example) FUN72 5 (AR is masked by 5 bits from the left.)
Argument		0 ~ 63	Used as an argument of FUN08 (AJMP), FUN09 (AJEND), FUN42 (CALL), FUN43 (SB), FUN93 (INT) and FUN97 (MODE). (Example) FUN08 63 (AJMP63)

Table 2-3 Function of Special Internal Output (1/4)

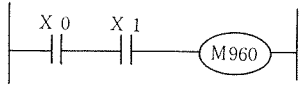
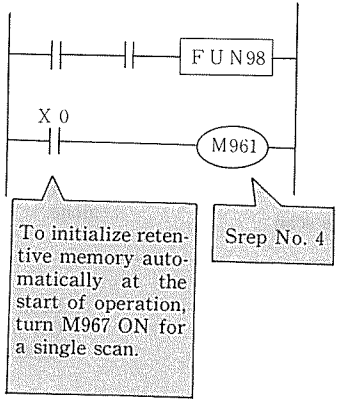
No.	Function	Description
M960	All outputs OFF	When M960 is switched ON by the program, all external output signals go OFF except for the RUN contacts.  ○ Suppose that an error program is written. (The X0 and X1 are not closed simultaneously during normal operation.) As a result, M 960 is switched ON. In this status, the PC judges that there is a system error and it switches all output signals OFF. However, program operation does not stop. ○ Eliminate the cause of the error and turn on power supply again.
M961	Initializing retentive memory	 ○ In the sysytem shown in the figure, retentive memory is or is not initialized depending on whether X0 is ON or OFF at the start of operation. XO: ON……Retentive memory is initialized when power is switched ON. XO: OFF……Retentive memory is not initialized when power is switched ON. ○ Retentive memory is initialized only at the start of operation. During operation, it is not initialized even if M961 is switched ON. ○ M961 coil operates only when it is written in step 4. It is invalid when it is weitten in any other step.

Table 2-3 Function of Special Internal Output (2/4)

No	Function	Description
M962	Cyclic oscillation	t: Period of one scan (scan time) Signal goed ON/OFF alternately for each scan.
M963	0.1 sec clock	M963: 0.05 sec / 0.1 sec M964: 0.5 sec / 1 sec
M964	1 sec clock	
M965	10 sec clock	M965: 5 sec / 10 sec M966: 30 sec / 1 min
M966	1 min clock	
M969	10 ms clock	5 ms / 10 ms
M967	ON for a single scan after start of operation	Start of operation One scan To initialize all volatile memories at the start of operation, use M967 in combination with M961. To initialize memory individually, use M967 alone.
M968	1000-scan cycle	t 1000 × t t: Scan time ON once every 1,000 scans. Used for measuring scan time.

Table 2-3 Function of Special Internal Output (3/4)

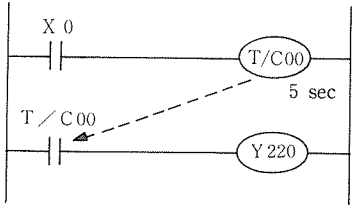
<table>
<tr><th>No.</th><th>Function</th><th>Description</th></tr>
<tr><td>WM970</td><td>System error factor</td><td>If system error occurs (when ERR lamp comes on), an error code within 0 to 65535 is displayed. Details are given number system Error codes in section 7. The code cannot be cleared by turning on power supply again.</td></tr>
<tr><td>WM972</td><td>Program counter at occurrence of system error</td><td>If system error, occurs, count on the program counter of microprocessor is displayed.</td></tr>
<tr><td>WM974</td><td>Designation of read address at occurrence of system error</td><td rowspan="2">If system error occurs, data at the address designated by WM 974 is presented in M976.</td></tr>
<tr><td>M976</td><td>Data readout at occurrence of system error</td></tr>
<tr><td>M977</td><td>Registration of system ROM sum</td><td rowspan="2">System ROM sum appears in WM978 only when this I/O is ON upon turning on power supply.</td></tr>
<tr><td>WM978</td><td>System ROM sum</td></tr>
<tr><td>M989</td><td>System attribute</td><td>Systm attribute appears in each bit of b7 to b5. Bits b4 to b0 are undefined.
<table>
<tr><th>Bit</th><th>At 0</th><th>At 1</th></tr>
<tr><td>b7</td><td>CPM－E2</td><td>CPM－E3</td></tr>
<tr><td>b6</td><td>9600bps</td><td>4800bps</td></tr>
<tr><td>b5</td><td>RUN instruction valid</td><td>RUN instruction invalid</td></tr>
</table>
Used for hardware check</td></tr>
</table>

Table 2-3 Function of Special Internal Output (4/4)

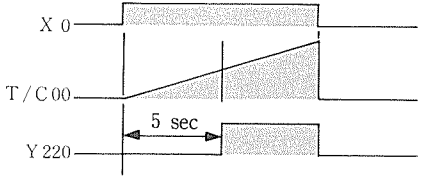
No.	Function	Description
WM980	Syntax error factor	If syntax error is detected in the check specified by a peripheral or in the check before start of operation, an error code within 0 to 65535 is displayed. The code cannot be cleared by turning on power supply again.
WM982	Scan time	The latest scan time is indicated in steps of 10 ms, though the first scan is shown as 65535 ms. Indication contains an error of ±10 ms. Unit is millisecond (ms). (Indicated as 0, 10, 20, ms……)
WM984	Max. scan time	Of scan times after the start of operation, the maximum time is displayed in steps of 10 ms, though the first scan is shown as 0 ms. Indication contains an error of ±10 ms. Unit is millisecond (ms). (Indicated as 0, 10, 20, ms……)
M990	Normally ON	Always ON irrespective of run/stop status.
M991	ON during run	ON during run and OFF during stop

M986 through M988 are for functional expansion and unused (undefined) by the system.

Timers: T/C00 to T/C95 (96 points shared with counter)



Coding		Remarks
ORG	X 0	(1) Write a decimal point (.) before the preset value. (2) Write the I/O classification code "T/C."
OUT	T / C00.5	
ORG	T / C00	
OUT	Y 220	



Time lapse is fetched in incremental mode.

Timer contacts close when the current time value reaches the preset value.

[Explanation]

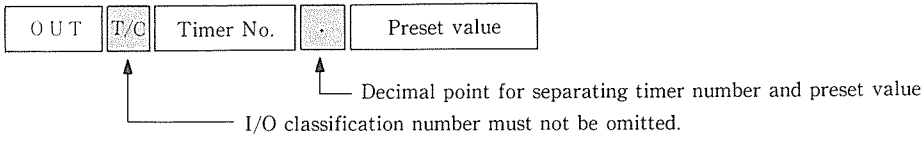
1. **Kinds of timer**

(1) On-delay timers are used. In the above sequence, the timer coil T/C00 is excited when input X0 turns ON. After 5sec, the timer contacts close. There are many timers with "a" and "b" contacts. They are used for generating a sequence in the PC.

(2) The same data area is shared by timers and counters, a total of 96 points (T/C00 throuh T/C95). A number used for a counter cannot be used for a timer.

2. **Key input of timer**

For specifying a timer coil using the programmer, enter the timer number (1 or 2 digits), a decimal point (.) as a separator and the preset value in this order.



3. Time base

The timers have two time bases: 0.01 and 0.1 sec. Time base is automatically selected according to the key-in method.

Time base	Key-in method	Preset value range
0.1sec	OUT [T/C] [Timer No.] [·] [6] [3] [·] [5] → 63.5sec	T／C 0～ 9…0.1～999.9sec T／C 10～95… { 0.1～99.9sec / 1～999sec
	OUT [T/C] [Timer No.] [·] [7] [7] [0] → 770.0sec	
0.01sec	OUT [T/C] [Timer No.] [·] [0] [·] [5] [5] → 0.55sec	T／C 0～95…0.01～9.99 (settable only in 3 digits)

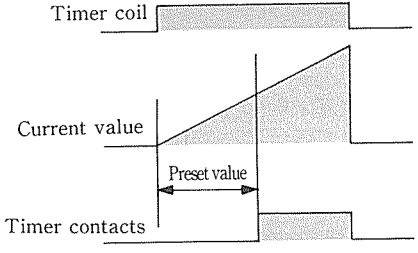
4. Preset value

Up to 10 timers/counters (T/C0 to T/C9) can be set using 4 digits (except for the timer adopting 0.01 sec time base which must be set using 3 digits).

Up to 86 timers/counters (T/C10 to T/C95) can be set using 3 digits.

5. Current value

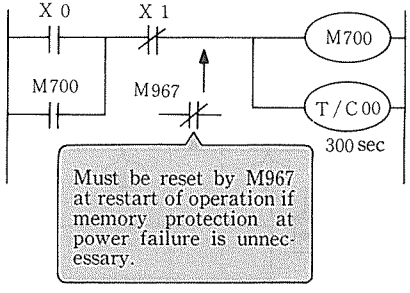
Each timer operates in the incremental mode. It starts timing when the timer coil is energized. When the current valuue reaches the present value, the timer contacts close. When the timer coil is deenergized, the current value is reset to 0.

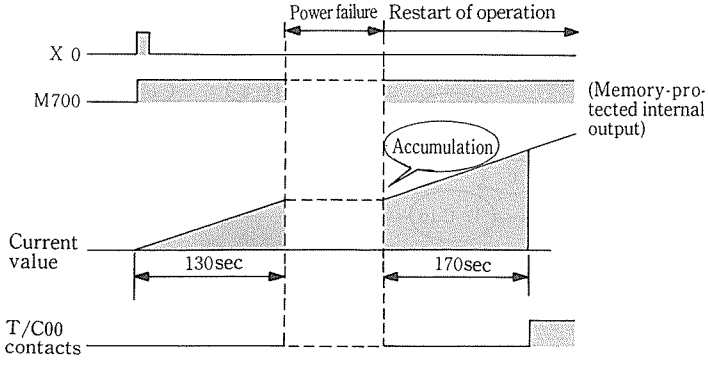


NOTE

The current value of each timer is retained in memory even if power failure occurs or when power supply is turned off. When combining the timer with the retentive internal output, an accumulation timer can be composed.

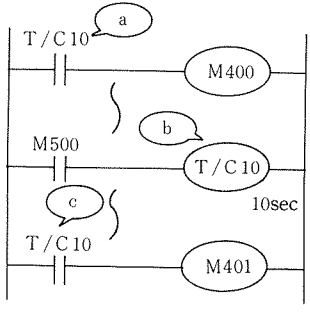
[Example]

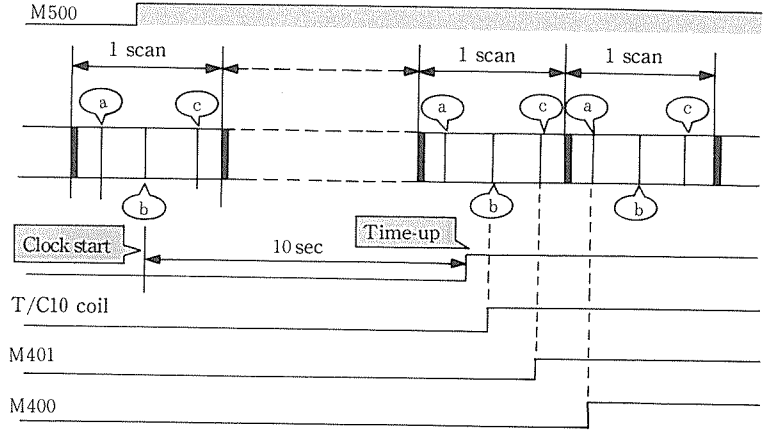

X 0
X 1
M700
M700
M967
T / C 00
300 sec
Must be reset by M967 at restart of operation if memory protection at power failure is unnecessary.


Power failure
Restart of operation
X 0
M700
(Memory-protected internal output)
Accumulation
Current value
130sec
170sec
T/C00 contacts

6. Contacts operation timing and accuracy

[Example]





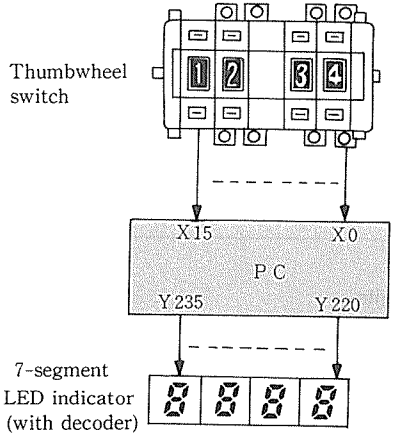
The clock starts when the timer coil is energized (time point (b)). When the coil instruction is executed after time-up the output contacts close.

Condition	Timer starts by other than external input signal		Timer starts by external input signal.	
	Timer contacts (a) before coil	Timer contacts (b) after coil	Timer contacts (a) before coil	Timer contacts (b) after coil
Timer accuracy	+2 scans	+1 scan	Input fethch delay (4 ms single scan filter) +2 scans	Same as left + 1 scan

Total timer accuracy

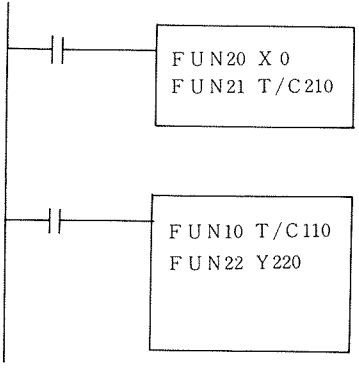
Preset time +2 scans
- time base (0.01 sec or 0.1 sec)

7. Handling timer preset value and current value in arithmetic instructions in application



The preset value of a timer can be changed by using the thumbwheel switch, and the current value of a timer can be read on the 7-segment LED indicator. An example of program is shown below. The table below lists the number assignment when using the timer preset value and current value in the arithmetic operation.

Segment	Assignment No.	Remarks
Current value	T／C100～T／C195	Add 100 to timer coils T/C00 to T/C95.
Preset value	T／C200～T／C295	Add 200 to timer coils T/C00 to T/C95.



X0 through X15(thumbwheel switch data) → AR

AR → timer T/C preset value

(T/C 210 programmed for preset value)

} The timer preset value can be changed by using the thumbwheel switch.

Current value of T/C10 timer → AR

(T/C110 programmed for current value)

AR → Y220～Y235

} The timer current value is read on the indicator.

The timer preset value and current value are data to be processed in blocks of 16 bits as shown below.

Segment	Kind of timer	Data to be processed by arithmetic instruction
Preset value and current value	0.1 sec timer	b 15 … b 0 2 6 4 5 Indicates 264.5 sec. …… BCD 4 digits The least significant digit represents 0.1 sec order.
	0.01 sec timer	b 15 … b 0 F 0 5 5 Indicates 0.55 sec. …… BCD 3 digits The most significant digit stands for "F" (0.01 sec timer). The least significant dight represents 0.01 sec order.

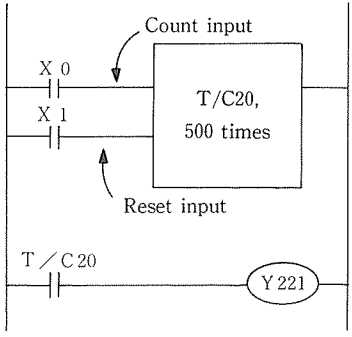
Q What is EEPROM?

A The memory device (EEPROM) of the E series does not require a battery. So it is easy to maintain.

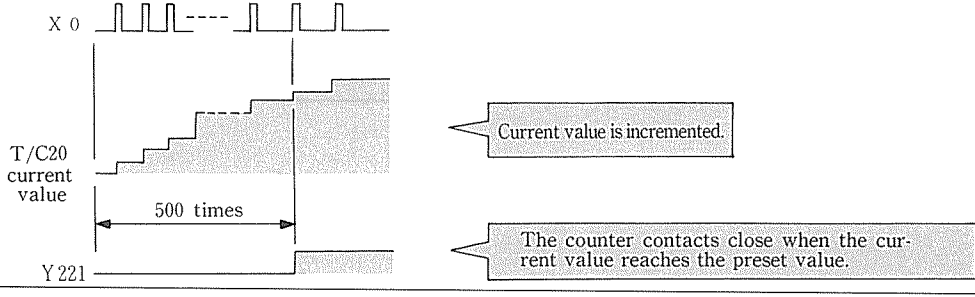
The EEPROM does not require battery backup for program. Hence, program will not be lost because of the end of useful life or abnormal discharge of a battery, and there is no need for tiresome battery replacement. Despite being a ROM, the EEPROM allows a program to be written and erased electrically like a RAM without using a ROM writer or UV eraser.
Compare the EEPROM with the already popular RAM and EPROM for an easier understanding.

Kind of memory	Program write	Program erase	Program protection reliability	Program store
EEPROM	Can be written electrically	Can be erased electrically	Intermediate	Battery unnecessary
EPROM	ROM writer necessary	UV eraser required	High	Battery unnecessary
RAM	Can be written electrically	Can be erased electrically	Low	Battery necessary

Counter: T/C00 through T/C95(selectable to function as timer or counter, total of 96)



Code		Remarks
O R G	X 0	(1)The reset input is given in the form of STR instruction. (2)Place a decimal point(.)before the preset value. (3)Use the I/O claasification code "T/C."
S T R	X 1	
O U T	T／C 20.500	
O R G	T／C 20	
O U T	Y 221	



[Explanation]

1. **Kind of counter**

(1) An up-counter is used. In the above sequence, the counter T/C20 counts ON/OFF cycles of input X0. When the count reaches 500, the counter contacts close.
The counters can provided with any number of "a" and "b" contacts.
They are used for generating sequences in PC.

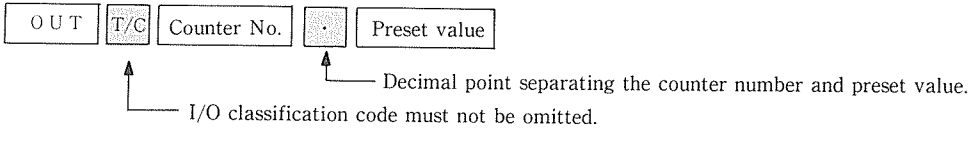
(2) Timers and counters share the same data area. There are 96 timers/counters in total (T/C00 through T/C95).
Once a T/C number is assigned to a timer, it cannot be reused for a counter.

(3) When the reset input turns ON, the counter is reset and the current to 0.

2. **Counter key input**

(1) Program the count input and reset input in this order. Reset input must be programmed by an STR instruction.

(2) A counter preset value can be entered in the same way as for a timer.



3. **Preset value**

Up to 10 timers/counters (T/C0 to T/C9) can be set using 4 digits.

Up to 86 timers/counters (T/C10 to T/C95) can be set using 3 digits.

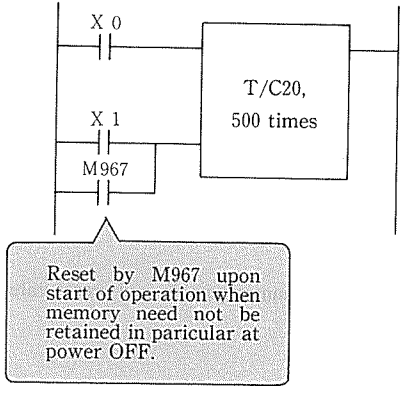
4. **Current value**

The current value of each counter is incremented by 1 (one) whenever the count input turns from OFF to ON. The counter contacts close when the current value reaches the preset value.

When the reset input turns ON, the current value is reset to 0.

The current value of the counter is retained in memory even if power in turned OFF.

[**Example**]

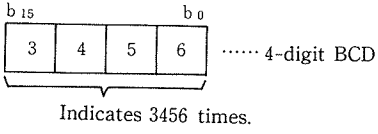


If the retentive data is unnecessary, use the special internal output M967, which turns on a single scan at start of operation. Program as shown at left.

5. Handling the counter preset value and current value in the arithmetic instructions.

When using a combination of counter preset value and current value in arithmetic instructions, the current value must be equal to the counter coil number (T/C0 throuh 95) incremented by 100, namely T/C100, to T/C195. The preset value must be equal to the coil number incremented by 200, namely T/C200 to T/C295.

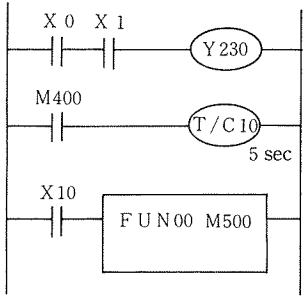
The counter preset value and current value are 16-bit data (4-digit BCD value) and processed as shown in the table below.

Item	Assingnment No.	Data to bi processed by arithmetic ininstruction
Current value	T/C100 through T/C195 (equal to counter coil numbers T/C0 to T/C95 incremented by 100)	
Preset value	T/C200 through T/C295 (equal to counter coil numbers T/C0 to T/C95 incremented by 200)	

Instruction can be in the form of bits, words and bit data handled as words.

1. Bit-type operating instruction

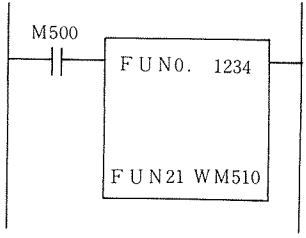
[Example]



A bit-type operating instruction affects only a single set of contracts (via coil) as shown in the figure.
The basic instructions are all bit-type instructions.

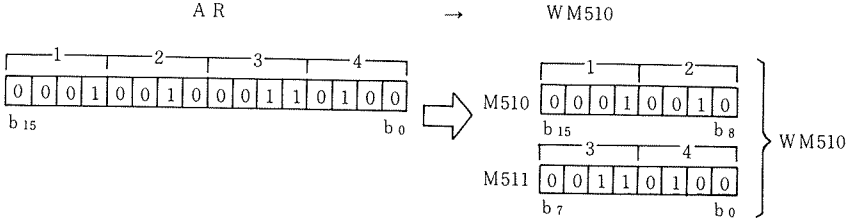
2. Word-type operating instruction

[Example]



Constant 1234H → AR

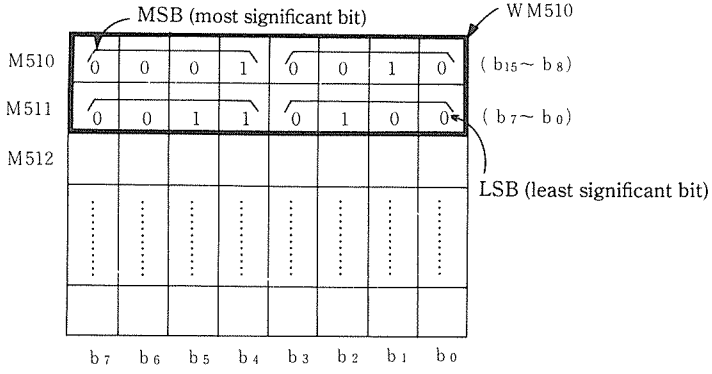
Aword-type operating insturction handles 16 bits as one word.



(1) In the above circuit, the constant 1234H is stored in the AR (arithmetic register) by "FUNO. 1234." The AR data is output to the 16 bits of M510 and M511 by FUN21 WM510."

(2) When an internal output number is specified by a word-type operating instruction, it is handled as 16-bit data in the following way. The 8-bit data of the specified internal output (M510 in the above example) is taken as b8 through b15, while that of the next internal output (M511) Is taken as b0 through b7.

[**Example**]

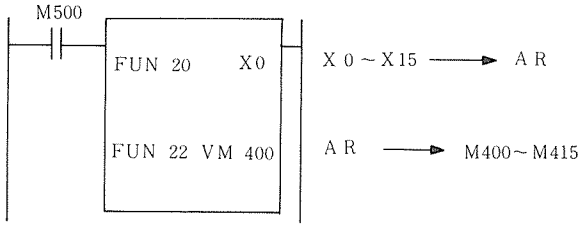


(3) The timer and counter preset values, current values and constants (0000H to 9999H) are all 16-bit data. So they are directly processed as a word when specifying their numbers by a word-type operating instruction.

3. Handling of bit-type data as a word

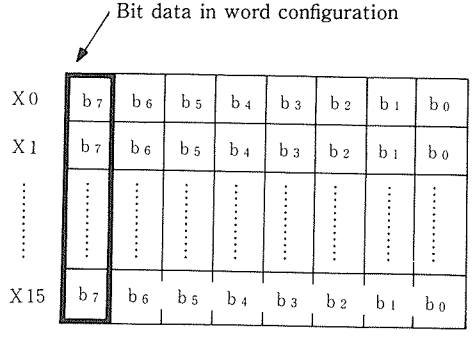
An instruction that treats 16 one-bit data (X0 to X15) as a single word is called a "word-type instruction for bit data."

[Example]



(1) In the above sequence, the 16-bit data of X0 through X15 is stored in the AR by the "FUNO20 X0" instruction, and the data in the AR is output to M400 to M415 by the "FUN22 M400" instruction.

(2) When external I/O or internal output number is specified by a word-type instruction for bit data, only the most significant bit (b7) of the 16 points (namely, 16 bits) starting from the specified No. (X0 and M400 in the above example) is handled as a single-word data.

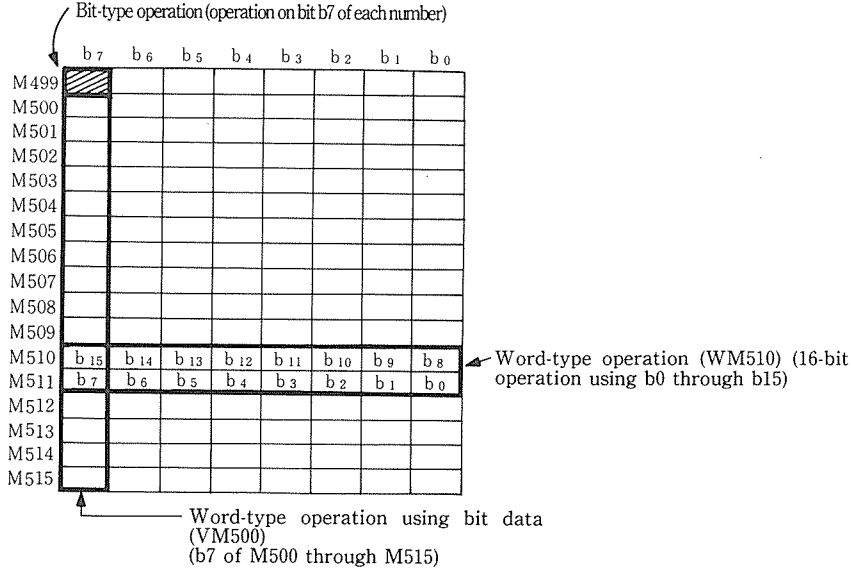


Example of external input (Word configuration remains the same in case of external output and internal output.)

(3) Word-type instructions for bit data are used for connecting the thumbwheel switch,etc. as an external input device for storing BCD data.
They are also used to output data in the AR to an external output terminal. (See the section "Handling of the timer preset value and current value by arithmetic instructions.")

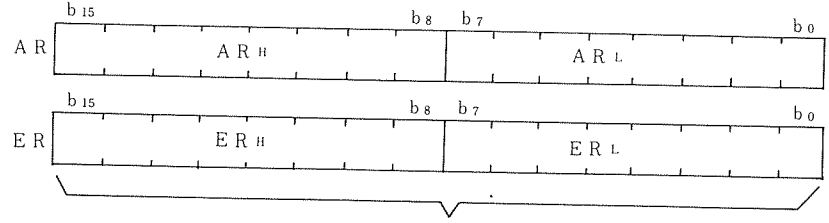
4. Summary of instructions

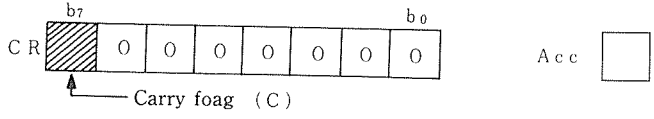
[Example]

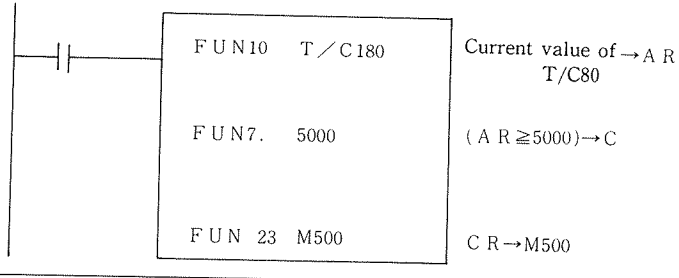


(1) Internal outputs are all 8 bits long.
In word-type operation, a total of 16 bits in the specified internal output number and the next number are handled. This data is given an element code WM. WM510 in the above example consists of M510 and M511 (b0 to b15).

(2) In the word-type operation using bit data, the most significant bit (b7) of sixteen 8-bit data starting from the specified number is handled as a single-word data.
This word data is made up of the 16 bits in the verticcal direction. Hence it is given an element code VM.


b 15
b 8
b 7
b 0
AR
AR H
AR L
ER
ER H
ER L
16 bits


b7
b 0
CR
0 0 0 0 0 0 0
Carry foag (C)
Acc


FUN10 T/C180
Current value of T/C80 → AR
FUN7. 5000
(AR≧5000)→C
FUN 23 M500
CR→M500

[**Explanation**]

1. The registers of EM-II series come in 4 kinds below.
   (1) AR: Arithmetic Register used for instructions. It has a 16-bit configuration.
   (2) ER: Expansion Register used for storing upper word resulting from multiplication and remainder of division. It has a 16-bit cofiguration.
   (3) CR: Carry Register. Carry flag (C) turns to "1," for example when the condition for comparison is satisfied. Bits b0 to b6 are always "0."
   (4) Acc: 1-bit register which automatically changes along with execution of a basic instruction such as ORG or AND.
2. Data in the AR, ER and CR are cleared every time scan starts and they change in response to the processing of arithmetic instructions.

1 PRINCIPLE OF PC

2 INPUT/OUTPUT AND NUMBERS

3 PROGRAMMING

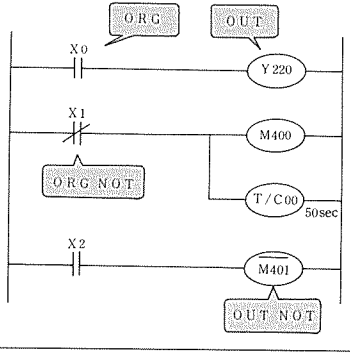
- 3.1 Basic Instructions
- 3.2 Application Instructions (I)
- 3.3 Arithmetic Instructions
- 3.4 Application Instructions (II)

Table 3-1 Basic Instructions

Instruction	Symbol	Function	Coponent	No. of words	Change in register				Reference page
					AR	ER	C	Acc	
ORG		Connection of normally open contacts ("a" contacts) to bus	X, Y, M, T/C 0~95	1	●	●	●	I	51
ORG NOT		Connection of normally closed contacts ("b" contacts) to bus		1	●	●	●	I	51
STR		Start of branching normally open contacts ("a" contacts)	X, Y, M	1	●	●	●	I	57
STR NOT		Start open branching normally closed contacts ("b" contacts)	T/C 0~T/C95	1	●	●	●	I	57
AND		Serial connection of normally open contacts ("a" contacts)	X, Y, M	1	●	●	●	I	53
AND NOT		Serial connection of normally closed contacts ("b" contacts)	T/C 0~T/C95	1	●	●	●	I	53
OR		Parallel connection of normally open contacts ("a" contacts)	X, Y, M	1	●	●	●	I	55
OR NOT		Parallel connection of normally closed contacts ("b" contacts)	T/C 0~T/C95	1	●	●	●	I	55
AND STR		Serial Connection of logic block	None	1	●	●	●	I	57
OR STR		Parallel connection of logic block		1	●	●	●	I	57
OUT		Output of calculation result	Y, M T/C 0~T/C95 (with preset value)	1	●	●	●	●	51
OUT NOT		Inverted output of calculation result	Y, M	1	●	●	●	●	51

●: Register remains unchanged.
I : Register changes.

Instruction	Symbol	Meaning	Function	Component	No. of words	Change in register AR	ER	C	Acc
ORG	⊢⊣	Origin	Connection of normally open contacts ("a" contacts) to bus	X, Y, M, T/C0~95	1	•	•	•	↕
ORG NOT	⊢⫽	Inverted origin	Connection of normally closed contacts ("b" contacts) to bus		1	•	•	•	↕
OUT	—○⊣	Output	Output of calculation result	Y, M T/C0~T/C95 (with preset value)	1	•	•	•	•
OUT NOT	—⊖⊣	Inverted output	Inverted output of calculation result	Y, M	1	•	•	•	•

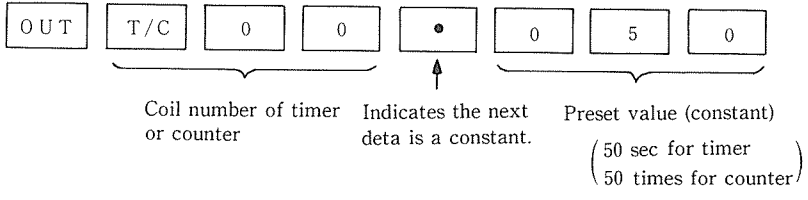


Code	Remarks
ORG X 0 OUT Y 220	Output in connection with bus
ORG NOT X 1 OUT M 400 OUT T/C00 .050	 Tmier (coil) Tmier (preset value)
ORG X 2 OUT NOT M 401	Inverted output

Element codes X, Y and M need not be keyed in.

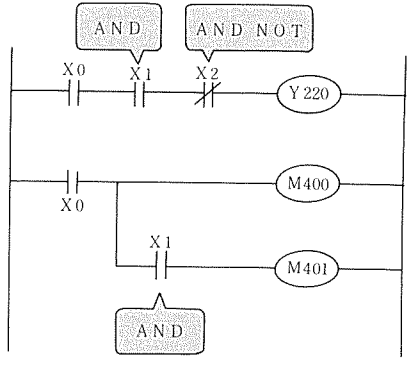
[Explanation]

1. The ORG and ORG NOT instructions are used for the contact next to the bus (at the head of circuit).
2. The OUT instruction drives each coil of external output (Y), internal output (M), timer (T) and counter (C). This instruction is not used for external input (X). The OUT NOT instruction is used for inverted output.
3. More than one OUT instruction (multiple outputs) can be used in parallel.
4. A preset value (constant) is required after an OUT instruction for a timer or counter coil.



After OUT instruction, the element number of timer/counter coil (T/C00 through T/C95), period ". " for indicating a constant and preset value must be entered in this order. This occupies a single step.

Instruction	Symbol	Meaning	Function	Component	No. of words	Change in register AR	ER	C	Acc
AND	⊣⊢	And	Serial connection of normally open contacts ("a" contacts)	X, Y, M	1	•	•	•	↕
AND NOT	⊣/⊢	Inverted and	Serial connection of normally closed contacts ("b" contacts)	T/C 0~T/C95	1	•	•	•	↕



Code		Remarks
ORG	X 0	
AND	X 1	Serial contacts
AND NOT	X 2	Serial contacts
OUT	Y 220	
ORG	X 0	
OUT	M 400	
AND	X 1	Serial contacts
OUT	M401	Cascaded output

Element codes X, Y and M need not be keyed in.

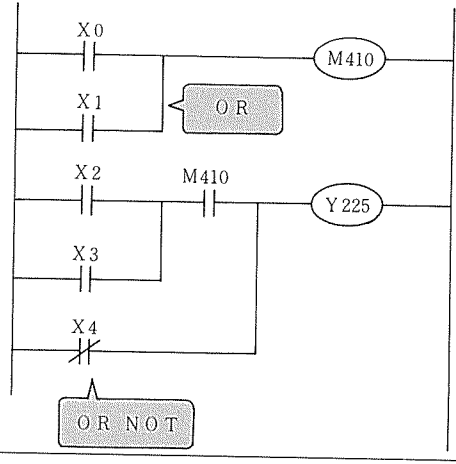
[Explanation]

1. The AND and AND NOT instructions are used for connecting a single set of contacts in series to the existing circuit.
2. Driving another coil via a contact set after OUT instruction is called a cascaded output (M400 and M401 in the figure above). Cascaded output can be repeated any number of times.

NOTE

It is recommended not to use more than 8 contact sets horizontally nor more than 26 lines vertically in a circuit, although the number of series contacts and cascaded outputs is not limited. This is because of functional restrictions on the portable graphic programmer (PGM-GPE2) and printer.

Instruction	Symbol	Meaning	Function	Component	No. of words	Change in register AR	ER	C	Acc
OR	⊣⊢	Or	Parallel connection of normally open contacts ("a" contacts)	X, Y, M	1	•	•	•	↕
OR NOT	⊣/⊢	Inverted or	Parallel connection of normally closed contacts ("b" contacts)	T/C 0~T/C95	1	•	•	•	↕

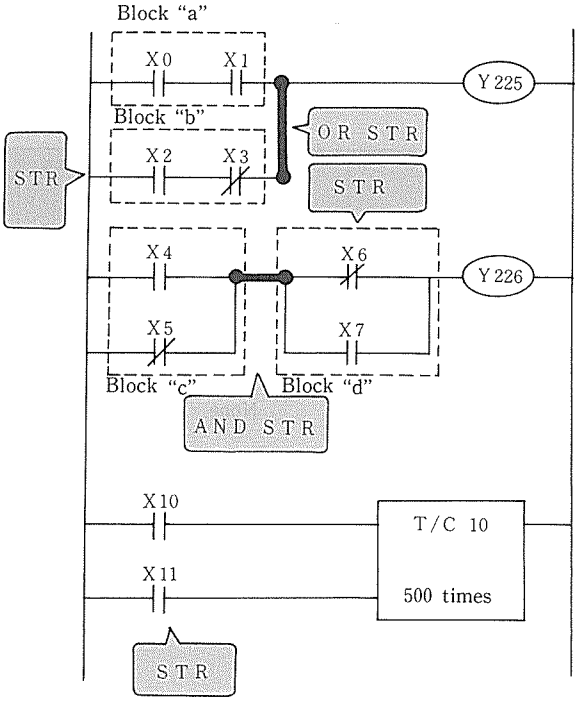


Code		Remarks
ORG	X0	Parallel connection
OR	X1	
OUT	M410	
ORG	X2	Parallel connection
OR	X3	
AND	M410	
OR NOT	X4	
OUT	Y225	

[**Explanation**]

The OR and OR NOT instructions establish a parallel connection of a contact set to the existing circuits. To connect a serial circuit block consisting of two or more serially connected contact sets (—┤├—┤├—) in parallel with another circuit, use the OR STR instruction explained later.

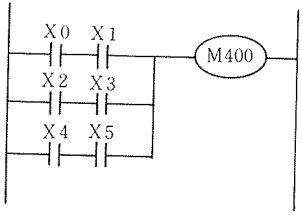
Instruction	Symbol	Meaning	Function	Component	No. of words	Change in register			
						AR	ER	C	Acc
STR		Store	Start of branching normally open contacts ("a" contacts)	X, Y, M	1	•	•	•	↕
STR NOT		Inverse of store	Start of branching normally closed contacts ("b" contacts)	T/C 0~T/C95	1	•	•	•	↕
AND STR		And store	Serial connection of logic block	None	1	•	•	•	↕
OR STR		Or store	Parallel connection of logic block		1	•	•	•	↕



Code		Remarks
ORG	X0	a: • Block "a" is programmed.
AND	X1	
STR	X2	b: • Block "b" is programmed.
AND NOT	X3	
OR STR		a + b • Blocks "a" and "b" are combined by OR STR.
OUT	Y225	
ORG	X4	c: • Block "c" is programmed.
OR NOT	X5	
STR NOT	X6	d: • Block "d" is programmed.
OR	X7	
AND STR		c•d • Blocks "c" and "d" are combined by AND STR.
OUT	Y226	
ORG	X10	
STR	X11	
OUT T/C	10	
	500	

[Explanation]

1. A circuit with two or more contact sets connected in series is called a serial circuit block. When connecting series circuit blocks in parallel, use the STR or STR NOT instruction to begin the branch, and the OR STR instruction to end the branch.
2. A circuit with two or more contact sets connected in parallel is called a parallel circuit block. When connecting parallel circuit blocks in series,use the STR or STR NOT instruction to begin the branch,and the AND STR instruction to end the branch.
3. The circuit shown below can be programmed according to either coding example (1) or (2).



Coding example (1)

```
ORG      X0
AND      X1
STR      X2
AND      X3
OR STR
STR      X4
AND      X5
OR STR
OUT      M400
```

○

Coding example (2)

```
ORG      X0
AND      X1
STR      X2
AND      X3
STR      X4
AND      X5
OR STR
OR STR
OUT      M400
```

△

2 levels of nesting

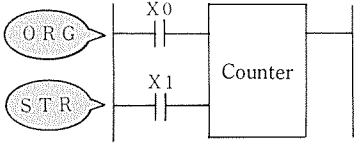
(1) Even when many parallel blocks are to be used, each circuit block is connectable to the previous one by specifying the OR STR instruction. The number of connections is not limited. (See coding example (1))

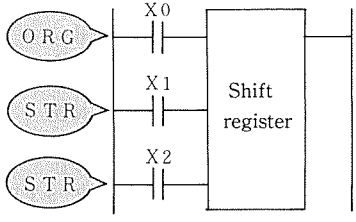
(2) The OR STR instruction can be used in the batch mode. In this case, however, the number of interations of the STR (STR NOT) instruction is limited to 7 times (up to 7 levels of nesting). (See coding example (2).)

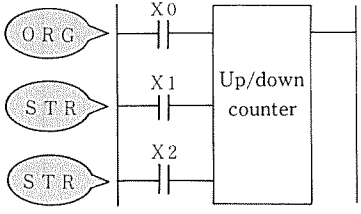
(3) The same rule applies to the AND STR instruction as well.

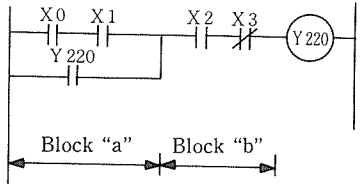
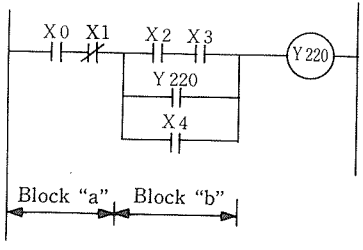
(4) If the STR or STR NOT is not used in correct combination with AND STR (or OR STR), it is detected as a syntax error.

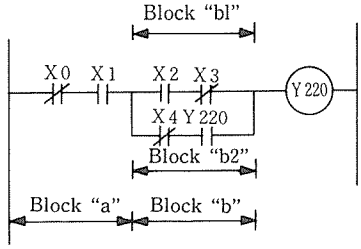
4. The STR or STR NOT instruction does not correspond to the AND STR (or OR STR) instruction if the counter, up/down counter, shift register or similar circuit has two or more input conditions.

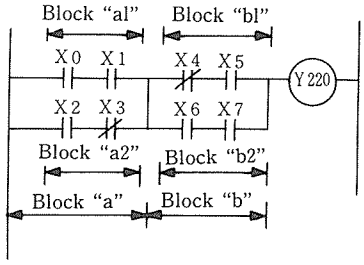


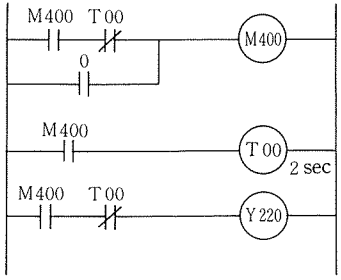
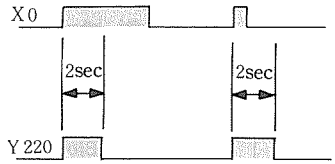
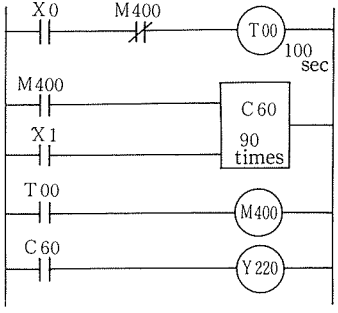
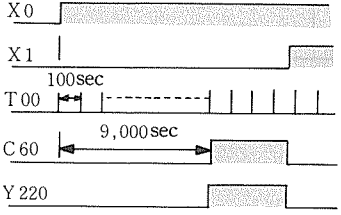


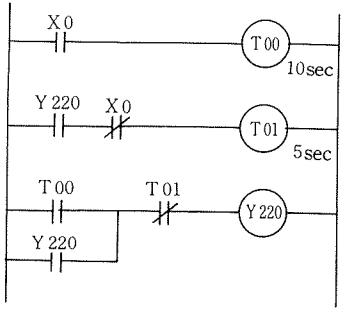
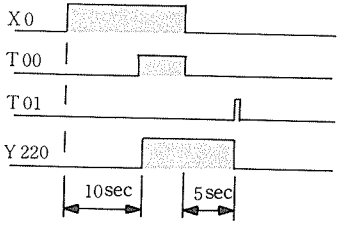
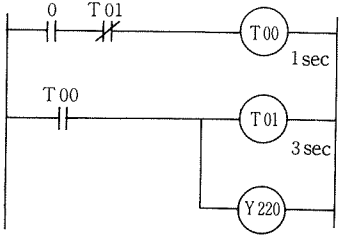
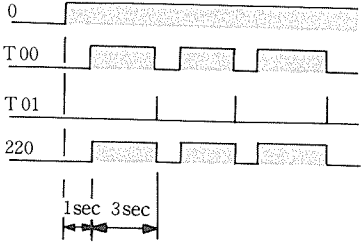


Circuit	Configuration	Program: Instruction code	Program: Data	Explanation
Parallel-to-serial circuit		ORG AND OR AND AND NOT OUT	X 0 X 1 Y 220 X 2 X 3 Y 220	a b • First the parallel circuit of block "a" and then the serial circuit of block "b" are programmed.
Serial-to-parallel circuit		ORG AND NOT STR AND OR OR AND STR OUT	X 0 X 1 X 2 X 3 Y 220 X 4 Y 220	a b • The circuit is divided into blocks "a" and "b" which are programmed separately. a · b • Blocks "a" and "b" are combined by AND STR.

Circuit	Configuration	Program		Explanation	
		Instruction code	Data		
Serial-to-parallel circuit		ORG NOT AND STR AND NOT STR NOT AND OR STR AND STR OUT	X 0 X 1 X 2 X 3 X 4 Y 220 Y 220	a b 1 b 2 b 1 + b 2 a · b	• Block "a" is programmed. • Block "bl" is programmed. • Block "b2" is programmed. • Blocks "bl" and "b2" are combined by OR STR. • Blocks "a" and "b" are combined by AND STR.

Circuit	Configuration	Program		Explanation	
		Instruction code	Data		
Serial connection of parallel circuits		ORG	X 0	a1	• First block "a1" and then block "a2" are programmed.
		AND	X 1		
		STR	X 2	a2	
		AND NOT	X 3		
		OR STR		a1+b2	• These blocks are combined. by OR STR.
		STR NOT	X 4	b1	• Blocks "bl" and "b2" are programmed in the same way as above.
		AND	X 5		
		STR	X 6	b2	
		AND	X 7		
		OR STR		b1+b2	
		AND STR		a・b	• Blocks "a" and "b" are combined by AND STR.
		OUT	Y220		

Circuit		Configuration	Program		Explanation
			Instruction code	Data	
Timer/counter application circuit	One-shot circuit		ORG AND NOT T/C OR OUT ORG OUT T/C ORG AND NOT T/C OUT	M400 00 X0 M400 M400 00.002 M400 00 Y220	
	Timer and counter circuits		ORG AND NOT OUT T/C ORG STR OUT T/C ORG T/C OUT ORG T/C OUT	X0 M400 00.100 M400 X1 60.090 00 M400 60 Y220	

Circuit		Configuration	Program		Explanation
			Instruction code	Data	
Timer/counter application circuit	ON/OFF delay circuit		ORG OUT T/C ORG AND NOT OUT T/C ORG T/C OR AND NOT T/C OUT	X0 00.010 Y220 X0 01.005 00 Y220 01 Y220	
	Flicker circuit		ORG AND NOT T/C OUT T/C ORG T/C OUT T/C OUT	X0 01 00.001 00 01.003 Y220	

1 PRINCIPLE OF PC

2 INPUT/OUTPUT AND NUMBERS

3 PROGRAMMING

Application Instructions (I) (1/2)

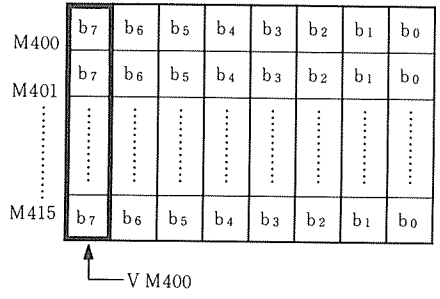
Classification	Instruction	Symbol	Name	Function	Component	No. of words	Change in register AR	ER	C	Acc	Reference page
Edge	FUN00	DIF	Rising edge	Detects rising edge (↑)of signal	M	1	●	●	●	●	73
	FUN01	DFN	Trailing edge	Detects trailing edge (↓)of signal.	M	1	●	●	●	●	73
Step process	FUN02	IF	If	Set/reset	None	1	●	●	●	●	75
	FUN03	IFR	If reset	Step process		1	●	●	●	●	77
Master control	FUN04	MCS	Master control	Sets common serial contacts.	None	1	●	●	●	●	84
	FUN05	MCR		Releases common serial contacts.		1	●	●	●	●	84
Jump	FUN06	JMP	Jump without addressing	Skip program up to corresponding JEND.	None	1	●	●	●	●	87
	FUN07	JEND				1	●	●	●	●	87
	FUN08	AJMP	Jump with addressing	Jumps to AJEND at corresponding address number.	Address No. (O to 63)	2	●	●	●	●	87
	FUN09	AJEND				2	●	●	●	●	87
Branch	FUN28	BRANCH	Branch	Stores Acc.	None	1	●	●	●	●	95
	FUN29	RETURN	Return	Returns stored Acc.	None	1	●	●	●	↕	95
Up/down counter	FUN40	UDC	Up/down counter	Up/down counter	VM (Note)	1	●	●	●	●	92
NOP	FUN41	NOP	No operation	Nothing occurs.	None	1	●	●	●	●	102
Latch	FUN45	LATCH	Latch	Resetting priority latch	M	1	●	●	●	●	97

Application Instructions (I) (2/2)

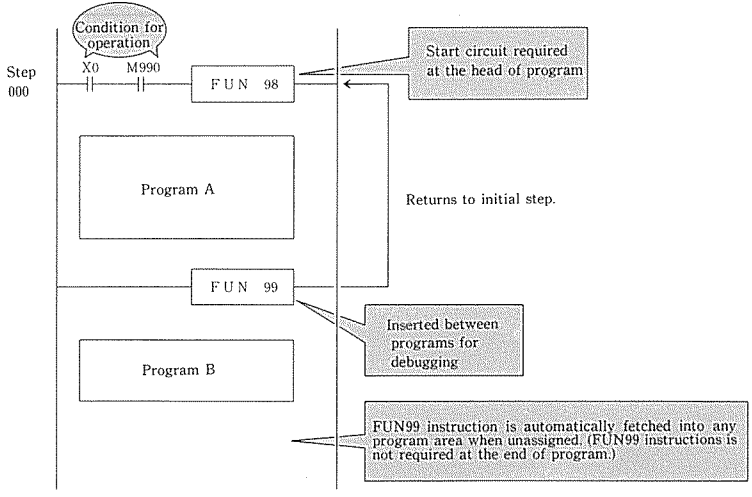
Classification	Instruction	Symbol	Name	Function	Component	No. of words	Change in register				Reference page
							AR	ER	C	Acc	
Shift register	FUN47	SFR	Shift register	16-bit shift register	VM (Note)	1	●	●	●	●	100
Set and reset	FUN88	SET	Set	Turns on component when Acc is at ON.	Y, M	1	●	●	●	●	75
	FUN89	RES	Reset	Turns off component when Acc is at ON.	Y, M	1	●	●	●	●	75
Start and end	FUN98	STA	Start	Operation start cntrol	None	1	●	●	●	●	70
	FUN99	END	End	Returns program to initial step.	None	1	—	—	—	—	70

●: Register remains unchanged
↕: Register changed
—: Register cleared

(Note) VM represents vertical 16 bits.
In the example below, VM is made up of 16 most significant bits of M400 throgh M415.



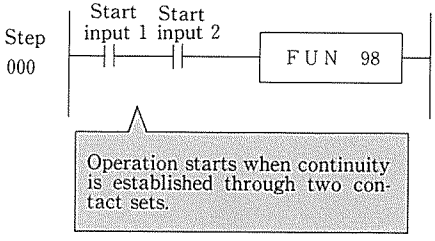
Instruction	Symbol	Name	Function	Component	No. of words	Change in register			
						AR	ER	C	Acc
FUN98	STA	Start	Operation start control	None	1	•	•	•	•
FUN99	END	End	Returns program to initial step.	None	1	—	—	—	—



[Explanation]

1. Start circuit

Operation start input is to be specified through a program. This means that start circuit must be written at the head of any program. (There is no restriction on number.)

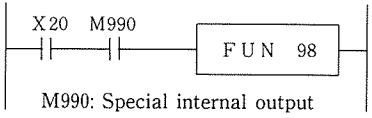


Condition for operation

Continuity is established across both start inputs 1 and 2, and program does not have an error.

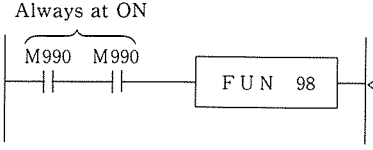
(⊣⊬⊢ ("b" contacts) specifiable)

Example of start circuit



M990: Special internal output always turned on

(a) Operation starts with external input X20 at ON and stops at OFF.



(b) Operation starts when turning on supply.

NOTE

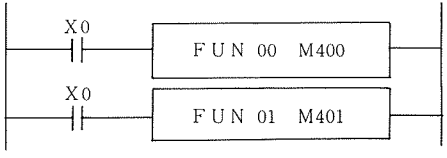
Under this configuration, operation starts as soon as the power supply is turned on. So avoid it for test run.
Such a configuration should be programmed when needed after completion of test run.

2. End

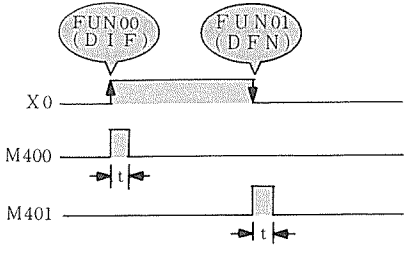
(1) The FUN99 instruction is not required usually. However, it is recommended to insert this instruction for separating programs at the time of test run since operation can be checked more easily. Program is executed from step 000 to FUN99 instruction.
Once operation has been confirmed, delete the FUN99 instruction.

(2) After completely clearing a program, all user memories are written with the FUN99 instruction (through indication is not provided).
Since the FUN99 instruction is assumed in an area not yet programmed, there is no need for writing that instruction at the end of a program.

Instruction	Symbol	Name	Function	Component	No. of words	Change in register			
						AR	ER	C	Acc
FUN00	DIF	Rising edge	Detects rising edge (_/‾) of signal.	M	1	•	•	•	•
FUN01	DFN	Trailing edge	Detects trailing edge (‾_) of signal.	M	1	•	•	•	•



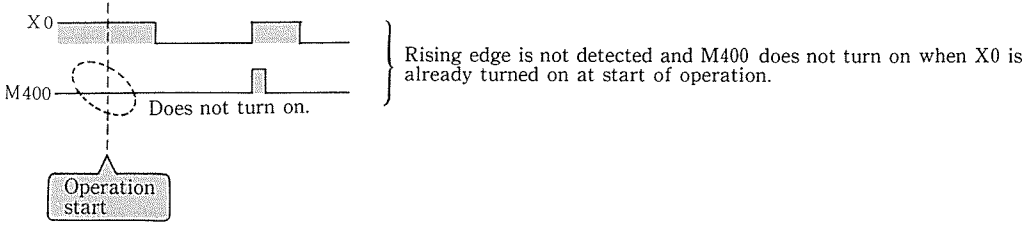
Code		Remarks
ORG	X0	
FUN00	M400	Detects rising edge.
ORG	X0	
FUN01	M401	Detects trailing edge.



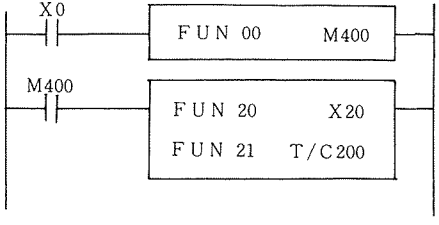
t: one (1) scan time

[**Explanation**]

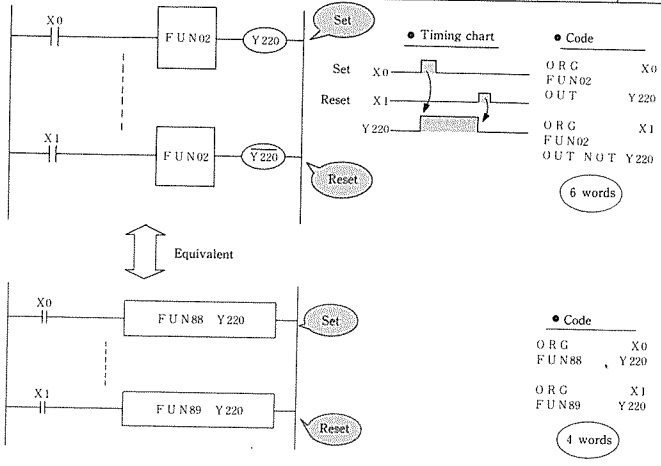
1. The FUN00 (DIF) instruction is used to detect the rising edge of an input signal (status change from LOW to HIGH), and the FUN01 (DFN) instruction is used to detect the trailing edge of the signal (status change from HIGH to LOW). These instructions are programmed in combination with an internal output (M) so that the specified internal output (M) turns on only for 1 scan time when the edge is detected. Any number of FUN00 and FUN01 instructions can be used (so far as internal output permits).
2. The edge detect instructions are executed according to the input change after operation start.



3. The edge detect function is effective for word LOAD, COMPARE and the like instructions, because they can be executed only when input condition changes.
(For instance, this function is used as a startup condition of arithmetic instructions.)



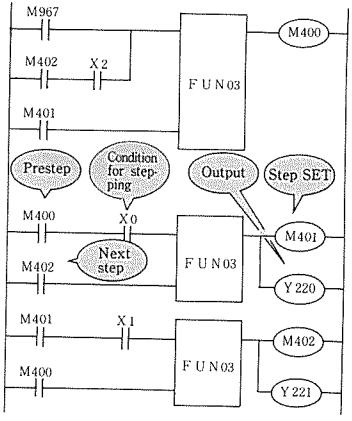
Instruction	Symbol	Name	Function	Component	No. of words	Change in register			
						AR	ER	C	Acc
FUN02	IF	If	Set/reset	None	1	•	•	•	•
FUN88	SET	Set	Turns on component when Acc is at ON.	Y, M	1	•	•	•	•
FUN89	RES	Reset	Turns off component when Acc is at ON.	Y, M	1	•	•	•	•



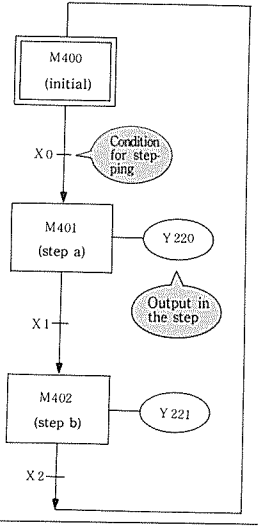
[**Explanation**]

1. Instructions FUN02 and OUT are combined and used as the SET instruction.
   Instructions FUN02 and OUT NOT are combined and used as the RESET instruction.
   - ○ ON status is held under SET input and OFF status is held under RESET input.
   - ○ Any other program may be inserted between SET coil and RESET coil. The program written last is given the highest priority.
   - ○ A keep relay can be composed when combining a FUN02 instruction with the memory-protected internal output.
2. FUN88 is the SET instruction. It provides the same function as a combination of the FUN02 and OUT instructions.
   FUN89 is the RESET instruction. It provides the same function as a combination of the FUN02 and OUT NOT instructions.
   Each of FUN88 and FUN89 instructions requires fewer words than the corresponding combination of instructions.
   If an output coil is programmed using both FUN88 and FUN89 instructions, a syntax error (double coil error E.) occurs, but operation is continuable.

Instruction	Symbol	Name	Function	Component	No. of words	Change in register			
						AR	ER	C	Acc
FUN03	IFR	If reset	Step process	None	1	•	•	•	•



• Status transition diagram



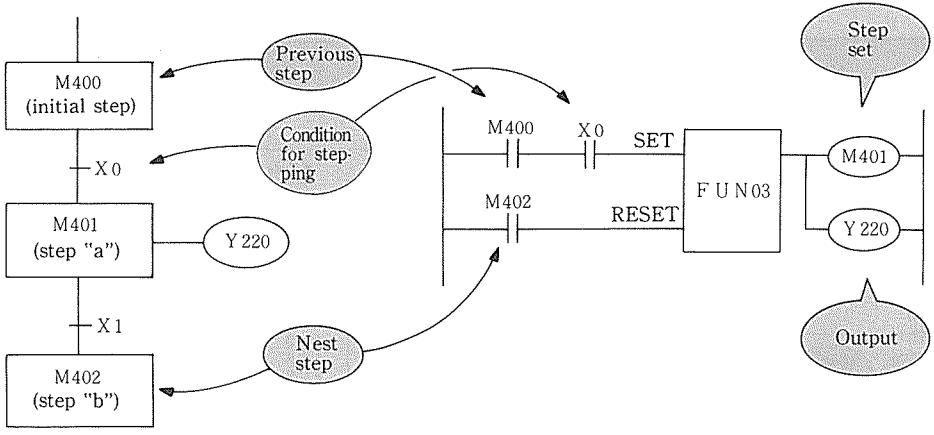
• Code

```
ORG     M967
STR     M402
AND       X2
OR STR
STR     M401
FUN 03
OUT     M400
ORG     M400
AND       X0
STR     M402
FUN 03
OUT     M401
OUT     Y220

ORG     M401
AND       X1
STR     M400
FUN 03
OUT     M402
OUT     Y221
```

[**Explanation**]

1. FUN03 is the step process (sequential control) instruction. Set input and reset input are provided. A step process program can be created in the regular format using the status transition diagram.



Status transition diagram

Explanation of operation

(1) If step condition X0 is set to ON in the initial step (M400), step "a" (M401) turns ON and Y220 is output.

(2) Y200 holds its output even when step condition X0 is set to OFF.

(3) When step condition X1 is set to ON, step "b" turns ON and Y220 is set to OFF.

(4) Even when step condition X1 is set to ON in the initial step (M400), step "b" (402) won't turn ON. All steps are executed in correct sequence.

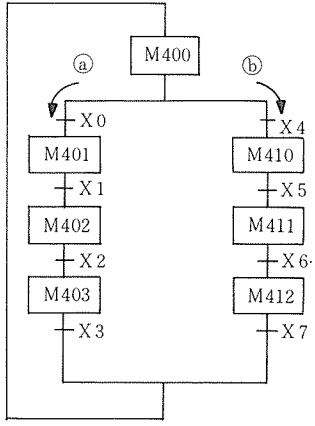
Progromming method

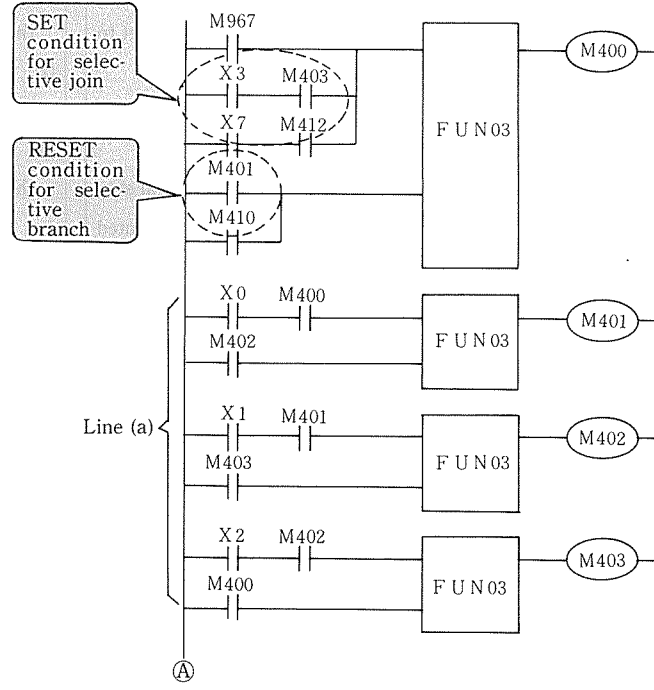
(1) To program a FUN03 SET input, the internal output (M400) specifying the previous step is ANDed with the condition for stepping (X0).

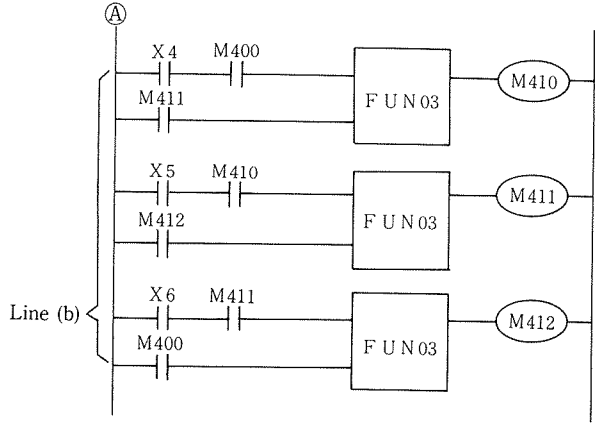
(2) For FUN03 RESET input, the internal output (M402) specifying the next step is programmed.

(3) After FUN03 the internal output (M401) specifying the current step and output (Y220) are programmed.

Selective branch and join







Explanation

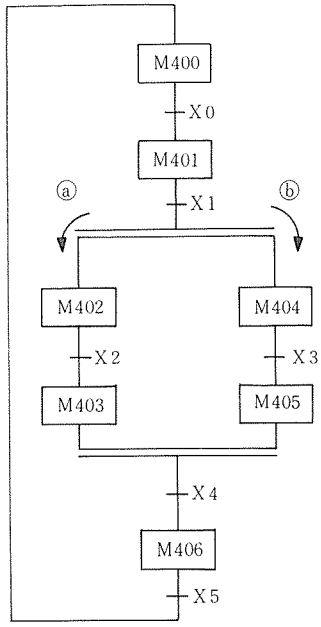
(1) Upon start If operation,step M400 (initial status) turns ON.

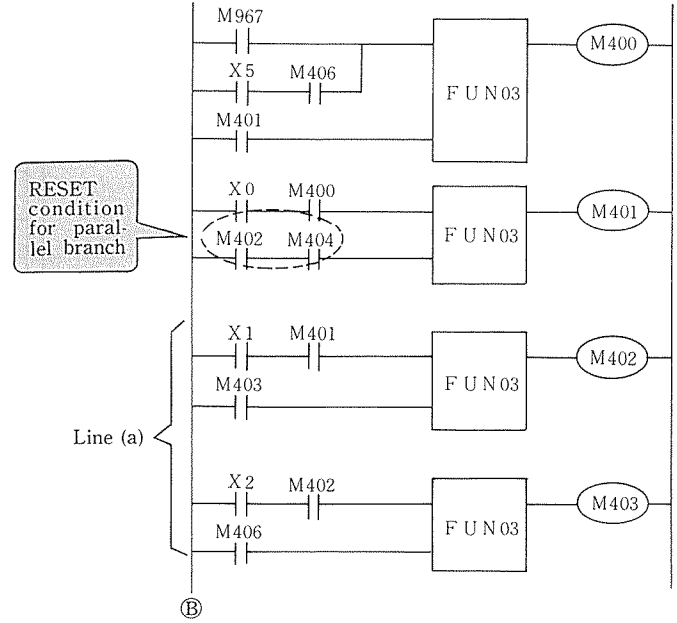
(2) When input X0 turns ON, line (a) is executed. When input X4 turns ON, line (b) is executed. Line (a) or (b) is selected according to which of inputs X0 and X4 turns ON earlir.

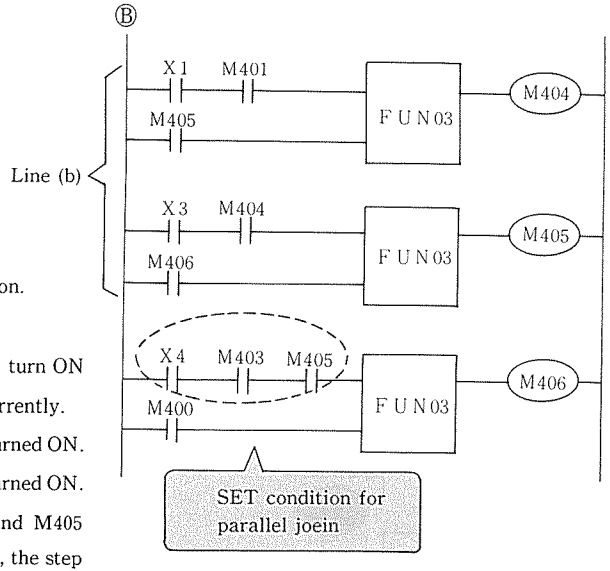
(3) In line (a), steps involving M401, M402 and M403 are executed in this order. When stepping condition input X3 turns ON, control returns to the step M400.

(4) In line (b), steps involving M410, M411 and M412 are executed in this sequence. When stepping condition input X7 turns ON, control returns to the step M400.

Parallel branch and join







Explanation

(1) Step M400 (intial status) is sit on start of operation.

(2) When input X0 is turned ON, step 401 turns on.

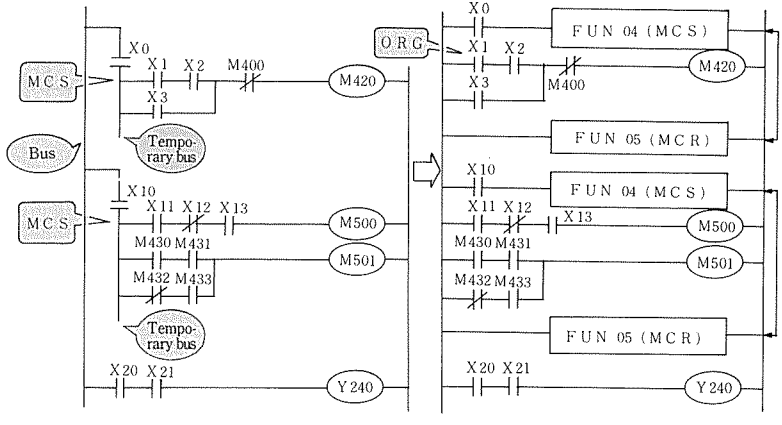
(3) When input X1 is turned ON, steps 402 and M404 turn ON simultaneously.Lines (a) and (b) are executed concurrently.

(4) In line (a), step M403 turns ON when input X2 is turned ON.

(5) In line (b), step M405 turns ON when input X3 is turned ON.

(6) When input X4 is turned ON with both M403 and M405 activated, the common step M406 turns on. However, the step M406 won't turn ON when M403 and M404 are activated in lines (a) and (b), respectively. Control of lines (a) and (b) returns to step M400 simultaneously when input X5 is set to ON.

Instruction	Symbol	Name	Function	Component	No. of words	Change in register			
						AR	ER	C	Acc
FUN04	MCS	Master control	Sets common serial contacts.	None	1	•	•	•	•
FUN05	MCR		Resets common serial contacts.		1	•	•	•	•



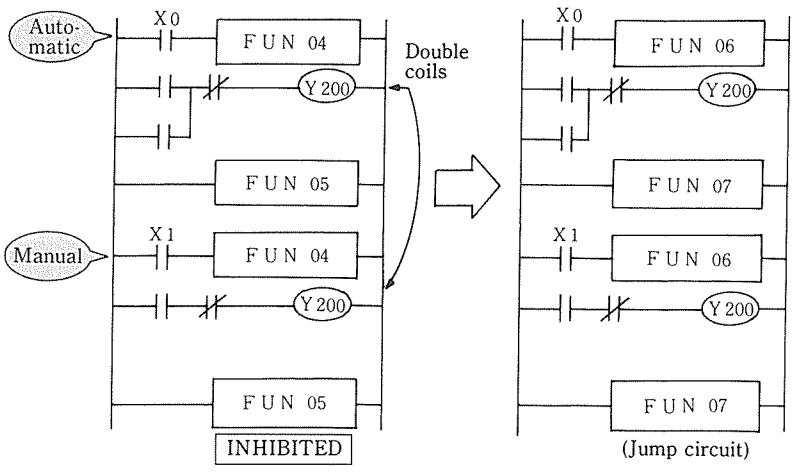
Code

```
ORG        X0
FUN04
ORG        X1
AND        X2
OR         X3
AND NOT    M400
OUT        M420
FUN05

ORG        X10
FUN04
ORG        X11
AND NOT    X12
AND        X13
OUT        M500
ORG        M430
AND        M431
STR NOT    M432
AND        M433
OR STR
OUT        M501
FUN05
ORG        X20
AND        X21
OUT        Y240
```

[Explanation]

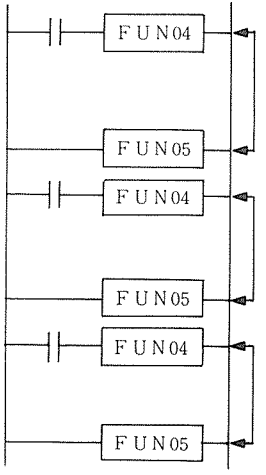
1. The FUN04 (MCS) and FUN05 (MCR) instructions are used for setting and resetting the common serial contacts, respectively. They must always be used as a pair. Otherwise, a syntax error occurs.
2. The FUN04 instructions must be followed by an ORG (or ORG NOT) instruction.
3. When the master control contacts are open, the subsequent output coil is set to OFF. In the example above, M420 is unconditionally OFF if input X0 is OFF.



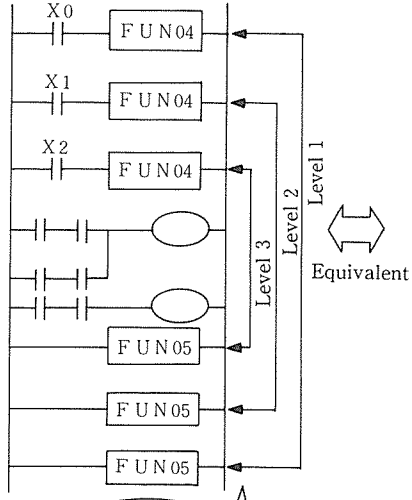
NOTE

For switchover between manual and automatic controls according to a master control instruction, take care not to use double coils. If double coils must be used, specify the FUN06 or FUN07 instruction. Refer page 65 to use FUN06 and FUN07.

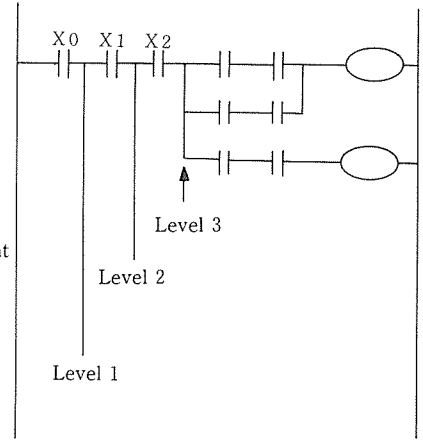
4. Any number of master control instructions can be used if they are paired unless nesting.



Any number of instructions usable if paired.

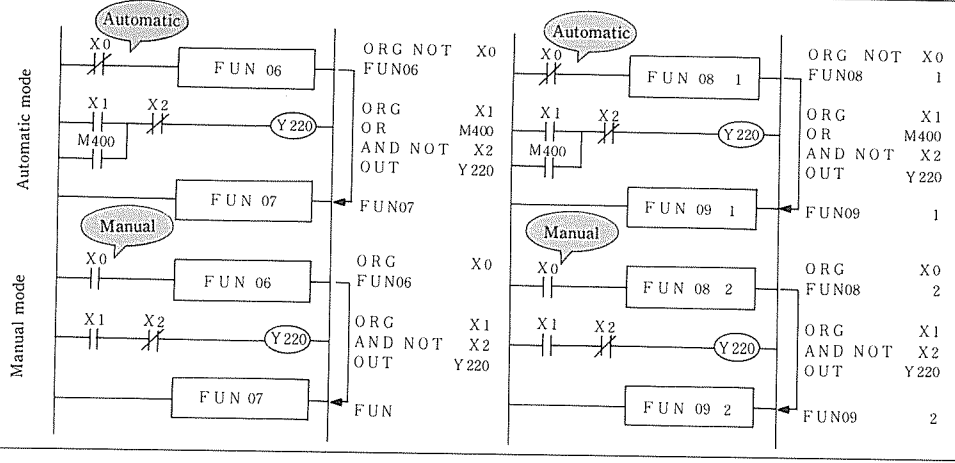


Nestig of instructions allowed up to 3 levels



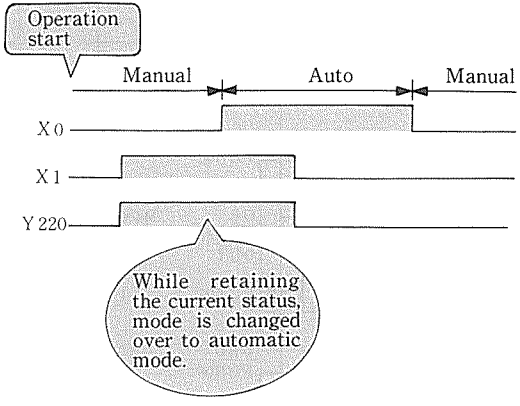
Instructions can be nested up to 3 levels. At four levels or more, syntax error will occur.

Instruction	Symbol	Name	Function	Component	No. of words	Change in register AR	ER	C	Acc
FUN06	JMP	Jump without addressing	Skips program till JEND.	None	1	•	•	•	•
FUN07	JEND				1	•	•	•	•
FUN08	AJMP	Jump with addressing	Jums to AJEND of corresponding address number.	Address No. (0 ~ 63)	2	•	•	•	•
FUN09	AJEND				2	•	•	•	•



[Explanation]

1. The FUN06 and FUN07 instructions specify jump without addressing, while the FUN08 and FUN 09 instructions specify jump with addressing. These instructions all cause cotrol to jump to JUMP END when the jump condition is set to ON.
2. When the jump conditions are satisfied, the program lines located between the current address and destination address are not executed. The output is held in the status before the jump. By using this function, a manual/auto switching circuit can be composed as illustrated above. If the same output coil is programmed between the jump circuits, a syntax error (double coil error E.) occurs, but operation can continue.



NOTE

If jump conditions are satisfied, the timer in the jump circuit stops operating. It restarts when the jump condetions are reset.

3. A jump instruction cannot be used between master control instructions.
4. The table below lists differences between the FUN06/07 instructions and FUN08/09 instructions. Scan time can be shortened by using the functional combination of FUN08/09.

Table: Differences between FUN06/07 and FUN08/09 (1/3)

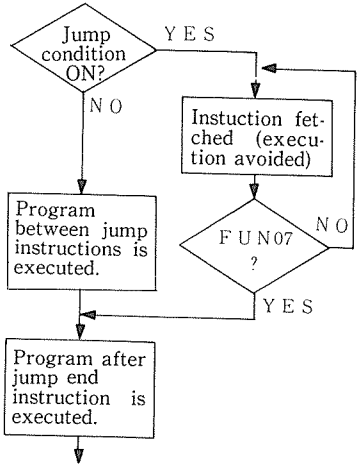
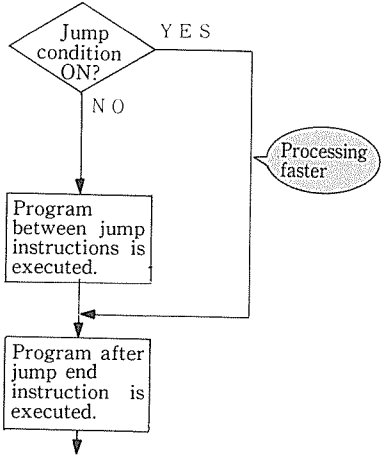
Item	FUN 06, FUN 07	FUN 08, FUN 09
Method of processing instruction		

Table: Differences between FUN06/07 and FUN08/09 (2/3)

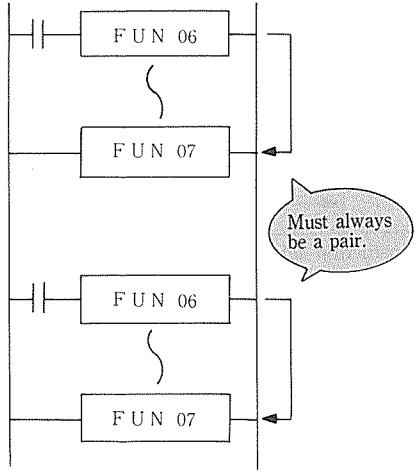
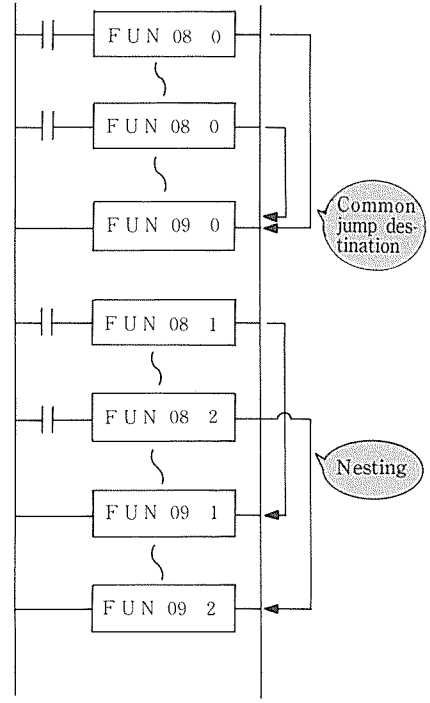
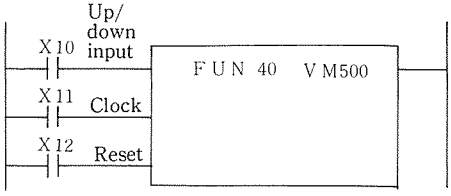
Item	FUN 06, FUN 07	FUN 08, FUN 09
Jump method		

Table: Differences between FUN06/07 and FUN08/09 (3/3)

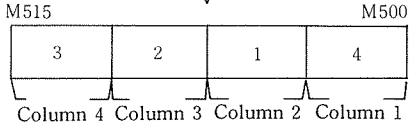
Item	FUN 06, FUN 07	FUN 08, FUN 09
Jump method	1. These instructions must always be used as a pair. If not paired, a syntax error will occur. 2. Nesting is unallowable.	1. Jump from multiple FUN08 instur-cions to a single FUN09 instruction is allowed. 2. Nesting is allowed at different ad-dresses. 3. Jump to a preceding step is also posible.

Instruction	Symbol	Name	Function	Component	No. of words	Change in register			
						AR	ER	C	Acc
FUN40	UDC	Up/down counter	Up/down counter	VM	1	•	•	•	•

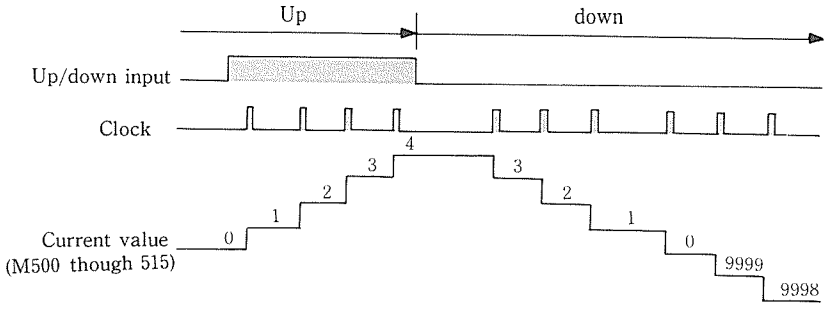


Code		Remarks
ORG	X10	
STR	X11	Use STR.
STR	X12	
FUN 40	VM500	UDC

16bits from M500 to M515are used for current value register.

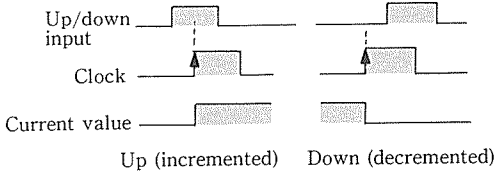


···· Indicates the current value of 3214 times (4-digit BCD value).

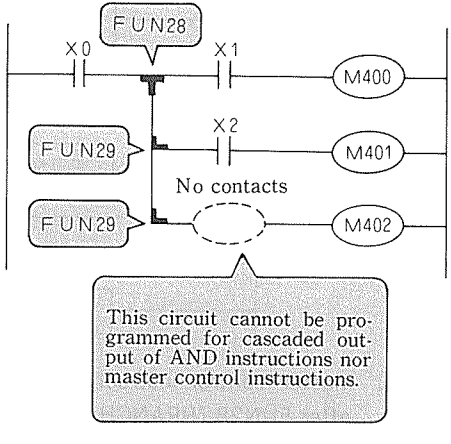

Up
down
Up/down input
Clock
Current value
(M500 though 515)
0
1
2
3
4
3
2
1
0
9999
9998

[**Explanation**]

1. FUN40 (UDC) is the up/down counter instruction. It is to be programmed in combination with an internal output (VM).
2. 16 bits staring from the coil number specified by that instruction (M500 through M515 in the example shown above) are used as the current value register of up/down counter. The current value is presented in BCD 4 digits.
3. The up/down input, clock input and reset input are programmed in that order.
   The current value changes at the rising edge of the clock (from OFF to ON). Either UP or DOWN condition is selected according to the ON or OFF status of up/down input as shown below.



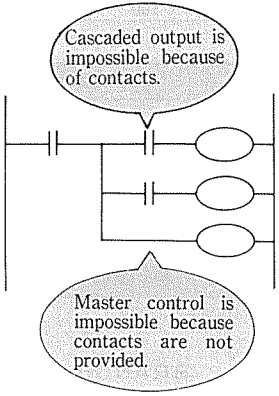
Instruction	Symbol	Name	Function	Component	No. of words	Change in register			
						AR	ER	C	Acc
FUN28	BRANCH	Branch	Stores Acc.	None	1	•	•	•	•
FUN29	RETURN	Return	Rturns stored Acc data .	None	1	•	•	•	↕



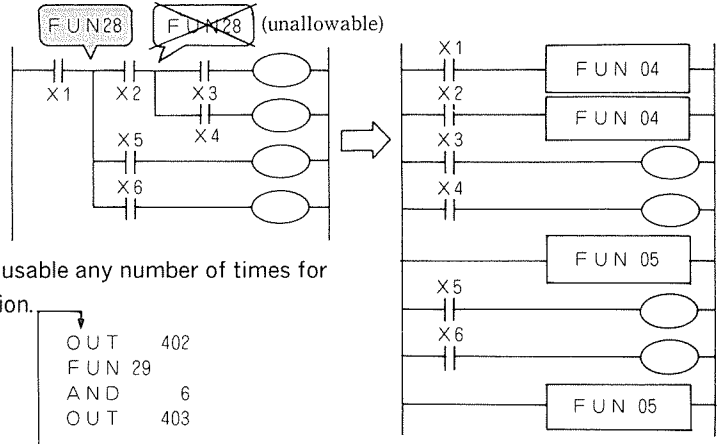
Code		Remarks
ORG	X0	
FUN 28		BRANCH
AND	X1	
OUT	M400	
FUN 29		RETURN
AND	X2	
OUT	M401	
FUN 29		RETURN
OUT	M402	

[**Explanation**]

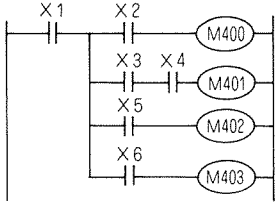
1. FUN28 (BRANCH) and FUN29 (RETURN) instructions allow programming of a circuit which is incompatible with the cascaded output of AND instruction and the master control instruction.



2. The FUN28 (BRANCH) instruction cannot be used more than once in the same circuit.
   For such a circuit below, master control instruction must be used.



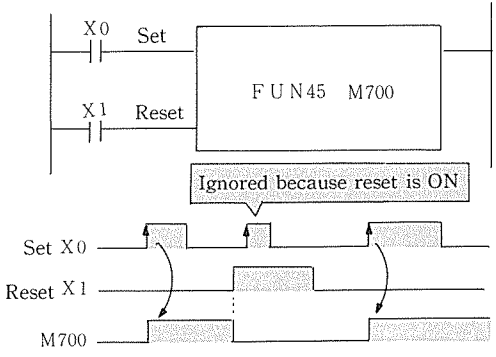
3. The FUN29 (RETURN) instruction is usable any number of times for a single FUN28 (BRANCH) instruction.




```
ORG      1
FUN 28
AND      2
OUT    400
FUN 29
AND      3
AND      4
OUT    401
FUN 29
AND      5
OUT    402
FUN 29
AND      6
OUT    403
```

4. Master control instructions is unusable after FUN28 (BRANCH) instruction.

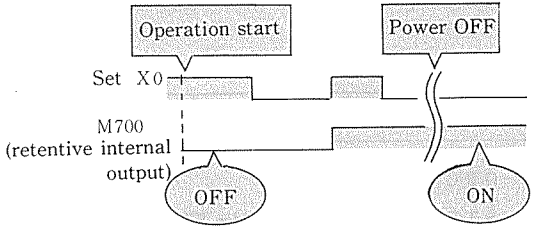
Instruction	Symbol	Name	Function	Component	No. of words	Change in register			
						AR	ER	C	Acc
FUN45	LATCH	Latch	Reset priority latch	M	1	•	•	•	•



Code		Remarks
ORG	X0	
STR	X1	Use STR.
FUN 45	M700	

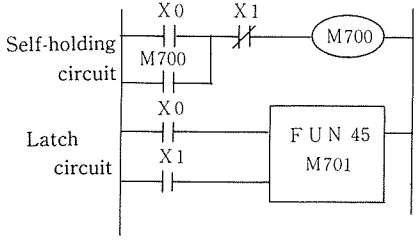
[Explanation]

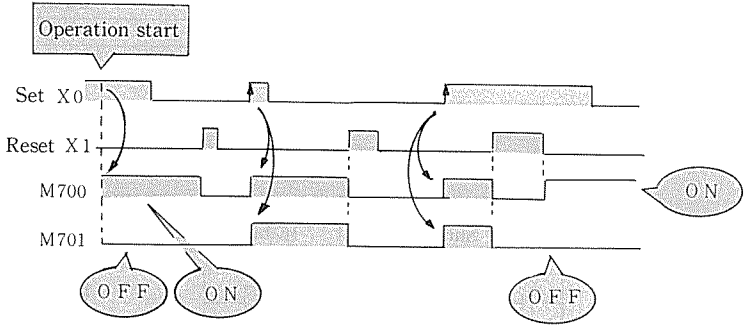
1. FUN45 (LATCH) is an edge triggered latch instruction with the reset priority signal. It must be programmed in combination with an internal output (M).
2. The ON status is set at the rising edge of the set input signal (from OFF to ON). The OFF status is set when the reset input goes ON. When the reset input is ON, the set input is rejected. If the set input and reset input go ON simultaneously, the reset input has a priority.
3. The FUN45 instruction can be combined with a retentive memory internal output (M) to produce the function of a keep relay.



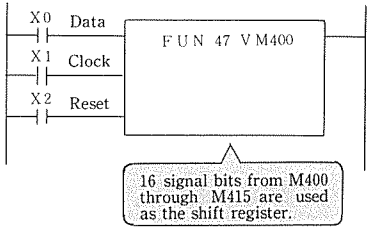
In the above sequence, the status of M700 at occurrence of power interruption is retained till its recovery because M700 is a retentive internal output.
Even when the set input X0 is turned ON at start of operation, edge will not be detected and M700 will not turn ON.

4. The self-holding circuit operates at a specific level (ON or OFF status), but the lath is operated at the signal edge. This causes the difference shown below.

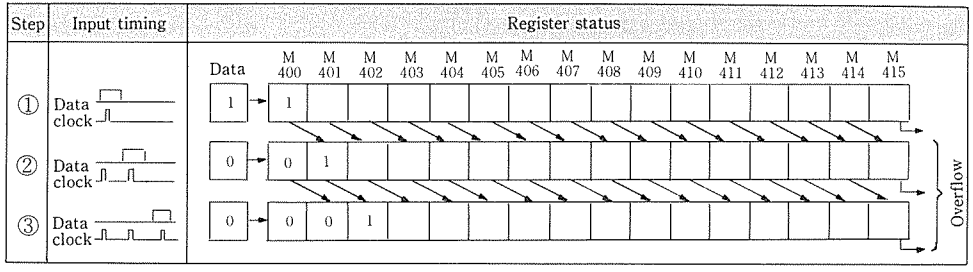




Instruction	Symbol	Name	Function	Component	No. of words	Change in register			
						AR	ER	C	Acc
FUN47	SFR	Shift register	16-bit shift register	VM	1	•	•	•	•

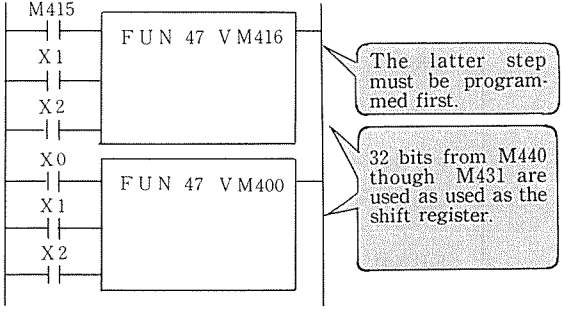


Code		Remarks
ORG	X0	Use STR.
STR	X1	
STR	X2	
FUN 47	VM400	

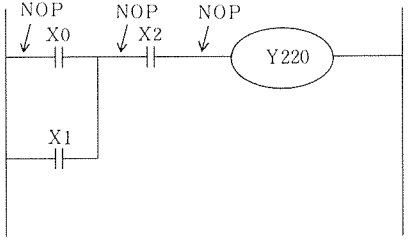


[**Explanation**]

1. FUN47 (SFR) is the shift register instruction. It is to be programmed in combination with internal output (VM).
2. 16 bits (M400 through M415 in the example shown above), from the coil number specified by the FUN47 instruction and higher, are used as the register.
3. When the clock input rises (from OFF to ON), concurrent data input (ON/OFF status) is written in the least significant bit of the register (M400 in this example). The ON/OFF status of each register is shifted to the next high-order bit synchronized with the rise of clock input.
4. Data of most significant bit (M415 in this example) may overflow as a result of shift operation. When connecting two (2) or more shift registers, the latter step (with a larger I/O number) must be programmed first in order to prevent data being lost due to overflow.



Instruction	Symbol	Name	Function	Component	No. of words	Change in register			
						AR	ER	C	Acc
FUN41	NOP	NOP	No operation	None	1	•	•	•	•



Code		Remarks
FUN41		NOP
ORG	X0	
OR	X1	
FUN41		NOP
AND	X2	
FUN41		NOP
OUT	Y220	

[**Explanation**]

1. FUN41 is NOP instruction. This instruction does not cause any execution in its step. It may be located anywhere in a program.

1 PRINCIPLE OF PC

2 INPUT/OUTPUT AND NUMBERS

3 PROGRAMMING

Arithmetic Instructions (1/4)

Classification	Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register AR	ER	CR	Acc	Reference page
Load	FUN 0	LOADI	Load	Constant→AR	Constant (0000H~9999H)	2	↕	•	•	•	112
	FUN10	LOADW		I/O→AR	WX,WY,WM,T/C100~295	2	↕	•	•	•	112
	FUN20	LOADB		I/O→AR	VX,VY,VM,T/C0~95	2	↕	•	•	•	112
	FUN50	LBYTI		1 byte constant → AR_L (lower 8 bits)	Constant (00~FF)	2	↕	•	•	•	112
	FUN60	BLOAD		I/O→AR_L(lower 8 bits)	WX,WY,WM,T/C100~295	2	↕	•	•	•	112
Out	FUN21	OUTW	Out	AR→I/O	WY,WM,T/C100~295	2	•	•	•	•	116
	FUN22	OUTB		AR→I/O	VY,VM	2	•	•	•	•	116
	FUN71	BOUT		AR_L→I/O	WY,WM,T/C100~295	2	•	•	•	•	116
Add	FUN 1	ADDI	BCD add	AR B+constant→AR	Constant (0000H~9999H)	2	↕	•	↕	•	122
	FUN11	ADD		AR B+I/O→AR	WX,WY,WM,T/C100~295	2	↕	•	↕	•	122
	FUN51	ABYTI	BIN add	AR+constant→AR	Constant (0~FFFF)	2	↕	•	↕	•	122
	FUN61	ADBNR		AR+I/O→AR	WX,WY,WM,T/C100~295	2	↕	•	↕	•	122
Subtract	FUN 2	SUBI	BCD subtract	AR B−constant→AR	Constant (0000H~9999H)	2	↕	•	↕	•	124
	FUN12	SUB		AR B−I/O→AR	WX,WY,WM,T/C100~295	2	↕	•	↕	•	124
	FUN52	SBYTI	BIN subtract	AR−constant→AR	Constant (0~FFFF)	2	↕	•	↕	•	124
	FUN62	SUBNR		AR−I/O→AR	WX,WY,WM,T/C100~295	2	↕	•	↕	•	124
Multiply	FUN 3	MULI	BCD multiply	AR B∗constant→AR	Constant (0000H~9999H)	2	↕	•	↕	•	126
	FUN13	MUL		AR B∗I/O→AR	WX,WY,WM,T/C100~295	2	↕	•	↕	•	126
	FUN53	MBYTI	BIN multiply	AR∗constant→AR	Constant (0~FFFF)	2	↕	↕	↕	•	126
	FUN63	MUBNR		AR∗I/O→AR	WX,WY,WM,T/C100~295	2	↕	↕	↕	•	126

Arithmetic Instructions (2/4)

Classification	Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register AR	ER	CR	Acc	Reference page
Divide	FUN 4	DIVI	BCD divide	AR B/constant→AR	Constant (0000H～9999H)	2	↕	•	↕	•	128
	FUN14	DIV		AR B/ I/O→AR	WX,WY,WM,T/C100～295	2	↕	•	↕	•	128
	FUN54	DBYTI	BIN divide	AR/constant→AR	Constant (0～FFFF)	2	↕	↕	↕	•	128
	FUN64	DIBNR		AR/I/O→AR	WX,WY,WM,T/C100～295	2	↕	↕	↕	•	128
Logic	FUN 5	ANDI	AND	AR AND constant→AR	Constant (0000H～9999H)	2	↕	•	•	•	130
	FUN15	AND		AR AND I/O→AR	WX,WY,WM,T/C100～295	2	↕	•	•	•	130
	FUN55	BANDI		AR_L AND constant→AR_L	Constant (00～FF)	2	↕	•	•	•	130
	FUN 6	ORI	OR	AR OR constant→AR	Constant (0000H～9999H)	2	↕	•	•	•	130
	FUN16	OR		AR EOR I/O→AR	WX,WY,WM,T/C100～295	2	↕	•	•	•	130
	FUN56	BORI		AR_L OR constant→AR_L	Constant (00～FF)	2	↕	•	•	•	130
	FUN66	EXOR	Exclusive-OR	AR EOR I/O→AR	WX,WY,WM,T/C1100～295	2	↕	•	•	•	130
	FUN85	WNOT	Logical not	$\overline{AR}$→AR	None	1	↕	•	•	•	130
Compare	FUN 7	CPEHI	Compare(≧)	AR ≧ constant……1→C	Constant (0000H～9999H)	2	•	•	↕	•	133
	FUN17	CPEH		AR≧I/O……1→C	WX,WY,WM,T/C100～295	2	•	•	↕	•	133
	FUN57	BCPHI		AR_L≧constant……1→C	Constant (00～FF)	2	•	•	↕	•	133
	FUN 8	CPEI	Compare(=)	AR=constant……1→C	Constant (0000H～9999H)	2	•	•	↕	•	133
	FUN18	CPE		AR=I/O……1→C	WX,WY,WM,T/C100～295	2	•	•	↕	•	133
	FUN58	BCPEI		AR_L=constant……1→C	Constant (00～FF)	2	•	•	↕	•	133
	FUN 9	CPLI	Compare(<)	AR<constant……1→C	Constant (0000H～9999H)	2	•	•	↕	•	133
	FUN19	CPL		AR<I/O……1→C	WX,WY,WM,T/C100～295	2	•	•	↕	•	133
	FUN59	BCPLI		AR_L<constant……1→C	Constant (00～FF)	2	•	•	↕	•	133

Arithmetic Instructions (3/4)

Classification	Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register				Reference page
							AR	ER	CR	Acc	
Carry	FUN23	OUC	Out carry	C→I/O	Y,M	1	•	•	•	•	136
	FUN83	CLC	clear carry	C←"0"	None	1	•	•	0	•	136
	FUN84	SEC	Set carry	C←"1"	None	1	•	•	1	•	136
Convert	FUN24	BCD	BCD convert	BCD convert	None	1	↕	•	↕	•	137
	FUN25	BNR	BIN convert	BIN convert	None	1	↕	•	↕	•	137
	FUN74	SEG	7-segment convert	Decodes AR_{LL} data into 7-segment display.	None	1	↕	•	•	•	139
	FUN75	ASC	ASCII convert	Converts AR_{LL} data into ASCII code.	None	1	↕	•	•	•	139
	FUN78	ENCOD	Encode	16 to 4	None	1	↕	•	↕	•	141
	FUN79	DECOD	Decode	4 to 16	None	1	↕	•	↕	•	141
Shift	FUN26	LSFR	Shift	Left shift	None	1	↕	•	↕	•	143
	FUN27	RSFR		Right shift	None	1	↕	•	↕	•	143
	FUN76	ROL	Rotate	CW rotate	None	1	↕	•	↕	•	143
	FUN77	ROR		CCW rotation	None	1	↕	•	↕	•	143
Mask	FUN72	MASKL	Left mask	Masks AR by specified bits from left.	0~255	2	↕	•	•	•	145
	FUN73	MASKR	Right mask	Masks AR by specified bits from right.	0~255	2	↕	•	•	•	145
Exchange	FUN80	SWAP	AR_H/AR_L exchange	$AR_H \rightleftarrows AR_L$	None	1	↕	•	•	•	147
	FUN81	BSWAP	AR_{LH}/AR_{LL} exchange	$AR_{LH} \rightleftarrows AR_{LL}$	None	1	↕	•	•	•	147
	FUN82	XCG	AR/ER exchange	AR⇄ER	None	1	↕	↕	•	•	147

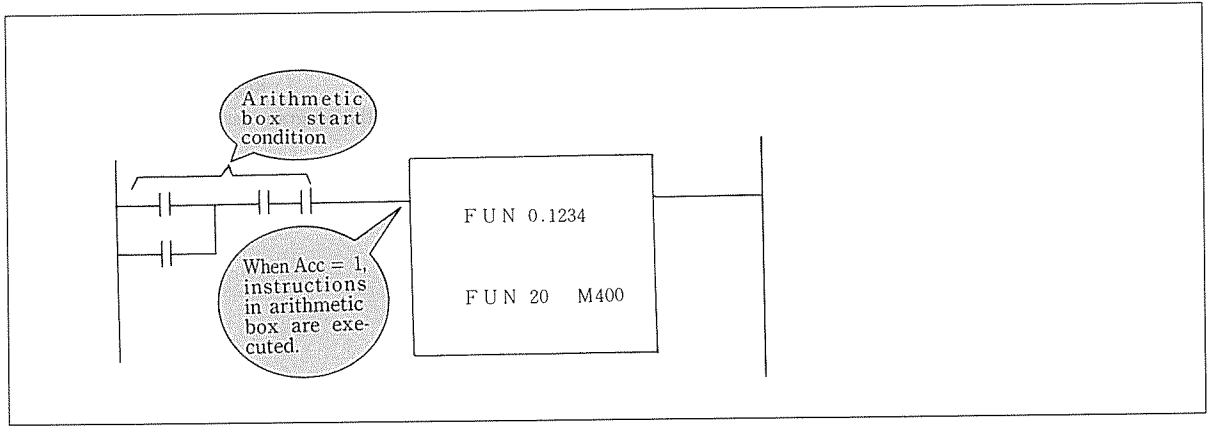
Arithmetic Instructions (4/4)

Classification	Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register				Reference page
							AR	ER	CR	Acc	
Distribute /extract	FUN48	EX	Extract	Fetches data into AR from I/O address-specified by ER.	None	1	↕	●	↕	●	149
	FUN49	DB	Distribute	Outputs data from AR to I/O address-specified by ER.	None	1	●	●	↕	●	149

●: Register remains unchanged
↕: Register changed

Note: Although the table above contains entries of constants (00~FF) and (0~FFFF), the programmer is not provided with A to F keys which are indispensable for specification of hexadecimal constants.
Therefore, specification must be made in decimal constants.
Besides, each constant can be entered in up to 3 digits.

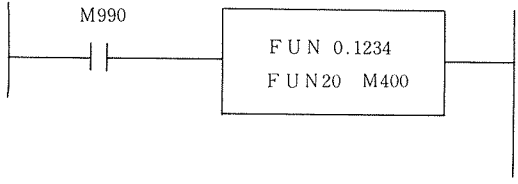
Example: FUN51 427 AR + 1ABH → AR
(Decimal 427 = hexadecimal 1ABH)



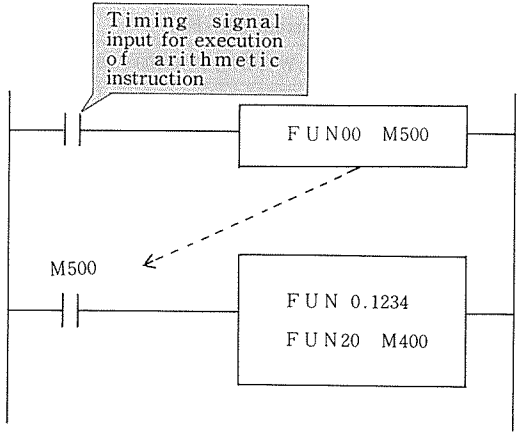
[Explanation]

1. Arithmetic instruction is assumed to be contained in the arithmetic box,and consecutive arithmetic instructions are put in the same arithmetic box. Before each arithmetic box, start condition is to be provided. When the start condition is satisfied (Acc=1), arithmetic instruction in the arithmetic box is excuted. This won't occur if the start condition is not satisfied (Acc=0), and the previous status is retained.

2. For an arithmetic instruction to be executed every scan, it is recommended to use the special internal output M990 for the start condition since ON status is always secured.

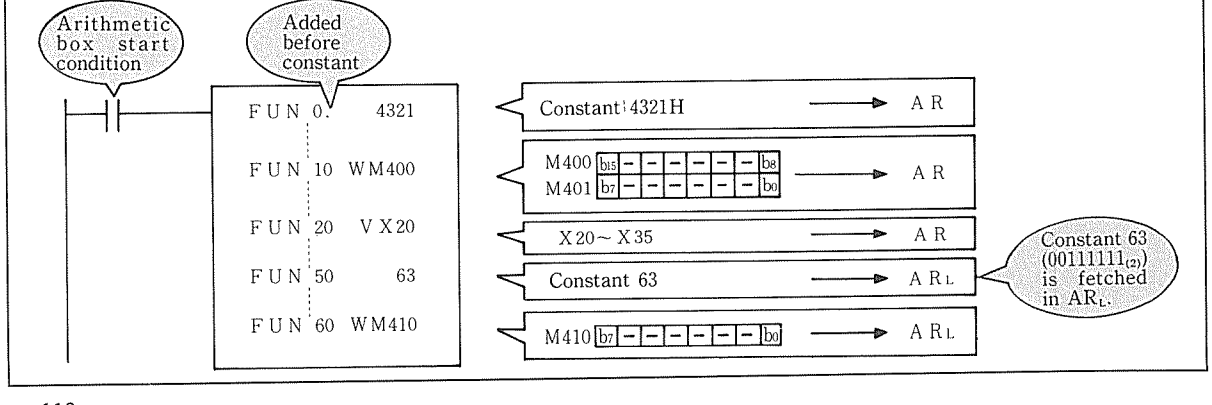


3. For an arithmetic instruction to be executed for only one scan at a certain timing, it is recommended to use the edge instruction as the start condition.



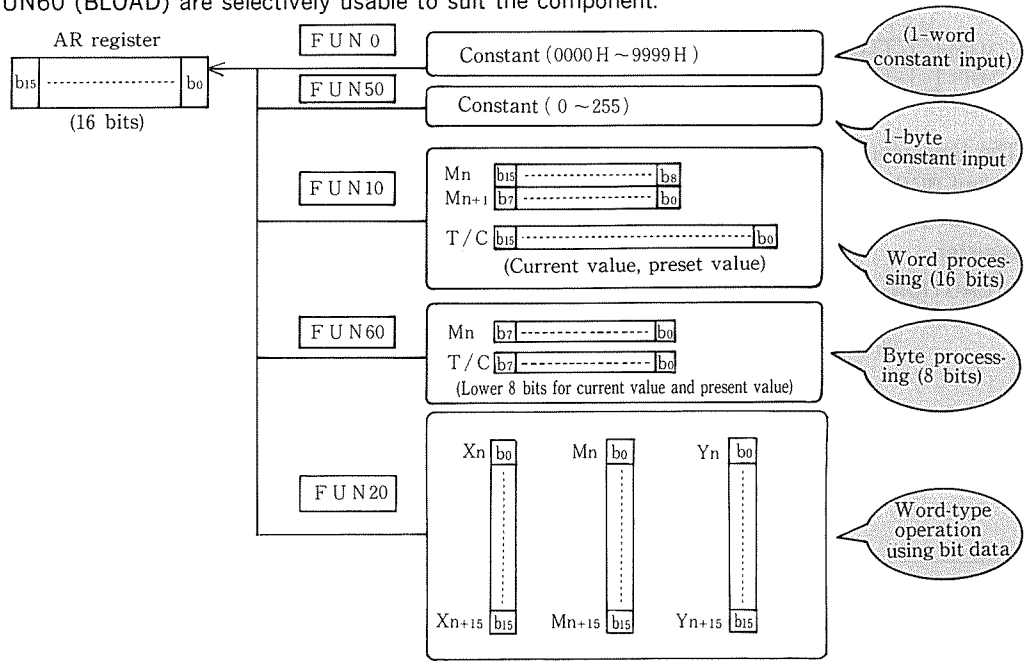
Concept of arithmetic instruction	Load	out	4-rule calculations				Logic	Compare	Carry	Convert	Shift	Mask	Exchange	Distribute/ extract
			Add	Subtra-ct	Multiply	Divide								
110	112	116	122	124	126	128	130	133	136	137	143	145	147	149

Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	Acc
FUN 0	LOADI	Load	Constant → AR	Constant (0000H ~ 9999H)	2	↕	•	•	•
FUN10	LOADW		I/O → AR	WX, WY, WM, T/C100 ~ 295	2	↕	•	•	•
FUN20	LOADB		I/O → AR	VX, VY, VM, T/C0 ~ 95	2	↕	•	•	•
FUN50	LBYTI		1 byte constant → AR_L (lower 8 bits)	Constant (00 ~ FF)	2	↕	•	•	•
FUN60	BLOAD		I/O → AR_L (lower 8 bits)	WX, WY, WM, T/C100 ~ 295	2	↕	•	•	•



[Explanation]

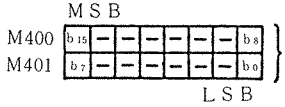
1. Load instruction loads the word data (16 bits) or byte data (8 bits) to be processed into the AR register. Five kinds of load instructions FUN0 (LOAD1), FUN10 (LOADW), FUN20 (LOADB), FUN50 (LBYT1) and FUN60 (BLOAD) are selectively usable to suit the component.



(1) The FUN0 (LOADI) instruction loads a one-word constant (0000H to 9999H) into the AR register. The constant must be preceded by a decimal point (.) when keying in.

(2) The FUN10 (LOADW) instruction loads one-word I/O data into the AR register.

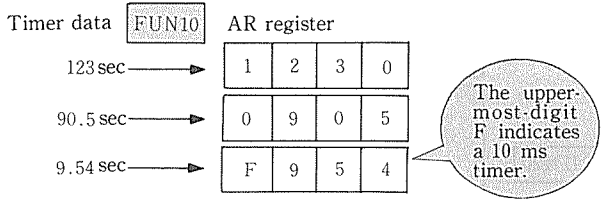
① Internal outputs are used for both bit and byte data (8-bit data for each number). 8 bit data of the specified internal output (Mn) and the next internal output (Mn+1), 16-bit data in total, are loaded into the AR register.



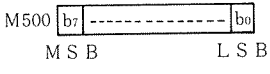
One 16-bit word consisting of data from M400 and M401 is used.

② The timer/counter current values (T/C100 to T/C195) and preset values (T/C200 to T/C295) are 4-digit BCD (16bits) data. The counter preset value and current value are loaded into the AR register without change. However, the timer value is processed as shown below before loaded into the AR register.

CAUTION

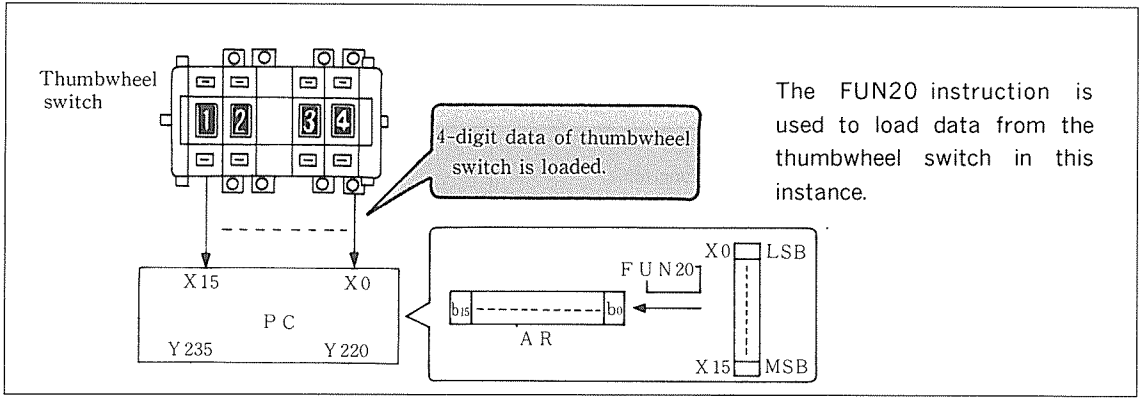


(3) The FUN60 (BLOAD) instruction loads 1-byte (8-bit) I/O data into the lower 8 bits (AR_L) of AR register. The upper 8 bits (AR_H) of AR register remain unchanged. The FUN60 instruction is used to load the external input of 8-bit analog module.



8-bit data of the specified number is processed.

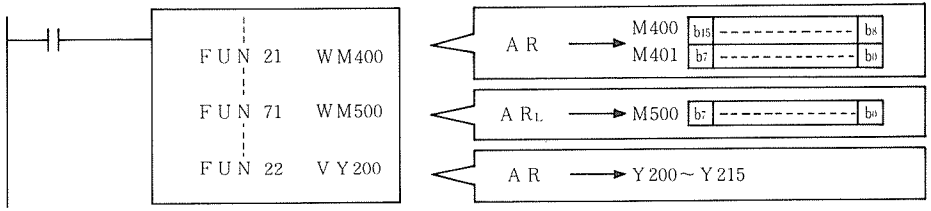
(4) The FUN20 (LOADB) instruction loads 16 I/O data simultaneously into the AR register. 16 data from the specified number and upward are loaded into the AR register.



(5) The FUN50 (LBYT1) instruction loads a described bit pattern into the AR register.
The E-series programmers (PGMJ and PGM-R2) do not have keys A through F required for hexadecimal notation. However, when a decimal constant (0 to 255) is specified by the FUN50 instruction,it is handled as a one-byte data (00H to FFH) and loaded into the lower 8 bits (AR_L) of the AR register. In this case, the upper 8 bits (AR_H) of the same register remain unchanged. When used in combination with the FUN80 instruction, the FUN50 instruction is capable of loading a desired bit pattern into the upper 8 bits (AR_H).

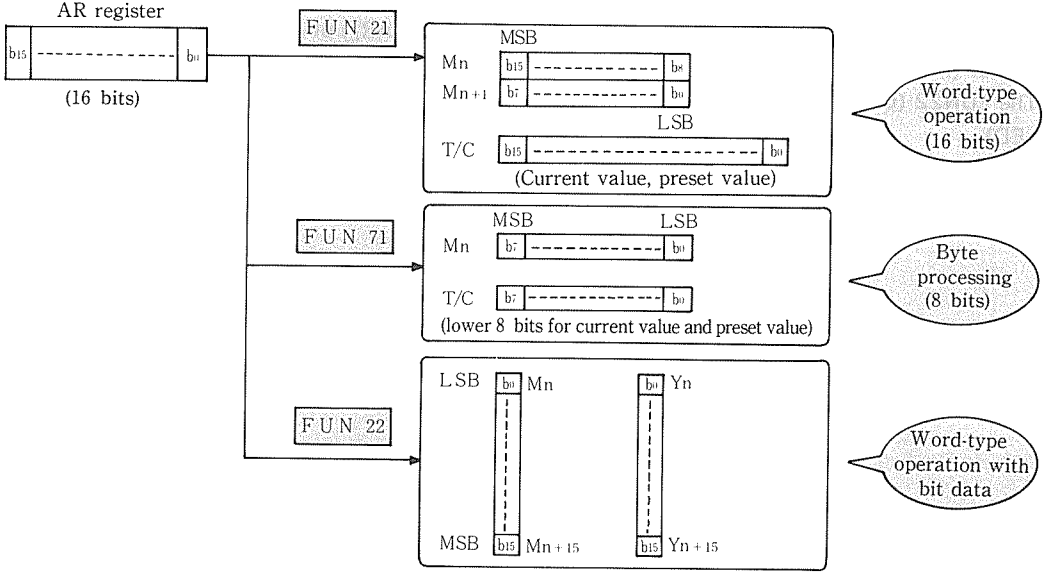
Concept of arithmetic instruction	Load	out	4-rule calculations				Logic	Compare	Carry	Convert	Shift	Mask	Exchange	Distribute/ extract
			Add	Subtra-ct	Multiply	Divide								
110	112	116	122	124	126	128	130	133	136	137	143	145	147	149

Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	Acc
FUN21	OUTW	Out	AR→I/O	WY, WM, T/C100~295	2	•	•	•	•
FUN71	BOUT		AR_L→I/O(8 bits)	WY, WM, T/C100~295	2	•	•	•	•
FUN22	OUTB		AR→I/O	VY, VM	2	•	•	•	•



[Explanation]

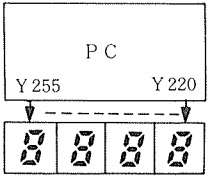
1. OUT instruction outputs data in the AR register to the destination component. Three kinds of OUT instructions below are selectively usable so as to meet the component.



(1) The FUN21 (OUTW) instruction outputs data in the AR register to the 16-bit area made up of the specified internal output (Mn) and the next internal output (Mn+1). This instruction is also used for outputting AR register data to current value (T/C100 through T/C195) or preset value (T/C200 to T/C295) of timer/counter.

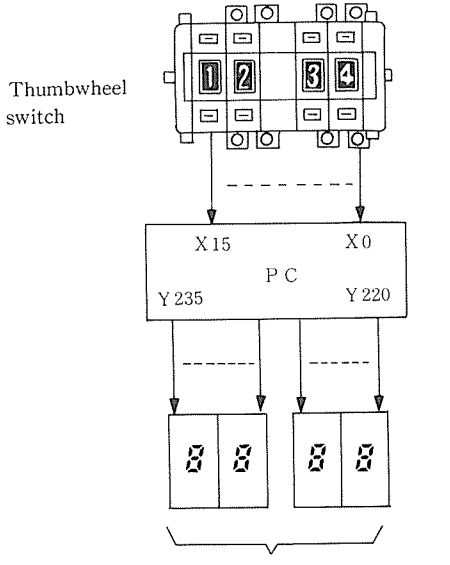
(2) The FUN71 (BOUT) instruction outputs the lower 8-bit data (AR_L) of AR register to the specified internal output (Mn). This instruction is used for analog output when the analog I/O module is mounted.

(3) The FUN22 (OUTB) instruction is used to output AR register data to the numerical display (7-segment LED).



[Application example of LOAD and OUT instructions]

1. Explanation of operations



(example)

Truth table of thumbwheel switch

Switch terminal		8	4	2	1
PC terminal		X 3	X 2	X 1	X 0
		X 7	X 6	X 5	X 4
		X11	X10	X 9	X 8
Digit		X15	X14	X13	X12
Thum bwheel switch dial	0				
	1				●
	2			●	
	3			●	●
	4		●		
	5		●		●
	6		●	●	
	7		●	●	●
	8	●			
	9	●			●

Indicates terminal wiring.

● ON

Truth table of display unit with decoder circuit

Digit \ Display terminal		D	C	B	A
	PC terminal	Y 223	Y 222	Y 221	Y 220
		Y 227	Y 226	Y 225	Y 224
		Y 231	Y 230	Y 229	Y 228
		Y 235	Y 234	Y 233	Y 232
Numerical display unit	0				
	1				●
	2			●	
	3			●	●
	4		●		
	5		●		●
	6		●	●	
	7		●	●	●
	8	●			
	9	●			●

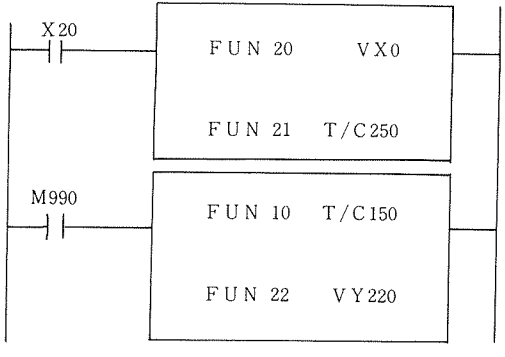
Indicates terminal wiring.

● ON

(1) Preset value of the counter in PC is set when X20 turns ON with a 4-digit BCD thumbwheel switch connected to the PC external input terminal.

(2) Current value of the counter in PC is output to the 7-segmnet display unit. This unit is provided with a decoder circuit.

2. Sequence



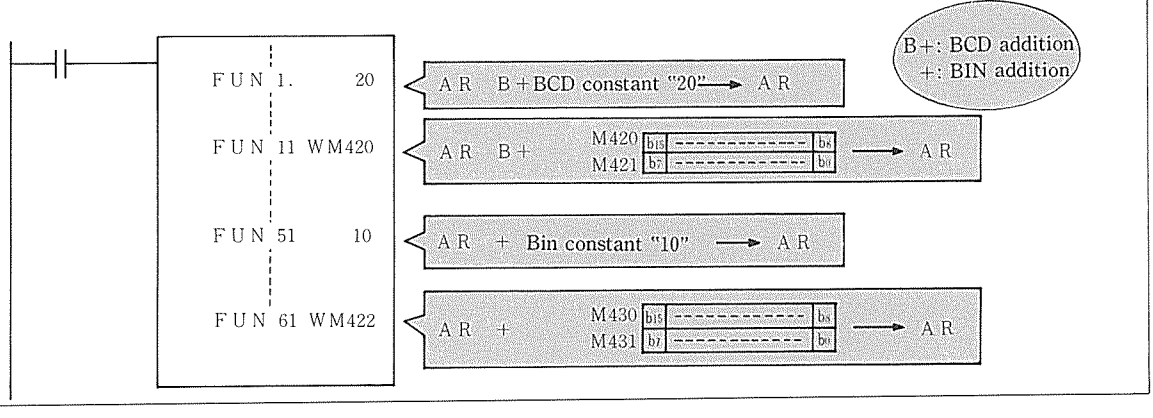
X0～X15→AR

AR→preset value of T/C50 (T/C 250)

Current value of T/C50 (T/C150) →AR

AR→Y220～Y235

Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	Acc
FUN1	ADDI	BCD add	AR B+ constant→AR	Constant (0000H~9999H)	2	↕	•	↕	•
FUN11	ADD	BCD add	AR B+ I/O→AR	WX, WY, WM, T/C100~295	2	↕	•	↕	•
FUN51	ABYTI	BIN add	AR + constant→AR	Constant (0~FFFF)	2	↕	•	↕	•
FUN61	ADBNR	BIN add	AR + I/O→AR	WX, WY, WM, T/C100~295	2	↕	•	↕	•



[**Explanation**]

1. ADD instructions add AR register data to component data and load the sum to the AR register. There are two kinds of ADD instructions; BCD and BIN ADD instructions, each of which consists of paired instructions for selective use depending on whether the compontent data is a constant or I/O.
2. When the sum is more than four (4) digits, the carry C turns OFF. In this case, instruction is handled as shown below.

Conditions	Instruction	AR	C	Remarks
Sum has exceeded 4 digits.	FUN 1	Remain unchanged	1	Carry C indicates occurrence of error.
	FUN 11		1	
	FUN 51	Sum of 4 digits or less loaded	1	Carry C indicates occurrence of a carry.
	FUN 61		1	

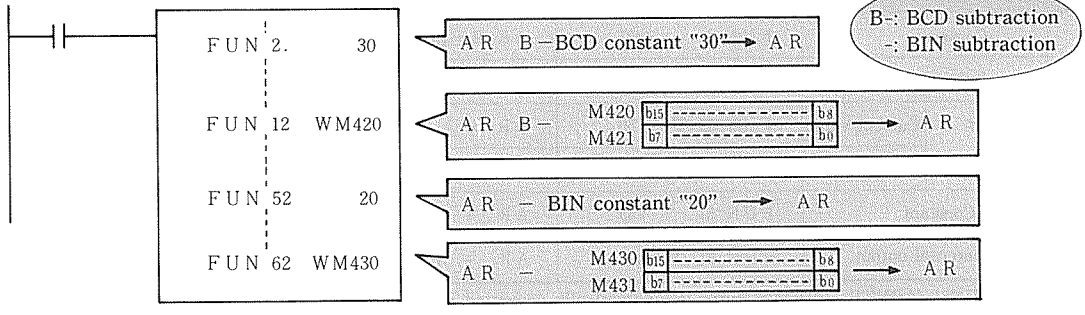
3. If a non-BCD data is handled by the FUN1 or FUN11 instruction, neither AR register nor carry C data is assured. The table below lists example programs for different components.

Classification	Component		Program	Explanation
BCD addition	Constant		FUN 1. 4321	AR B+ BCD constant 4321 →AR
	Internal output		FUN 11 WM500	AR B+ WM 500 →AR
	Timer counter	Current value	FUN 11 T/C150	AR B+ T/C50 current value →AR
		Preset value	FUN 11 T/C250	AR B+ T/C50 preset value →AR
Bin addition	Constant		FUN 51 735	AR + Bin constant 735 →AR
	Internal output		FUN 61 WM422	AR + WM 422 →AR

Note: In case of FUN51 (ABYTI), a decimal entry is automatically converted into a hexadecimal value before addition because of the restriction peculiar to the programmer. In addition, entry is possible only up to 3 digits. For instance, entry $(735)_{10}$ is converted into $(2DF)_{16}$.

Concept of arithmetic instruction	Load	out	4-rule calculations				Logic	Compare	Carry	Convert	Shift	Mask	Exchange	Distribute/ extract
			Add	Subtract	Multiply	Divide								
110	112	116	122	124	126	128	130	133	136	137	143	145	147	149

Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	Acc
FUN2	SUBI	BCD subtract	AR B− constant →AR	Constant (0000H~9999H)	2	↕	•	↕	•
FUN12	SUB		AR B− I/O→AR	WX, WY, WM, T/C100~295	2	↕	•	↕	•
FUN52	SBYTI	Bin subtract	AR − constant →AR	Constant (0~FFFF)	2	↕	•	↕	•
FUN62	SUBNR		AR − I/O→AR	WX, WY, WM, T/C100~295	2	↕	•	↕	•



[Explanation]

1. SUB instructions subtract component data from AR register data and load the difference to the the AR register.
   There are two kinds of SUB instructions; BCD and BIN SUB instructions, each of which consints of paired instructions for constant and I/O, respectively.
2. When subtraction results in 0 or a positive value, the carry C turns OFF. If the difference of subtraction is negative, each instruction is handled as listed below.

Condition	Instruction	AR	C	Remarks
Difference is negative.	FUN 2	Remain unchanged	1	Carry C indicates occurrence of error.
	FUN 12		1	
	FUN 52	Difference loaded (expressed in two's complement)	1	Carry C indicates decrement to next lower digit.
	FUN 62		1	

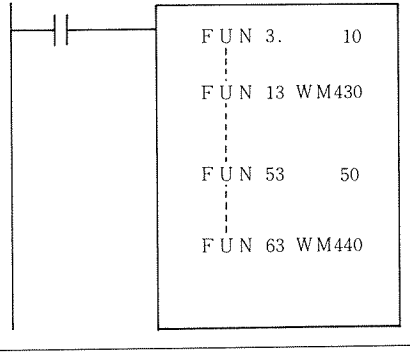
3. If a non-BCD data is handlded by the FUN2 or FUN12 instruction, neither AR register data nor carry C data is reliable. The table below lists example programs for different components.

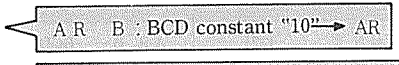
Classification	Component		Program	Explanation
BCD subtraction	Constant		FUN 2. 4321	AR B− BCD constant 4321 →AR
	Internal output		FUN 12 WM500	AR B− WM 500 →AR
	Timer/counter	Current value	FUN 12 T/C150	AR B− T/C 50 current value→AR
		Preset value	FUN 12 T/C250	AR B− T/C 50 preset value →AR
Bin subtraction	Constant		FUN 52 735	AR −BIN constant 735→AR
	Internal output		FUN 62 WM510	AR − WM510 →AR

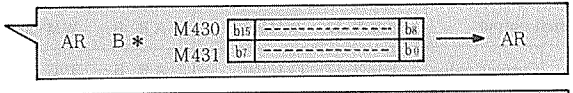
Note: In case of FUN52 (SBYTI), a decimal entry is automatically converted into a hexadecimal value before subtraction because of the restriction peculiar to the programmer. In addition, entry is possible only up to 3 digits. For instance, entry $(735)_{10}$ is converted into $(2DF)_{16}$.

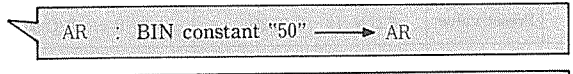
Concept of arithmetic instruction	Load	out	4-rule calculations				Logic	Compare	Carry	Convert	Shift	Mask	Exchange	Distribute/ extract
			Add	Subtract	Multiply	Divide								
110	112	116	122	124	126	128	130	133	136	137	143	145	147	149

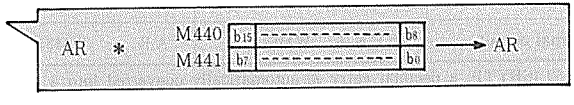
Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	Acc
FUN3	MULI	BCD multiply	AR B* constant →AR	Constant (0000H~9999H)	2	↕	•	↕	•
FUN13	MUL		AR B* I/O→AR	WX, WY, WM, T/C100~295	2	↕	•	↕	•
FUN53	MBYTI	BIN multiply	AR * constant →AR	Constant (0~FFFF)	2	↕	↕	↕	•
FUN63	MUBNR		AR * I/O→AR	WX, WY, WM, T/C100~295	2	↕	↕	↕	•











B* BCD multiplication
* BIN multiplication

[Explanation]

1. MUL instructions multiply AR register data with component data and load the product to the AR register. There are two kinds of MUL instructions; BCD and BIN MUL instructions, each of which consists of paired instructions for constant and I/O, respectively.
2. When multiplication results in 4 digits or less, the carry C turns OFF. If the product of multiplication exceeds 4 digits, each instruction is handled as listed below.

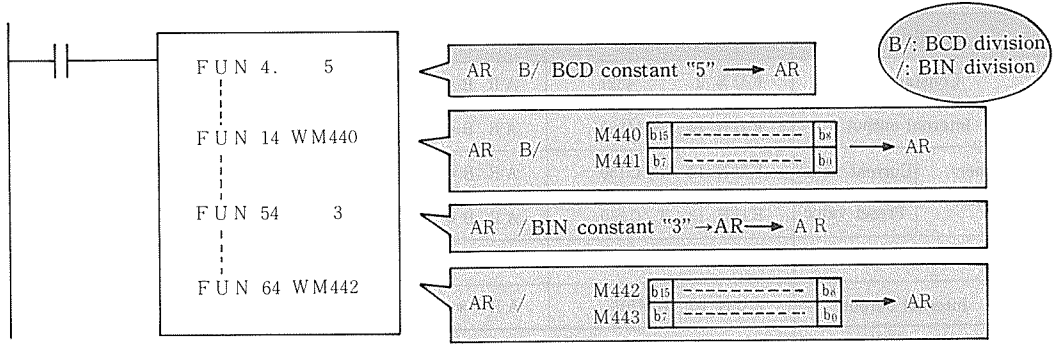
Condition	Instruction	AR	ER	C	Remarks
Product exceeds 4 digits.	FUN 3	Remain unchanged	Remain unchanged	1	Carry C indicates occurrence of error.
	FUN 13			1	
	FUN 53	4th digit and lower of product	5th digit and upper of product	1	Carry C indicates the product reaches 5 digits.
	FUN 63			1	

3. If a non-BCD data is handled by the FUN3 or FUN13 instruction, neither AR register data nor carry C data is reliable. The table below lists example programs for different components.

Classification	Component		Program	Explanation
Bin multiplication	Constant		FUN 3. 4321	AR B* BCD constant 4321 →AR
	Internal output		FUN 13 WM500	AR B* WM500 →AR
	Timer/counter	Current value	FUN 13 T/C150	AR B* T/C50 current value →AR
		Preset value	FUN 13 T/C250	AR B* T/C50 preset value →AR
BCD multiplication	Constant		FUN 53 735	AR * BIN constant 735 →AR
	Internal output		FUN 63 WM510	AR * WM510 →AR

Note: In case of FUN52 (SBYTI), a decimal entry is automatically converted into a hexadecimal value before multiplication because of the restriction peculiar to the programmer. In addition, entry is possible only up to 3 digits. For instance, entry $(735)_{10}$ is converted into $(2DF)_{16}$.

Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	Acc
FUN4	DIVI	BCD divide	AR B/ constant →AR	Constant (0000H～9999H)	2	↕	•	↕	•
FUN14	DIV		AB B/ I/O→AR	WX, WY, WM, T/C100～295	2	↕	•	↕	•
FUN54	DBYTI	BIN divide	AR / constant →AR	constant (0～FFFF)	2	↕	↕	↕	•
FUN64	DIBNR		AR / I/O→AR	WX, WY, WM, T/C100～295	2	↕	↕	↕	•



[Explanation]

1. DIV instructions devide AR register with component data and load the quotient to the AR register. There are two kinds of DIV instructions; BCD and BIN divide instructions, each of which consists of paired instructions for constant and I/O, respectively.
2. Each DIV instruction is handled as listed below in cases of usual division and 0 division.

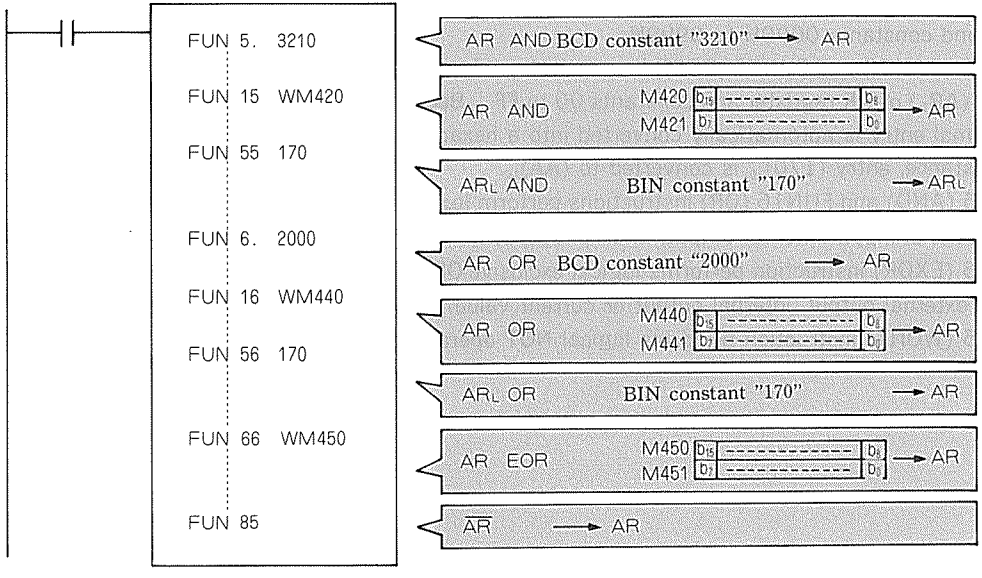
Condition	Instruction	AR	ER	C	Remarks
Usual division	FUN 4	Quotient	Remain unchanged.	0	Remainder is neglected.
	FUN 14			0	
	FUN 54	Quotient	Carry C indicates occurrence of error.	0	Remainder is loaded in ER.
	FUN 64			0	
÷ 0	FUN 4	Remain unchanged	Remain unchanged.	1	Carry C indicates occurrence of error.
	FUN 14			1	
	FUN 54			1	
	FUN 64			1	

3. If non-BCD constant is handled in in the FUN4 or FUN14 instruction, neither AR register data nor carry C data is reliable.
   The table below lists example programs for different components.

Classification	Component		Program	Explanation
BCD division	Constant		FUN 4. 5	AR B/ BCD constant 5 → AR
	Internal output		FUN 14 WM500	AR B/ WM 500 → AR
	Timer/ counter	Current value	FUN 14 T/C150	AR B/ T/C50 current value → AR
		Preset value	FUN 14 T/C250	AR B/ T/C50 preset value → AR
BIN division	Constant		FUN 53 12	AR / BIN constant 12 → AR
	Internal output		FUN 64 WM510	AR / WM510 → AR

Note: In case of FUN54 (DBYT1), a decimal entry is automatically converted into a hexadecimal value before division because of the restriction peculiar to the programmer. Besides, entry is possible only up to 3 digits. For instance, entry $(12)_{10}$ is converted into $(C)_{16}$.

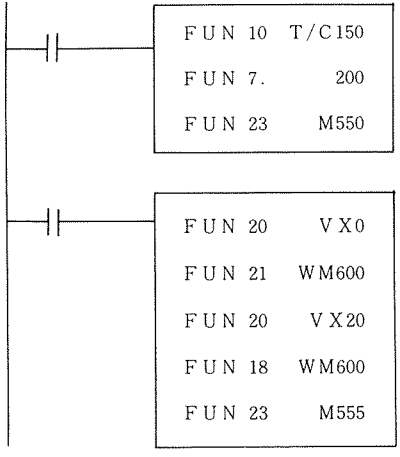
Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	Acc
FUN 5	ANDI	AND	AR AND constant → AR	constant (0000H ~ 9999H)	2	↕	•	•	•
FUN 15	AND		AR AND I/O → AR	WX, WY, WM. T/C 100 ~ 295	2	↕	•	•	•
FUN 55	BANDI		AR_L AND constant → AR_L	constant (00 ~ FF)	2	↕	•	•	•
FUN 6	ORI	OR	AR OR constant → AR	constant (0000H ~ 9999H)	2	↕	•	•	•
FUN 16	OR		AR OR I/O → AR	WX, WY, WM, T/C 100 ~ 295	2	↕	•	•	•
FUN 56	BORI		AR_L OR constant → AR_L	constant (00 ~ FF)	2	↕	•	•	•
FUN 66	EXOR	Exclusive-OR	AR EOR I/O → AR	WX, WY, WM, T/C 100 ~ 295	2	↕	•	•	•
FUN 85	WNOT	Logical NOT	$\overline{AR}$ → AR	None	1	↕	•	•	•


FUN 5. 3210
FUN 15 WM420
FUN 55 170
FUN 6. 2000
FUN 16 WM440
FUN 56 170
FUN 66 WM450
FUN 85
AR AND BCD constant "3210" → AR
AR AND M420 M421 → AR
ARL AND BIN constant "170" → ARL
AR OR BCD constant "2000" → AR
AR OR M440 M441 → AR
ARL OR BIN constant "170" → AR
AR EOR M450 M451 → AR
AR → AR

[**Explanation**]

1. FUN5 (ANDI) and FUN6 (ORI) instructions perform logical AND and OR operations between AR register data and constants 0000H to 9999H.
   FUN55 (BANDI) and FUN56 (BORI) instructions perform logical AND and OR operations between the lower 8 bits (AR_L) of AR regdster and constants 00 to FF. Because of the restriction peculiar to the progrmmer, a decimal entry is automatically converted into a hexadecimal value before logical AND/OR operations. For instance, entry $(170)_{10}$ is converted to $(AA)_{16}$.
2. FUN15 (AND) and FUN16 (OR) instructions perform logical AND/OR operations between AR register data and external input, external output, internal output or current value/preset value of timer/counter.
3. FUN66 (EXOR) instruction performs logical exclusive-OR operation between AR register data and external input, external output, internal output or current value/preset value of timer/counter.
4. FUN85 (WONT) instruction performs logical NOT operation of AR register data.

Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	Acc
FUN 7	CPEHI	Compare (≧)	AR≧constant……1→C AR<constant……0→C	constant (0000H～9999H)	2	•	•	↕	•
FUN 17	CPEH		AR≧I/O……1→C AR<I/O……0→C	WX, WY, WM. T/C 100～295	2	•	•	↕	•
FUN 57	BCPHI		AR_L≧constant……1→C AR_L<constant……0→C	constant (00～FF)	2	•	•	↕	•
FUN 8	CPEI	Compare(=)	AR=constant……1→C AR≠constant……0→C	constant (0000H～9999H)	2	•	•	↕	•
FUN 18	CPE		AR=I/O……1→C AR≠I/O……0→C	WX, WY, WM, T/C 100～295	2	•	•	↕	•
FUN 58	BCPEI		AR_L=constant……1→C AR_L≠constant……0→C	constant (00～FF)	2	•	•	↕	•
FUN 9	CPLI	Compare(<)	AR<constant……1→C AR≧constant……0→C	constant (0000H～9999H)	2	•	•	↕	•
FUN 19	CPL		AR<I/O……1→C AR≧I/O……0→C	WX, WY, WM, T/C 100～295	2	•	•	↕	•
FUN 59	BCPLI		AR_L<constant……1→C AR_L≧constant……0→C	constant (00～FF)	2	•	•	↕	•



T/C 50 current value → AR

(AR≧0200H)→C

C→M500

X0～X15→AR

AR →WM600

X20～X35→AR

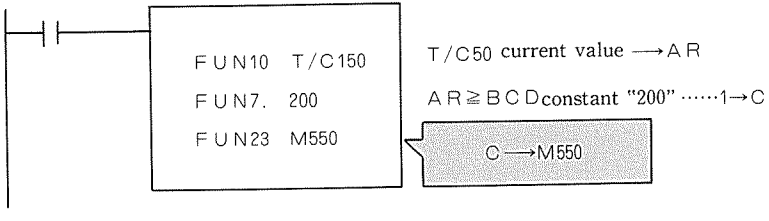
(AR＝WM600)→C

C→M555

[**Explanation**]

1. Compare instructions are classified into 3 types: ≧, = and<. Each type consists of 3 kinds of instructions. So nine kinds of compare instructions in total are selectable to suit the component. AR register and component data are compared as binary numbers without sign. If the result of comparison is true, carry C is set to ON. If it is false, carry C is set to OFF.
2. FUN7 (CPEHI) and FUN9 (CPLI) are instructions to compare AR register data with constants 0000H to 9999H. FUN57 (BCPHI), FUN58 (BCPEI) and FUN59 (BCPLI) are instructions to compare the lower 8 bits (AR_L) of AR registem with constants 00to FF. Because of the restriction peculiar to the programmer, a decimal entry is automatically converted into a hexadecimal value before comparison. For instance, entry $(255)_{10}$ is converted into $(FF)_{16}$.
3. FUN17 (CPEH), FUN18(CPE) and FUN19(CPL) are instructions to compareAR register data with external input, external output, internal output, timer/counter current value and preset value. Component data need not be BCD data (0000H through 9999H).

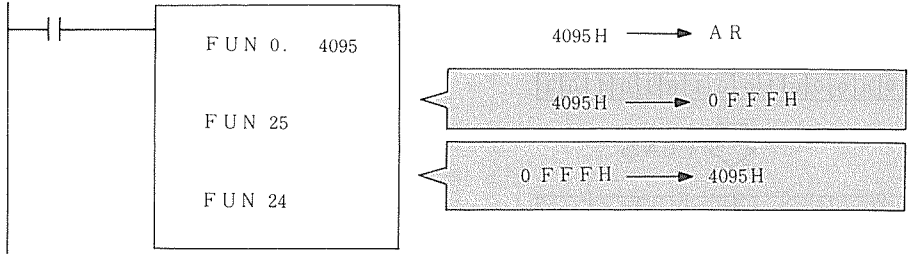
Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	Acc
FUN23	OUC	Out carry	C→I/O	Y, M	1	•	•	•	•
FUN83	CLC	Clear carry	C←0	None	1	•	•	0	•
FUN84	SEC	Set carry	C←1	None	1	•	•	1	•



[Explanation]

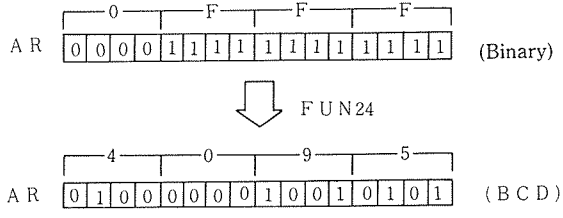
1. The FUN23 (OCU) instruction outputs carry C data to internal output (M) or external output (Y).
2. The FUN84 (SEC) instruction sets "1" to carry C. The FUN83 (CLC) resets carry C to "0"

Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	Acc
FUN24	BCD	BCD convert	AR $\xrightarrow{\text{BCD convert}}$ AR	None	1	↕	•	↕	•
FUN25	BNR	BIN convert	AR $\xrightarrow{\text{BIN convert}}$ AR	None	1	↕	•	↕	•



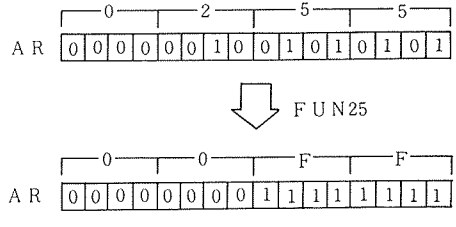
[Explanation]

1. The FUN24 (BCD) instruction converts the binary data in the AR register into BCD data. If the result of conversion is 4 digits or less, carry C turns OFF.



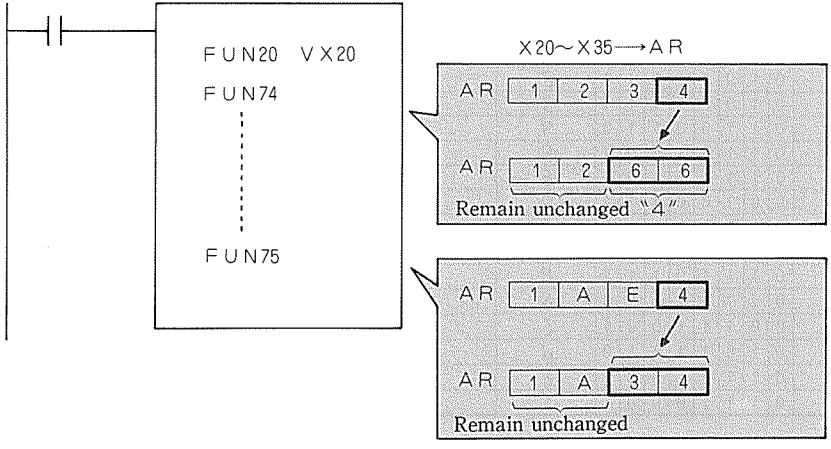
If the result of conversion overflows 4 digits, the AR register data is not converted (the contents of register remain unchanged) and carry C turns ON.

2. The FUN25 (NR) instruction converts the BCD data in the AR register into binary data. When the AR register contains BCD data before conversion, carry C turns OFF.



Before conversion, each digit of the AR register must be value in the range of 0 to 9. If the AR register data is within A to F, it will not be converted (the contents of register remain unchanged)and the value of carry C will become unreliable.

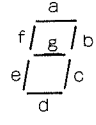
Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	ACC
FUN74	SEG	7-segment decode	AR$_{LL}$ $\xrightarrow{\text{7-segment convert}}$ AR	None	1	↕	•	•	•
FUN75	ASC	ASCII convert	AR$_{LL}$ $\xrightarrow{\text{ASCII convert}}$ AR	None	1	↕	•	•	•



[**Explanation**]

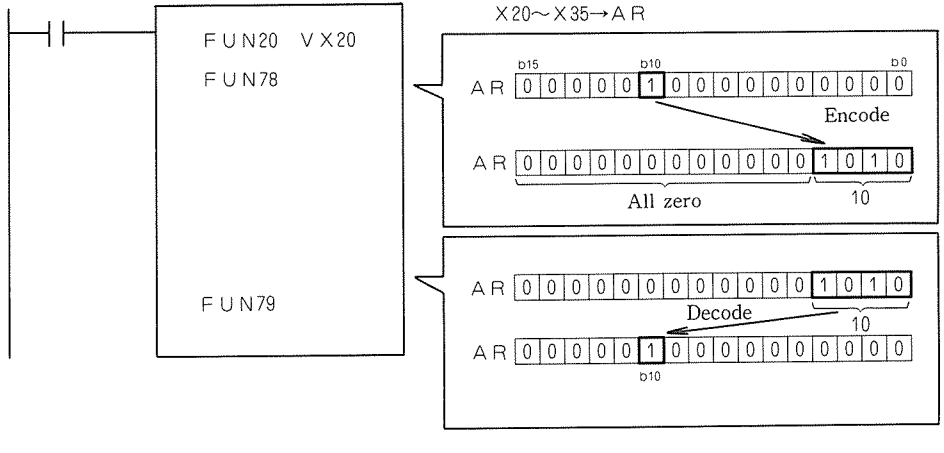
1. The FUN74 (SEG) instruction covers the lower 4 bit (AR_{LL}) data of AR register into a 7-segment display code and stores in the lower 8 bits (AR_L) of that register.
2. The FUN75 (ASC) instruction converts the lower 4-bit (AR_{LL}) data of AR register into an ASCII code and stores it in the lower 8 bits (AR_L) of that register.
3. The upper 8 bits (AR_H) of AR register remain unchanged before and after excution of the FUN74 (SEG) or FUN75 (ASC) instruction.
4. Shown below is a FUN74 (SEG) and FUN75 (ASC) conversion table.

Input data 4 bits	FUN74(SEG)									FUN75(ASC)
	Output data								Display	Output data
	g	f	e	d	c	b	a			
0	0	0	1	1	1	1	1		0	30
1	0	0	0	0	0	1	1	0	1	31
2	0	1	0	1	1	0	1	1	2	32
3	0	1	0	0	1	1	1	1	3	33
4	0	1	1	0	0	1	1	0	4	34
5	0	1	1	0	1	1	0	1	5	35
6	0	1	1	1	1	1	0	1	6	36
7	0	0	1	0	0	1	1	1	7	37
8	0	1	1	1	1	1	1	1	8	38
9	0	1	1	0	1	1	1	1	9	39
A	0	1	1	1	0	1	1	1	A	41
B	0	1	1	1	1	1	0	0	b	42
C	0	0	1	1	1	0	0	1	C	43
D	0	1	0	1	1	1	1	0	d	44
E	0	1	1	1	1	0	0	1	E	45
F	0	1	1	1	0	0	0	1	F	46



Display segments

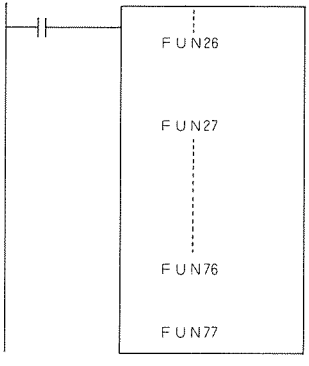
Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	ACC
FUN78	ENCOD	Encode	16→4 encode	None	1	↕	•	↕	•
FUN79	DECOD	Decode	4→16 decode	None	1	↕	•	↕	•

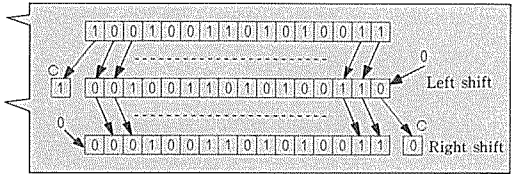


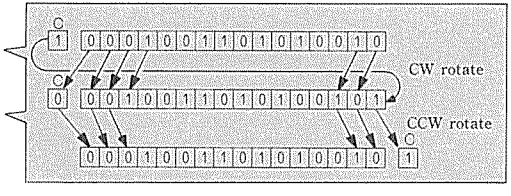
[**Explanation**]

1. The FUN78 (ENCODE) instruction sets into the AR the uppermost bit position (1 to 15), where "1" is set, among the bits of the register. In case all bits are 0, AR and C become 0 and 1, respectivenly. When two or more bits are 1, the uppermost bit is selected.
2. The FUN79 (DECODE) instruction sets 1 at the bit position corresponding to the value of AR register (0to 15) and clears all other bits to 0. In case the AR register value is 16 or more, AR and C become 0 and 1, respectively.

Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	ACC
FUN26	LSFR	Left shift	C← [AR] ←0	None	1	↕	•	↕	•
FUN27	RSFR	Right shift	0→ [AR] →C	None	1	↕	•	↕	•
FUN76	ROL	CW rotate	C← [AR] ←	None	1	↕	•	↕	•
FUN77	ROR	CCW rotate	→ [AR] →C	None	1	↕	•	↕	•



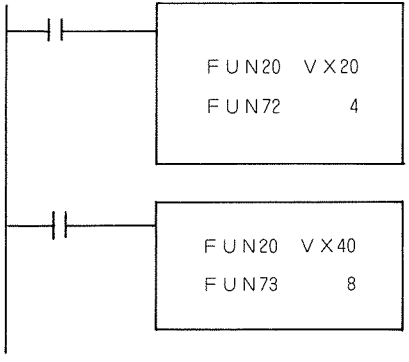




[**Explanation**]

1. The Fun26 (LSFR) instruction shifts AR register data 1 bit to the left. Upon shift, the least significant bit is padded with zero and the overflow bit is set to carry C.
2. The FUN27 (RSFR) instruction shifts AR register data 1 bit to the right. Upon shift, the most significant bit is paddedwith zero and the overflow bit is set to carry C.
3. The FUN76 (ROL) instruction shifts AR register data 1 bit to the left. Upon shift, the overflow bit is set to carry C and the least significant bit is padded with the previous data in the carry C.
4. TheFUN 77 (ROR) instruction shifts AR register data 1 bit to the right. Upon shift, the overflow bit is set to carry C and the most significant bit is padded with the previous data in the carry C.

Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	Acc
FUN72	MASKL	Mask	Masks by specified bits from uppermost bit.	0～255	2	↕	•	•	•
FUN73	MASKR		Masks by specified bits from lowermost bit.	0～255	2	↕	•	•	•



X20～X35→AR

AR 0 1 1 0 0 1 0 1 1 0 0 1 0 0 1 0

↓ Masking of 4 bits

AR 0 0 0 0 0 1 0 1 1 0 0 1 0 0 1 0

X40～X55→AR

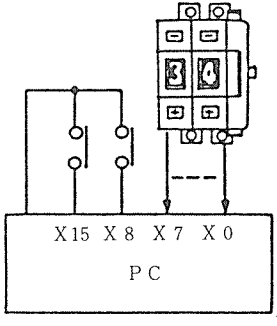
AR 0 1 1 0 0 1 0 1 1 0 0 1 0 0 1 0

↓ Masking of 8 bits

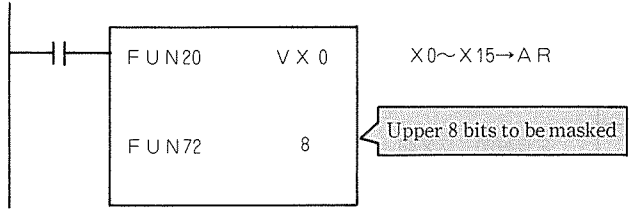
AR 0 1 1 0 0 1 0 1 0 0 0 0 0 0 0 0

[**Explanation**]

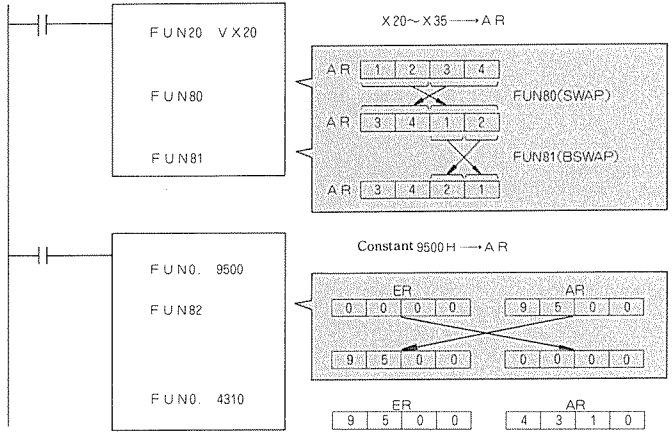
1. FUN72(MASKL) and FUN73 (MASKR) instructions mask the AR register data by the specified number of bits.
   The FUN72 (MASKL) instruction masks the data by the specified number of bits starting from the most significant bit (b_{15}).
   The FUN73 (MASKR) instruction masks the data by thespecified number of bits starting from the least significant bit (b_0).
   Even when 17 or more bits are specified, only 16 bits are validated.
2. Application example of mask instruction



When loading the 2-digit thumbwheel switch data into X0 through X7 to be followed by loading of ordinary input signals in X8 through X15, the switch data is also loaded into X8 through X15 automatically. This is because the FUN20 (LOADB) instruction operates on a data word of 16 bits long. To mask X8 through X15, use the FUN72 (MASKL) instruction.



Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	ACC
FUN80	SWAP	Exchange	$AR_H \rightleftarrows AR_L$	None	1	↕	•	•	•
FUN81	BSWAP		$AR_{LH} \rightleftarrows AR_{LL}$	None	1	↕	•	•	•
FUN82	XCG		AR ⇄ ER	None	1	↕	↕	•	•

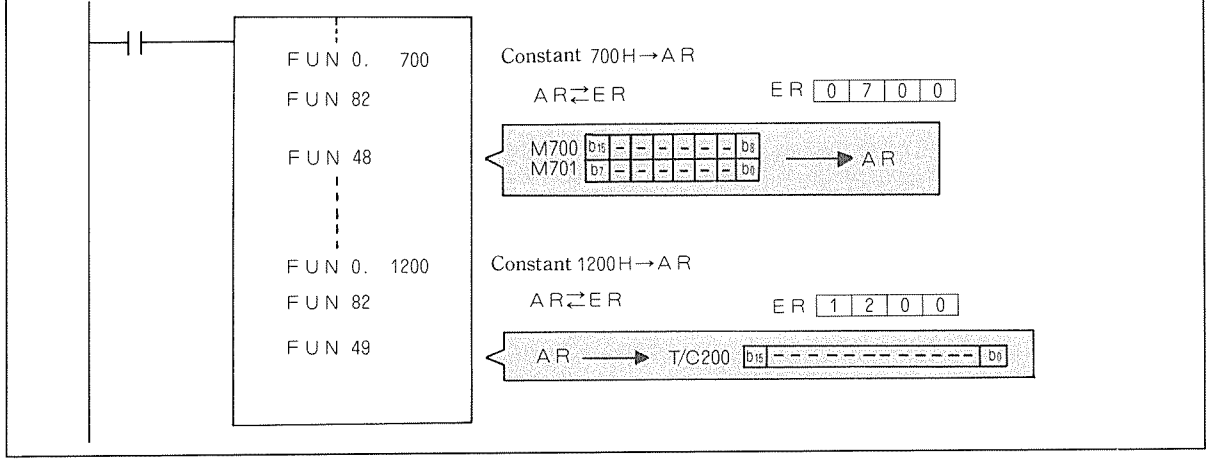


[**Explanation**]

1. The FUN80 (SWAP) instruction exchanges the upper byte (b_8 through b_{15}) and the lower byte (b_0 through b_7) of the AR register.
2. The FUN81 (BSWAP) instruction exchanges the upper nibble (b_4 through b_7) and lower nibble (b_0 through b_3) of lower byte in the AR register.
3. The FUN82 (XCG) instruction exchanges the AR register and ER register. TheFUN82 instruction is used for setting data in the ER register.

Concept of arithmetic instruction	Load	out	4-rule calculations				Logic	Compare	Carry	Convert	Shift	Mask	Exchange	Distribute/ extract
			Add	Subtract	Multiply	Divide								
110	112	116	122	124	126	128	130	133	136	137	143	145	147	149

Instruction	Abbreviation	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	Acc
FUN48	EX	Extract	Fetches data into AR from I/0 address-specified by ER.	None	1	↕	•	↕	•
FUN49	DB	Distribute	Outputs data from AR to I/0 address-spesifid by ER.	None	1	•	•	↕	•



[**Explanation**]

1. Data is to be exchanged between the I/O address-specified by the ER regitser and the arithmetic register AR. The ER register contains BCD data. The most significant digit 0 and1 stand for usual I/O and timer/counter, respectively. FUN48 (EX) fetches data into AR, and FUN49 (DB) outputs data to I/O.
2. CR will become 1 if either instruction is executed with an undefined I/O specified by theER register (only when Acc = 1).

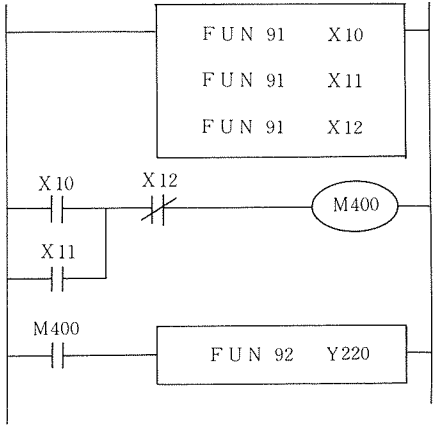
1	PRINCIPLE OF PC	
2	INPUT/OUTPUT AND NUMBERS	
3	PROGRAMMING	3.1 Basic Instructions
		3.2 Application Instructions (I)
		3.3 Arithmetic Instructions
		3.4 Application Instructions (II)

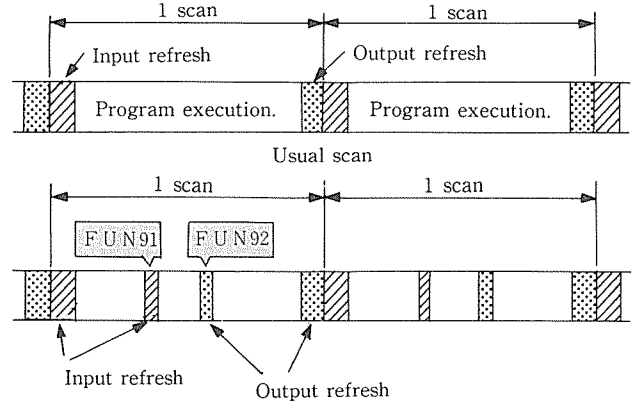
Application Instructions (II)

Classification	Instruction	Symbol	Name	Function	Component	No. of words	Change in register AR	ER	CR	Acc	Reference page
Refresh	FUN91	REFX	I/O refresh	Refreshes specified input.	X	1	●	●	●	●	153
	FUN92	REFY		Refreshes specified output.	Y	1	●	●	●	●	153
Interrupt	FUN93	INT	Declares interrupt.	Argument 2: Declares interrupt at fixed intervals of 10 ms.	Argument 2	2	–	–	–	–	155
	FUN94	RTI	Recovery from interrupt	Recovery from interrupt	None	1	Value before interrupt				155
Subroutine	FUN42	CALL	Subroutine	Calls subroutine.	Arguments 0 to 63	2	●	●	●	●	157
	FUN43	SB		Defines subroutine.	Arguments 0 to 63	2	–	–	–	–	157
	FUN44	RTS		Recovery from subroutine.	None	1	Value before subroutine call				157

●: Register remains unchanged
↕: Register changed
–: Register cleared

Instruction	Symbol	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	Acc
FUN91	REFX	I/O refresh	Inputs specified I/O.	X	1	•	•	•	•
FUN92	REFY		Outputs specified I/O.	Y	1	•	•	•	•





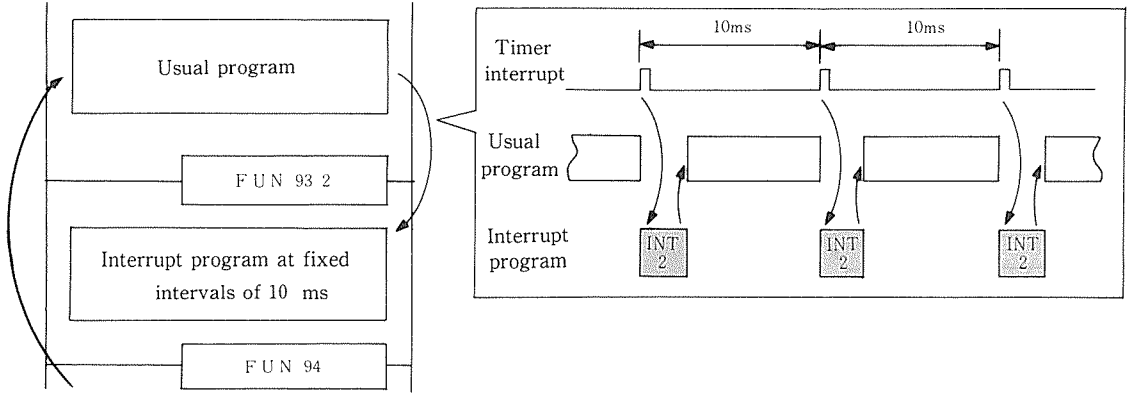
I/O refresh according to FUN91/FUN92 instructions

(Specified input and output can be refreshed in the middle of scan.)

[**Explanation**]

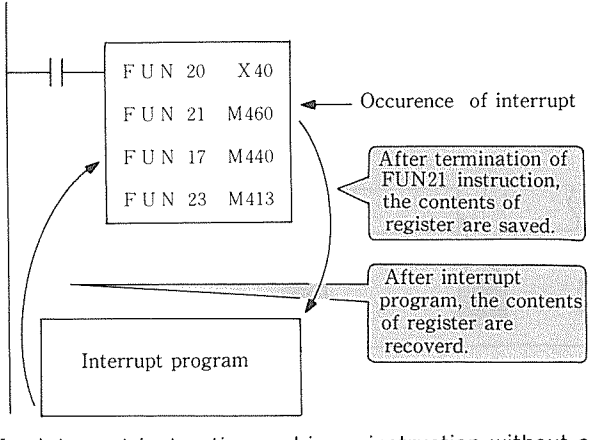
1. FUN91 (REFX) is imputrefresh instruction. It rewrites data memory of the specified input number in the course of scan (upon its execution). This instruction does not have a start condition.
2. FUN92 (REFY) is output refresh instruction. It rewrites the specified output number and its data memory the same as in the current Acc register during scan (upon its execution).
3. Input signals shorter than scan time can be acquired by uniform allocation of the refresh instruction at several locations in the entire program.

Instruction	Symbol	Name	Function		Component	No. of words	Change in register			
							AR	ER	CR	Acc
FUN93	INT	Declares interrupt.	Argument 2	Interrupt at fixed intervals of 10 ms	Argument 2	2	–	–	–	–
FUN94	RTI	Recovery from interrupt	Recovery from interrupt		None	1	Value before interrupt			



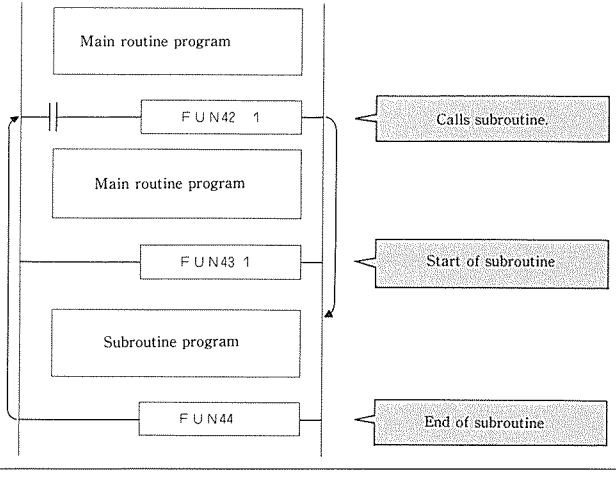
[**Explanation**]

1. An interrupt program is to be located next to a usual program. These programs are to be separated by the FUN93 (INT) instruction. FUN99 (END) is not used. The end of interrupt program must always be the FUN94 (RTI) instruction. Neither FUN93 nor FUN94 requires start condition.
2. Interrupt program is excuted every 10 ms when it is written between the FUN93 2 (INT2) and FUN94 (RTI) instructions after a usual program.
3. When applying an interrupt, the instruction under execution is terminated and the relevant interrupt program is excuted once. On this occasion, the contents of register are automatically saved.
   After termination of the interrupt program, the usual program before interrupt program returns and the contents of register are recovered.



4. Interrupt instruction and jump instruction without adressing cannot coexist.

Instruction	Symbol	Name	Function	Component	No. of words	Change in register			
						AR	ER	CR	Acc
FUN42	CALL	Subroutine	Calls subroutine.	Arguments O to 63	2	•	•	•	•
FUN43	SB		Defines subroutine	Arguments O to 63	2	—	—	—	—
FUN44	RTS		Recovery from subroutine	None	1	Value before subroutine call			



[**Explanation**]

1. Subroutine program is to be located next to the main routine program. At the head of subroutine program, the FUN43 (SB) instruction is required to be set. Each subroutine program must be terminated by the FUN44 (RTS) instruction. Subroutine can be called by the FUN42 (CALL) instruction.
2. The FUN99 (END) instruction is unnecessary between the main routine program and subroutine program. Neither FUN43 nor FUN44 requires the start condition.
3. In a subroutine, jump and master control instructions are unusable.