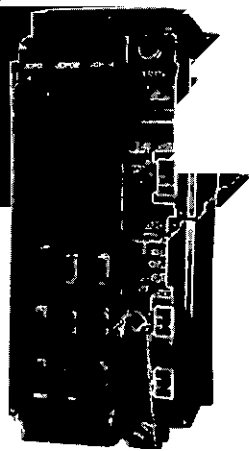
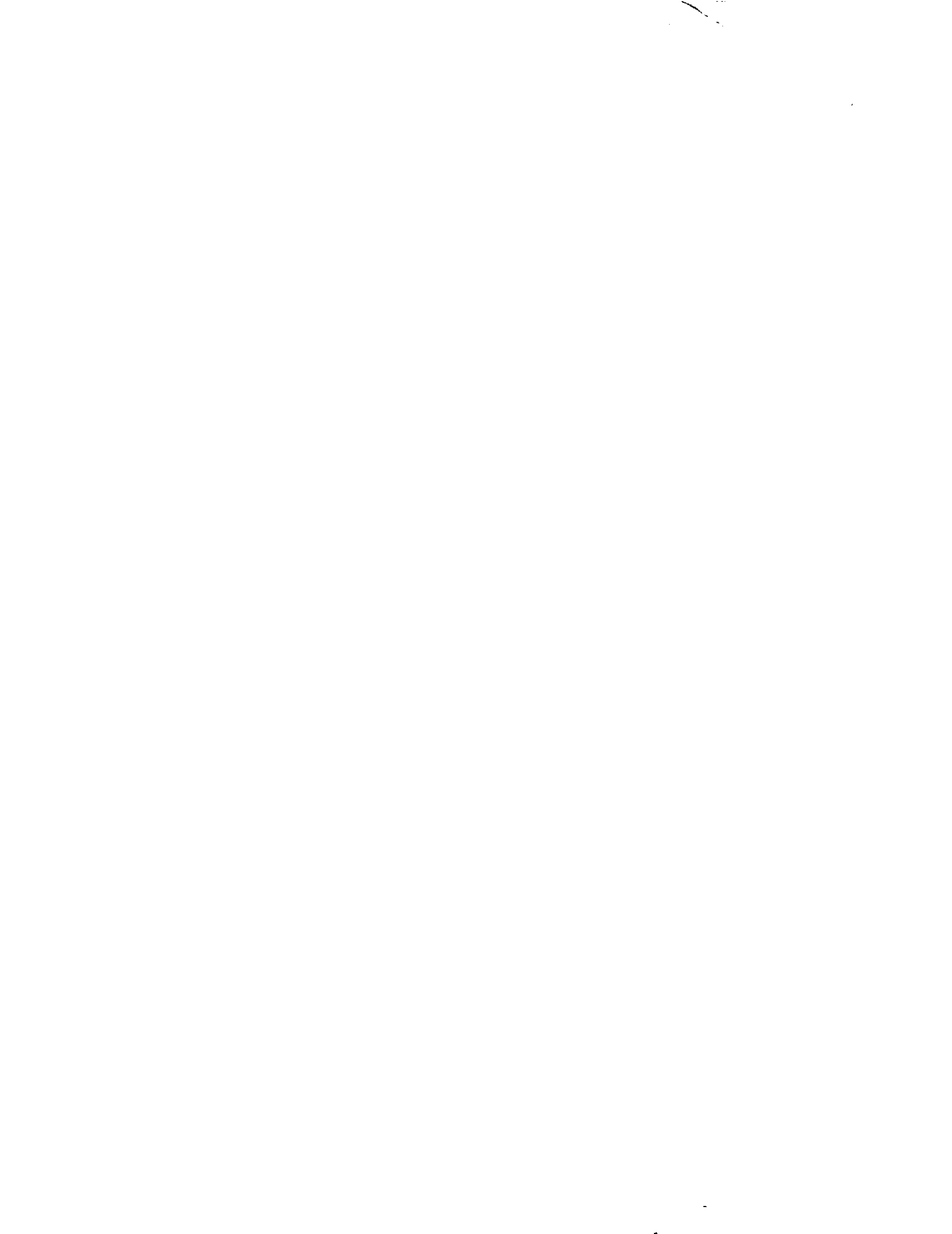


YASNAC J300 PLC PROGRAMMING MANUAL

```
RUNNING          RUN          000600 N00000
;
G40 G49 G80 ;
G91 G30 Y0 Z0 M05 ;
N1 T09 M06 ;

UNIVERSAL      INCREMENT      G/M CODE
X      49.042 X      0.000 G00 G80
Y      111.296 Y     0.000 G17 G98
Z       6.638 Z     0.000 G90 G52
                               G67
                               G94
T NO :T0000      ACT S:S      0      G21
FEED :F          ORDER:S     0      G40
```





FOREWORD

The Programmable Logic Controller (hereafter referred to as PLC) for YASNAC J300 series NC functions as the interface between the YASNAC NC and a machine tool to execute sequence control specific to the machine tools simply and efficiently by software.

For the edition of PLC sequence, a personal computer can be used. This feature allows of-line mode programming. The created programs can be checked by using the sequence debugging board (model: JANCD-JCP02-3, name: JXSD) installed in the NC after transmitting the programs to the NC. This feature allows efficient program development.

PLC sequence can be edited in online mode. The online mode editing can be executed by using the standard hardware without requiring the JXSD board, permitting simple modifications to the PLC sequence which is already in operation. It is of course possible to develop a sequence program without using a personal computer.

This manual explains the PLC instructions, program debugging methods, the procedure used for writing the programs to flash ROM, etc. so that a series of operation from PLC program development to writing the program to flash PROM can be performed smoothly.

CONTENTS

FOREWORD	i
1 SYSTEM CONFIGURATION	
1.1 SYSTEM CONFIGURATION OF YASNAC J300 PLC SYSTEM	1 - 2
2 SEQUENCE PROGRAM DEVELOPMENT PROCEDURE	
2.1 SEQUENCE PROGRAM DEVELOPMENT	2 - 2
3 PLC PROGRAM SPECIFICATIONS	
3.1 BASIC SPECIFICATIONS	3 - 2
3.2 PROGRAM FUNCTIONS	3 - 3
3.3 INPUT/OUTPUT SPECIFICATIONS	3 - 4
4 SEQUENCE CONTROL METHOD	
4.1 DIFFERENCES IN OPERATION	4 - 2
4.2 LEVEL AND OPERATION OF SEQUENCE PROGRAM	4 - 3
4.3 RELATIONSHIP BETWEEN THE LADDER STOP COUNT AND THE SEQUENCE TASK	4 - 6
4.4 SEQUENCE PROGRAM MEMORY CAPACITY AND MEMORY CONFIGURATION	4 - 7
5 ADDRESS NUMBERS AND ADDRESS MAP	
5.1 ADDRESS MAP	5 - 2
5.2 ADDRESS MAP AND DISPLAY SYMBOLS	5 - 3
6 PLC INSTRUCTIONS	
6.1 BASICS OF PLC INSTRUCTIONS	6 - 2
6.2 TYPES AND LIST OF INSTRUCTIONS	6 - 3

6.3	RELAY INSTRUCTIONS	6 - 8
6.4	TIMER INSTRUCTIONS	6 - 15
6.5	REGISTER INSTRUCTIONS	6 - 17
6.6	CONTROL INSTRUCTIONS	6 - 41
6.7	MACRO INSTRUCTIONS	6 - 44

7 JXSD OFFLINE SYSTEM

7.1	OUTLINE OF THE JXSD OFFLINE SYSTEM	7 - 2
7.1.1	Operating Environment	7 - 2
7.1.2	Execution Files	7 - 2
7.1.3	Outline of the Execution Files	7 - 2
7.1.4	Ladder Program Development Procedure	7 - 3
7.2	SOURCE FILE	7 - 6
7.2.1	Source File Format	7 - 6
7.3	COMPILER	7 - 18
7.3.1	7Compiler Operation	7 - 18
7.3.2	Compiler Error List	7 - 19
7.3.3	Compiler Check Items	7 - 19
7.4	LINKER	7 - 21
7.4.1	Object Data and Linker Processing	7 - 21
7.4.2	Linker Operation	7 - 22
7.4.3	Linker Output File	7 - 23
7.5	REMOTE CONTROLLER OPERATION	7 - 24
7.5.1	Connecting the JXSD to PLC	7 - 24
7.5.2	Starting the Remote Controller	7 - 25
7.5.3	Description of Screen Display Information	7 - 26
7.5.4	Operation of Remote Controller	7 - 28
7.6	LIST OF ERROR MESSAGES AND WARNING MESSAGES	7 - 33
7.6.1	Error Messages	7 - 33
7.6.2	Warning Messages	7 - 33

8 ONLINE EDITING

8.1	OUTLINE OF ONLINE EDITING	8 - 2
8.1.1	Creating a Sequence Program Newly	8 - 2
8.1.2	Creating a Sequence Program by Modifying the Existing Sequence Program	8 - 3
8.2	FUNCTION TREE AND DISPLAY SCREENS	8 - 4
8.2.1	Function Tree	8 - 4
8.2.2	Ladder Display Screen	8 - 5
8.3	LADDER DISPLAY FUNCTION	8 - 6
8.3.1	BT/TOP (Bottom/Top) Function	8 - 6
8.3.2	SYM DIS (Symbol Display) Function	8 - 6
8.3.3	NET SEL (Net Selection) Function	8 - 7
8.3.4	GO/STP (Run/Stop) Function	8 - 8
8.4	NET EDITING FUNCTION	8 - 10
8.4.1	Selection of Edit Mode	8 - 11
8.4.2	Keys Used for Editing the Ladder	8 - 15
8.4.3	Inputting Contacts	8 - 19
8.4.4	Inputting Vertical and Horizontal Lines	8 - 22
8.4.5	Inputting Register Instructions	8 - 23
8.4.6	Canceling the Net Edit Function	8 - 34
8.4.7	Exiting the Edit Function	8 - 34
8.5	TABLE EDIT FUNCTION	8 - 38
8.5.1	Editing the Data in the Conversion Table	8 - 39
8.5.2	Editing the Data in the Message Table	8 - 40
8.5.3	Editing the Data in the Symbol Table	8 - 41
8.6	INPUT/OUTPUT FUNCTION	8 - 43
8.6.1	Downloading the Sequence Program	8 - 43
8.6.2	Uploading the Sequence Program	8 - 45
8.7	SEQ STS (SEQUENCE STATUS) FUNCTION	8 - 47
8.7.1	Display of Sequence Status	8 - 47
8.7.2	INITI (Initialization) Function	8 - 48
8.8	LIST OF MESSAGES	8 - 51
8.8.1	List of Messages	8 - 51
8.8.2	List of Warning Messages	8 - 52
8.8.3	List of Alarm Messages	8 - 52

9 DOWNLOADING AND UPLOADING LADDER PROGRAM

- 9.1 DOWNLOADING LADDER PROGRAM
(PC CARD → FLASH ROM) 9 - 2

- 9.2 UPLOADING LADDER PROGRAM
(FLASH ROM → PC CARD) 9 - 4

1

SYSTEM CONFIGURATION

Chapter 1 describes the system configuration of YASNAC J300 PLC system.

1.1	SYSTEM CONFIGURATION OF YASNAC J300 PLC SYSTEM	1 - 2
-----	--	-------

1.1 SYSTEM CONFIGURATION OF YASNAC J300 PLC SYSTEM

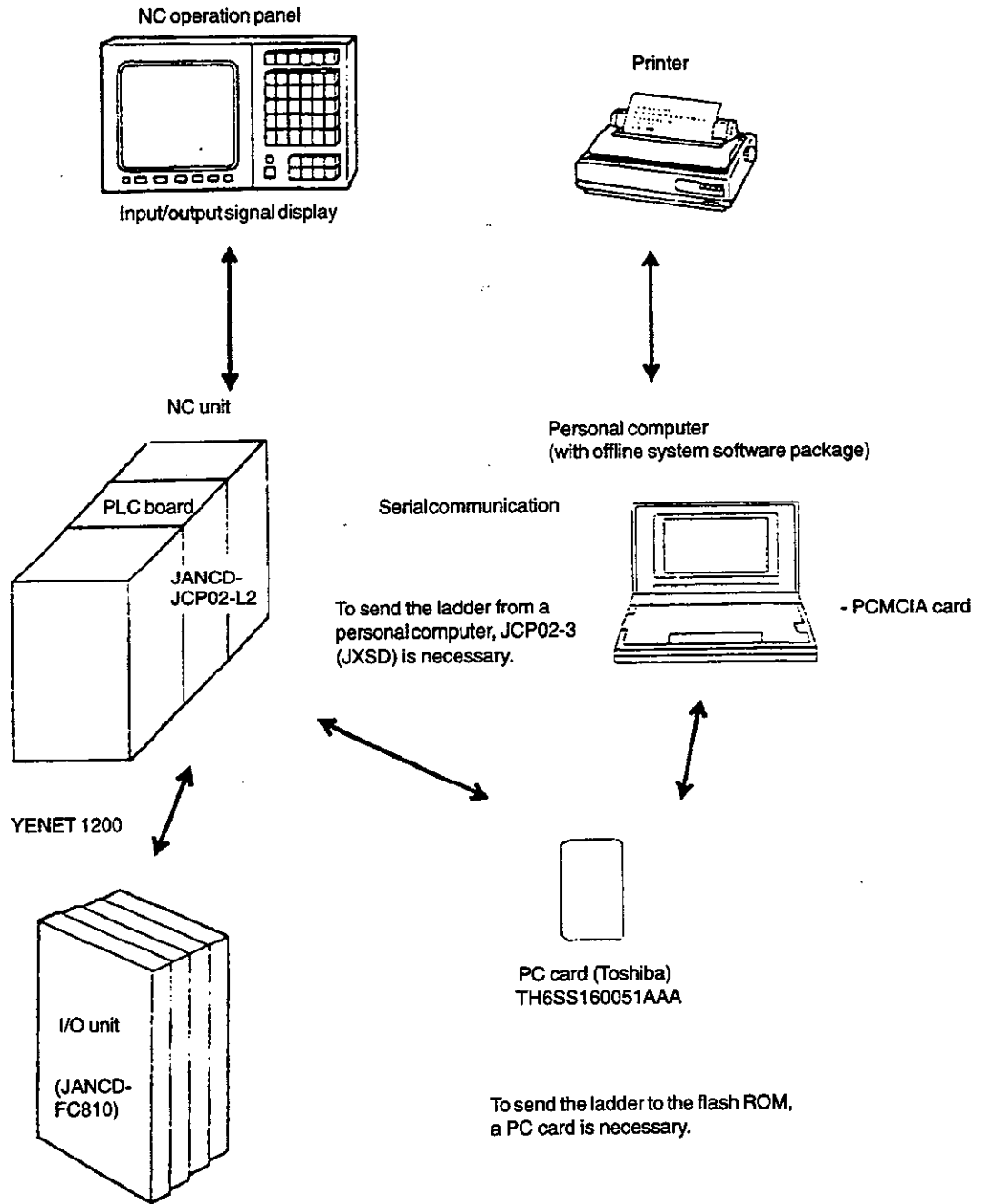


Fig. 1.1

2

SEQUENCE PROGRAM DEVELOPMENT PROCEDURE



Chapter 2 describes the procedure of sequence program development – offline and online mode operation.

2.1 SEQUENCE PROGRAM DEVELOPMENT	2 - 2
---	-------

2.1 SEQUENCE PROGRAM DEVELOPMENT

Processing on the Desk

Processing at Personal Computer

Processing at NC Unit

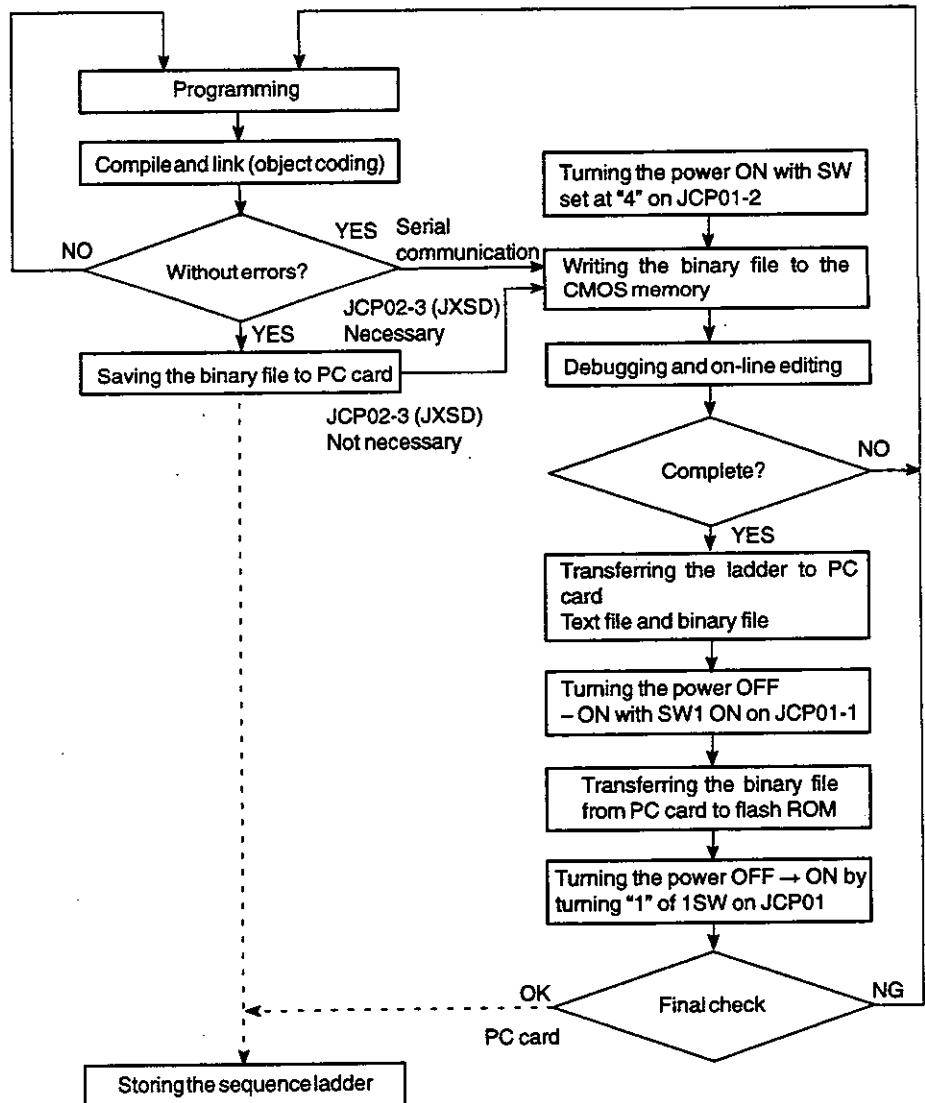
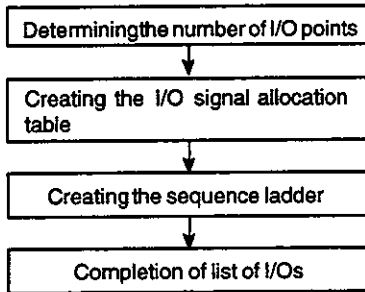


Fig. 2.1

3

PLC PROGRAM SPECIFICATIONS

Chapter 3 describes the specifications of PLC programs.



3.1	BASIC SPECIFICATIONS	3 - 2
3.2	PROGRAM FUNCTIONS	3 - 3
3.3	INPUT/OUTPUT SPECIFICATIONS	3 - 4

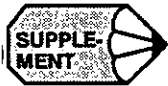
3.1 BASIC SPECIFICATIONS

The basic specifications of PLC programs are indicated below.

Control method	Scanning method
Processing time	0.5 μ s/step (approximate value)
High-speed processing scan time	4 ms
Low-speed processing scan time	4 ms \times n
Note: Value "n" is determined by the high-speed processing capacity and total program capacity.	
Program memory capacity	
Basic	256K bytes
Program	128K bytes
Table symbol	128K bytes
Note: 128K bytes are equivalent to approximately 32K steps of basic instructions.	
Instructions	
Basic instruction	61 kinds
Macro instruction	22 kinds

3.2 PROGRAM FUNCTIONS

Internal relays	11960 points
Registers	1495 registers (8 bits/register)
Timers	188 timers (5 types)
8 msec to 2.4 sec	40 timers
50 msec to 12.75 sec	60 timers
100 msec to 25.5 sec	60 timers
1 sec to 255 sec	20 timers
1 min to 255 min	8 timers
Sequence parameters	100 sets (8 bits/set)
Keep relays	7200 points
Battery back-up memory	2900 sets



1. Internal relays and registers occupy the same addresses and the addresses used for internal relays cannot be used for registers. Similarly, the addresses used for registers cannot be used for internal relays.
2. Keep relays and battery back-up memory occupy the same addresses, and the addresses used for keep relays cannot be used for battery back-up memory. Similarly, the addresses used for battery back-up memory cannot be used for keep relays. Note that the keep registers (#8000 to #9999) cannot be used for the keep relays.

3.3 INPUT/OUTPUT SPECIFICATIONS

General-purpose input/output ports are installed on the I/O module (JANCD-FC810, FC860, FC861) and the JSP board (JSP02) in the NC operation panel.

The number of I/O points on each module is indicated below.

Module JANCD-	Number of Input Points	Number of Output Points	Remark
FC810, FC860	112	96	For machine operation panel
FC861	64	56	
JSP02	64	56	



An input/output port is incorporated in the control board (JSP02) in the NC operation panel. Therefore, if modules FC810/FC860 are added, addition of a maximum of 4 boards (max. input: 512 points; max. output: 440 points) is possible, and if module FC861 is added, addition of a maximum of 9 boards (max. input: 640 points, max. output: 560 points) is possible.

4

SEQUENCE CONTROL METHOD

Chapter 4 describes the sequence control method.

Sequence control by PLC is executed sequentially by the software, which differs from the ordinary control by relay circuits in which processing is executed simultaneously. Due to this characteristic, the sequence control by PLC results in considerably different operation from ordinary relay circuit processing. When developing programs, this must be completely understood.



4.1	DIFFERENCES IN OPERATION	4 - 2
4.2	LEVEL AND OPERATION OF SEQUENCE PROGRAM	4 - 3
4.3	RELATIONSHIP BETWEEN THE LADDER STOP COUNT AND THE SEQUENCE TASK	4 - 6
4.4	SEQUENCE PROGRAM MEMORY CAPACITY AND MEMORY CONFIGURATION	4 - 7

4.1 DIFFERENCES IN OPERATION

There are two types of operation modes in the sequence control – relay sequence and PLC sequence.

(1) Relay Sequence

All devices are processed simultaneously.

(2) PLC Sequence

Devices are processed sequentially and the ladder is executed repeatedly in a fixed period. This period is called the scan time.

(Scan time example: 4 msec × n times)

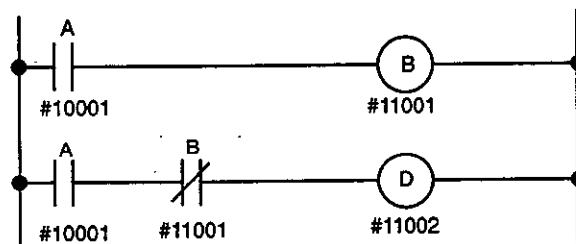


Fig. 4.1

The PLC sequence ladder given above operates in the following sequence. The operation is not processed simultaneously.

- ① The status of contact A is read.
- ② The read status is output to internal relay B.
- ③ The status of contact A is read.
- ④ AND operation is executed between the status of contact A and the status of NC contact of relay B.
- ⑤ The result of AND operation is output to internal relay D.

As the result of sequential processing, internal relay D is never turned ON.

However, if the same ladder is executed in the relay sequence, relay D is momentarily turned ON (one-shot operation).

As discussed above, programming must always be carried out taking into consideration that processing by the PLC is executed sequentially.

4.2 LEVEL AND OPERATION OF SEQUENCE PROGRAM

Length of time necessary for the execution of one cycle of a sequence program is called the scan time. The scan time of this model of PLC is indicated below.

- High-speed processing scan time: 4 msec
- Low-speed processing scan time: 4 msec × n

This means, with this PLC, the sequence program can be processed by dividing it into a high-speed processing part and low-speed processing part. Therefore, the sequence program must be written in the format indicated below.

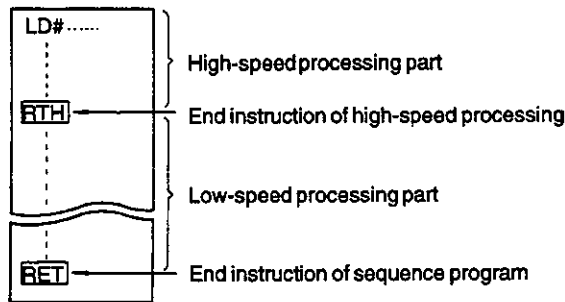


Fig. 4.2

As indicated above, the sequence program that requires high-speed processing should be entered first and the sequence program for which low-speed processing is acceptable should be entered after that.

(3) Operation Time Chart

The operation time chart of a sequence program is indicated below.

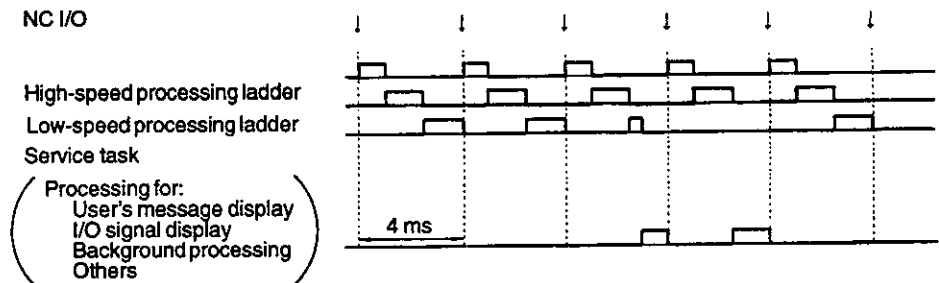


Fig. 4.3

(a) High-speed processing sequence program

The high-speed processing sequence program, from the beginning of the sequence program up to the RTH instruction, is executed once every 4 msec or less as shown in the time chart above.

During the processing of the high-speed sequence program, the input status remains unchanged.

(b) Low-speed processing sequence program

The low-speed processing sequence program entered following the RTH instruction is divided into "n" sections and one of these sections is executed in the remaining time in each 4-msec interval. That is, the low-speed processing sequence program is executed once in "4 msec × n".

As seen above, value "n" will be smaller as the total program capacity and the high-speed processing program capacity are smaller.

Since the low-speed processing sequence program is executed in several sections, the input status will be changed during its execution. Therefore, the precautions indicated in item (3) below must always be observed.

(c) Reading the input state

At the beginning of the 4-msec interval, the status of all inputs is read into the PLC collectively.

(d) Outputting the output status

At the beginning of the 4-msec intervals, the previous output status is output collectively.

(4) Precautions on High-speed Processing Sequence Program

The high-speed processing sequence program treats only the portion where high-speed response is required, such as counting the contact ON/OFF.

Therefore, this should be limited only to the requisite program. The capacity must be less than 1000 steps when converted into the contact instructions.

(5) Precautions on Low-speed Processing Sequence Program

- ① Scan time of the low-speed processing sequence program is influenced by the total capacity of the sequence program. It is calculated by "4 msec × n".

The sequence program size that can be processed in a 4-msec interval is approximately 3000 steps in contact instructions. This size is the total of high-speed processing sequence program and low-speed processing sequence program.

- ② Since the low-speed processing sequence program is executed in several sections, the status of inputs will be changed during the execution of the program. Therefore, the inputs that are used for the execution of the low-speed processing sequence program should be taken into the internal relays at the start of the low-speed processing sequence program, and for the execution of the low-speed processing sequence program, the contacts of the relays where the inputs have been received should be used as the input signals.

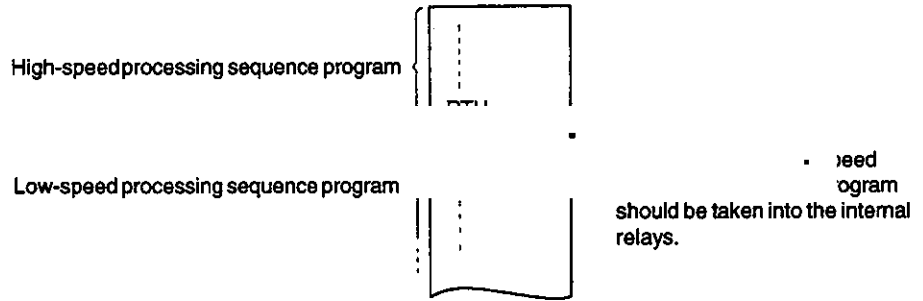
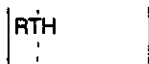


Fig. 4.4

By creating the program in this manner, one cycle of the low-speed processing sequence program can be executed under the same input signal status.

- ③ If the results of the high-speed processing sequence program are output to the low-speed processing sequence program, the same consideration must be given to the creation of the program.
- ④ The signals that should not be output until one cycle of the low-speed processing program is executed should not be directly output to the PLC address used for external outputs. Such signals should first be input to the internal relays and they should be connected to the external output addresses at the end of the low-speed processing sequence program.



— Write the outputs that should be output externally after the execution of one cycle of the low-speed processing sequence program at this part.

Fig. 4.5

4.3 RELATIONSHIP BETWEEN THE LADDER STOP COUNT AND THE SEQUENCE TASK

The service task includes the following:

- User's message display processing
- I/O signal display processing
- Background processing
- Others

The task operates during the period, from the completion of one scan of the low-speed processing sequence program and until the start of the next scan.

The ladder stop count (the number of times the low-speed processing ladder should be stopped) should be set based on the load during the processing of the service task.

Recommended value: 1

4.4 SEQUENCE PROGRAM MEMORY CAPACITY AND MEMORY CONFIGURATION

The sequence program is finally written to the flash memory (ROM).

The program memory of this PLC can be divided into the areas indicated below.

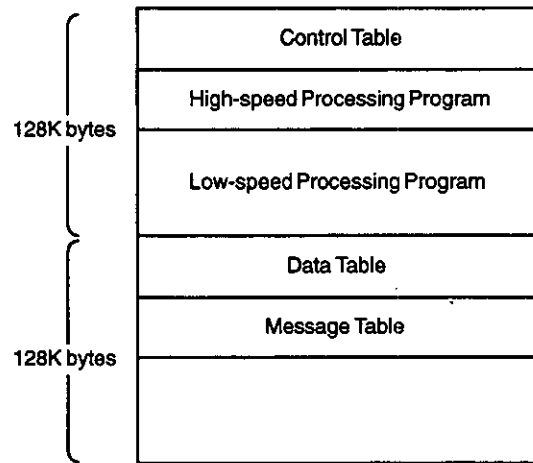


Fig. 4.6

Generally, relay instructions occupy 3 to 7 bytes and other instructions 1 to 25 bytes. Assuming that one instruction occupies an average of 4 bytes, 128K byte memory area is equivalent to 32K steps ($128K/4 = 32 K$).

5

ADDRESS NUMBERS AND ADDRESS MAP

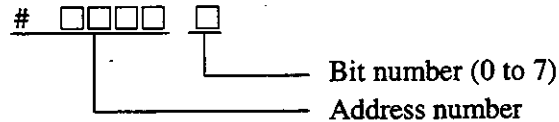
Chapter 5 describes the address numbers and address map.

5.1	ADDRESS MAP	5 - 2
5.2	ADDRESS MAP AND DISPLAY SYMBOLS	5 - 3



5.1 ADDRESS MAP

When creating a sequence program, input/output signals of PLC, internal relays, timers, battery back-up memory and other devices in the PLC are all designated by an address number (four-digit number following #) and a bit number (bit 0 to bit 7).



- a) Name of eight points of a signal
- b) Name of one-byte (eight bits) data

(1) Designation of I/O signals, Internal Relays, and Other Devices (One-bit Device)

The devices which have one-bit information are designated by a five-digit number (address number + bit number) following “#” as indicated below.

Device	Designation
1 I/O signals	
2 Internal relays	
3 Keep relays	

In this case, the address number has the same meaning as a) explained above, and it can be considered to be the name assigned collectively to eight points of a signal.

(2) Designation of the Registers, Timers, and Other Devices (One-byte Devices)

The devices which have one-byte (eight bits) information are designated by the address number. In this case, the address number has the same meaning as b) explained above, and it can be considered to be the name assigned to one-byte data.

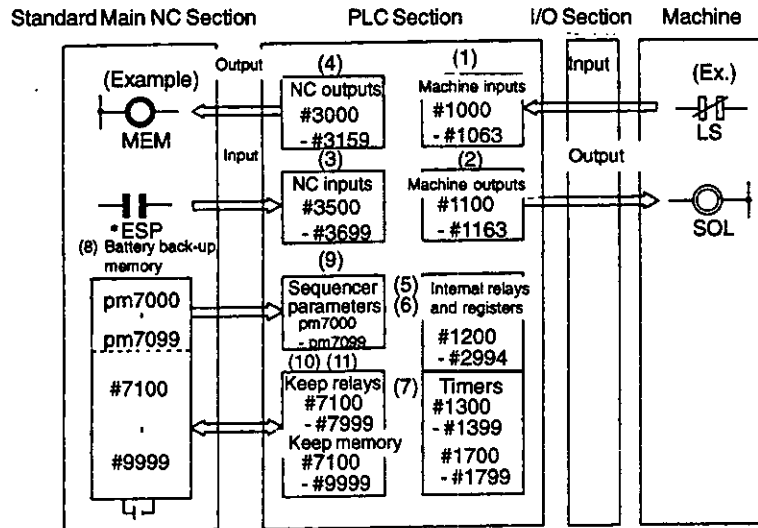
Device	Designation
4 Registers	
5 Timers	
6 Sequencer parameters	
7 Keep memory	

Note: With some types of instructions, designation of “#1500”, for example, specifies two bytes of “#1500” and “#1501”.

Example: PUSH #1500

5.2 ADDRESS MAP AND DISPLAY SYMBOLS

The address map and the relationship with external devices are shown below.

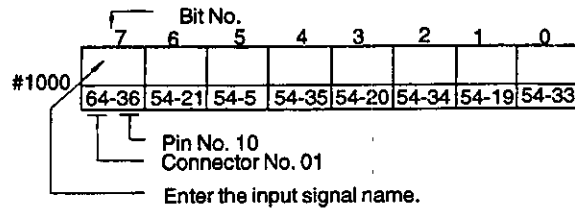


(1) Addresses of Input Signals from the Machine (#1000 to #1063)

For the signals input from the machine operation panel and electric control panel, such as those of pushbutton switches and limit switches, addresses #1000 to #1063 are assigned. The correspondence between the address and the input signal should be determined by the machine tool builder.

- ① One bit of address (#1000s) corresponds to one point of input signals.
- ② Address number and bit number are determined depending on the pin number and the connector number of the I/O board where the input signal is connected.

Example:



③ The input signals of #1000s are expressed by the following symbols.

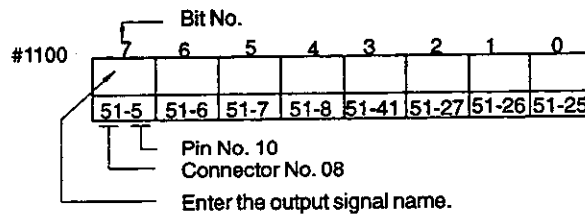


(2) Addresses of Output Signals to the Machine (#1100 to #1163)

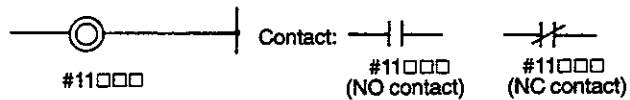
For the signals output to the machine operation panel and electric control panel, such as the signals of lamps and solenoids, addresses #1100 to #1163 are assigned. The correspondence between the address and the output signal should be determined by the machine tool builder.

- ① One bit of address (#1100s) corresponds to one point of output signals.
- ② Address number and bit number are determined depending on the pin number and the connector number of the I/O board where the output signal is connected.

Example:



- ③ The output signals of #1100s are expressed by the following symbols.



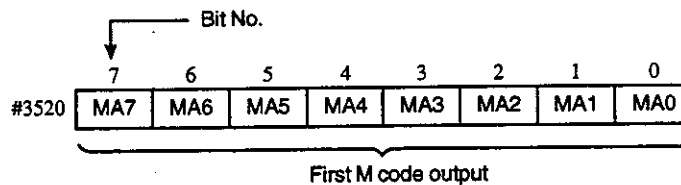
(3) Addresses of Input Signals from the NC's Main Section (#3500 to #3799)

For the signals input from the NC main section, in other words, the signals output from the NC main section to the PLC, such as M-BCD signal, addresses #3500 to #3799 are assigned.

The correspondence between the signal name and the address is determined by the NC and cannot be changed.

- ① One bit of address (#3500 to #3799) corresponds to one point of input signals.

Example:



- ② The input signals of #3500 to #3799 are expressed by the following symbols.



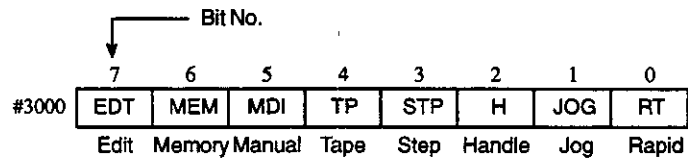
(4) Addresses of Output Signals to the NC Main Section (#3000 to #3159)

For the signals output from the PLC to the NC main section, such as EDIT and MEM mode selection signals, addresses #3000 to #3159 are assigned.

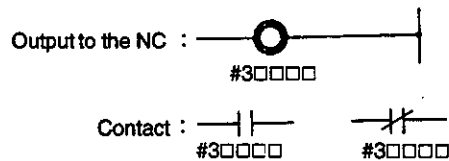
The correspondence between the signal name and the address is determined by the NC and cannot be changed.

- ① One bit of address (#3000 to #3159) corresponds to one point of output signals.

Example:



- ② The output signals of #3000 to #3159 are expressed by the following symbols.

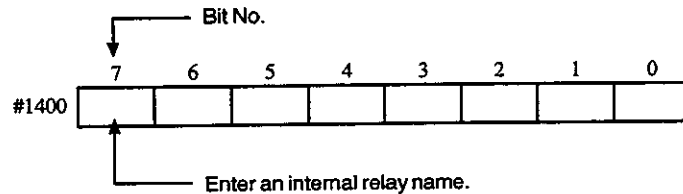


- (5) Addresses of Internal Relays (#1200 to #2994; excluding #1300 to #1399 and #1700 to #1799)

For the internal relays that can be used only in the PLC to create a sequence program, addresses #1200 to #2994 (excluding #1300 to #1399 and #1700 to #1799) are assigned.

- ① One bit of address of #1400s, for example, corresponds to one piece of internal relay.

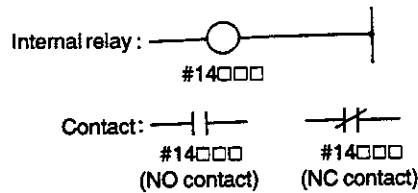
Example of I/O list:



- ② The number of usable internal relays is indicated below.

$$500 \text{ bytes} \times 8 \text{ bits} = 4000 \text{ relays}$$

- ③ The internal relay and its contact are expressed by the following symbol.



There are no limits to the number of contacts (NO and NC contacts) as long as the program capacity is not exceeded.

- ④ The addresses used for registers cannot be used for internal relays.

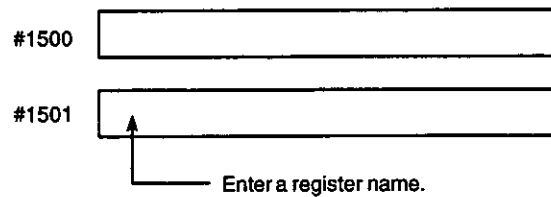
- (6) Addresses of Registers (#1200 to #2994; excluding #1300 to #1399 and #1700 to #1799)

For the general-purpose one-byte (eight bits) register, addresses #1200 to #2994 (excluding #1300 to #1399 and #1700 to #1799) are assigned.

These registers are used for register instructions and workpiece address for macro instructions.

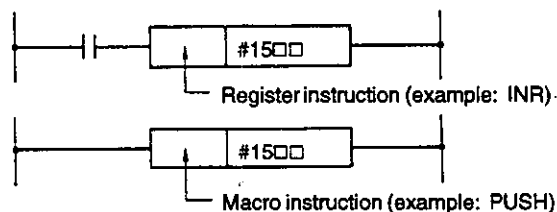
- ① One address number corresponds to a one-byte register.

Example of I/O list:



- ② The number of usable registers is in the range from #1200 to #2994 (excludes #1300 to #1399 and #1700 to #1799).
- ③ For the registers, the address number itself is used as the symbol in a ladder.

See the examples below.



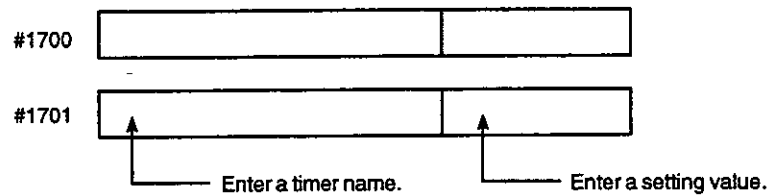
- ④ The addresses used for internal relays cannot be used for registers.

(7) Addresses of Timers (#1300 to #1399 and #1700 to #1799)

For the timers, addresses #1300 to #1399 and #1700 to #1799 are assigned.

- ① One address number corresponds to a timer.

Example of I/O list:



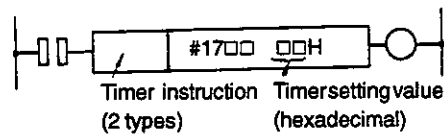
- ② The number of available timers and timer setting units are indicated below.

Table 5.1

Address No.	Timer Type	Number of Timers
#1700 to #1709, #1760 to #1769 #1300 to #1309, #1360 to #1369	1 = 8 msec	40
#1710 to #1729, #1790 to #1799 #1310 to #1329, #1390 to #1399	1 = 0.1 sec	60
#1730 to #1749, #1780 to #1789 #1330 to #1349, #1380 to #1389	1 = 50 msec	60
#1750 to #1759 #1350 to #1359	1 = 1 sec	20
#1770 to #1773 #1370 to #1373	1 = 1 min	8

- ③ An example of a timer symbol is indicated below.

(Example)



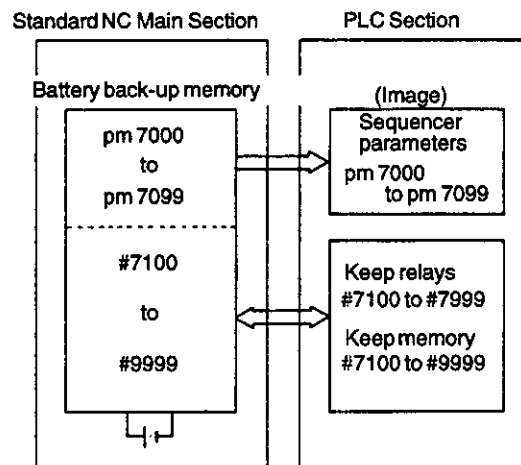
(8) Battery Back-up Memory (pm7000 to pm7099, #7100 to #9999)

For the memory to which addresses of #7000s are assigned, such memory is called the "battery back-up memory". The data saved to this type of memory are therefore retained even when the power is turned OFF.

In the PLC sequence ladder program, only the image data at the PLC side can be handled. It is not possible to handle (read and write) the source data in the NC main section.

The battery back-up memory data are classified into the following three types.

- Sequencer parameters: pm7000 to pm7099
- Keep relays: #7100 to #7999
- Keep memory: #7100 to #9999



(a) Transferring the sequencer parameter data to the PLC

The sequencer parameter data are transferred from the NC main section to the PLC in the following case in addition to the time when the power is turned ON.

If even one item of sequencer parameter data is changed by parameter write operation, the entire sequencer parameter data are transferred collectively.

In a sequence program, it is allowed only to read the sequencer parameter data. Do not attempt to change the data.

(b) Transferring the data in the keep relay and keep memory data to the NC

Image data saved in the keep relays and the keep memory in the PLC change continuously since the data are read and written as the sequence program is executed. Therefore, it is necessary to transfer the latest image data in the PLC to the battery back-up memory in the NC main section as the source data. This data transfer is called the automatic data transfer.

While the power is ON, the data in #7100 to #9999 are collectively transferred from the PLC to the NC.

(9) Addresses of Sequencer Parameters (pm7000 to pm7099)

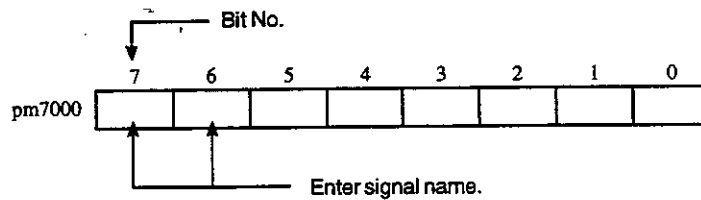
For the sequencer parameters, addresses pm7000 to pm7099 are assigned. The data set for sequencer parameters can be changed by using the normal parameter write operation.

When using the data of these parameters in a sequence parameter, the following two methods are available.

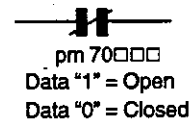
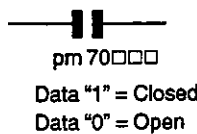
- To use as one-bit data
- To use as one-byte data

(a) Using as one-bit data

Example of I/O list:

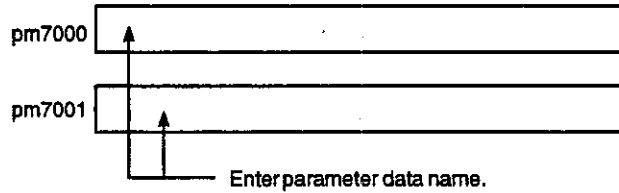


The symbol used in the ladder is indicated below.



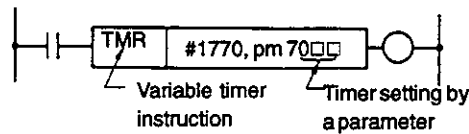
(b) Using as one-byte data

Example of I/O list:



In this case, the address number itself is used as the symbol.

See the example below where a parameter is used with a timer instruction.

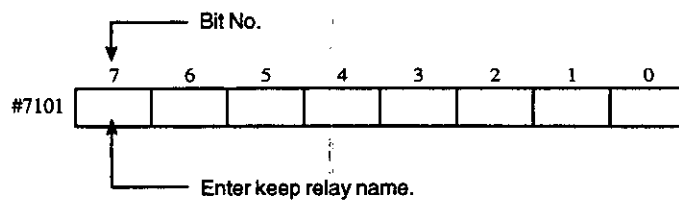


(10) Addresses of Keep Relays (#7100 to #7999)

For the keep relays that can be used in the PLC, address #7100 to #7999 are assigned.

- ① One bit corresponds to one piece of keep relay.

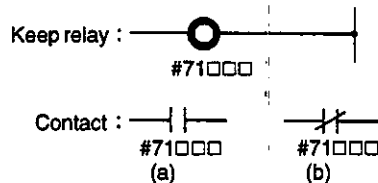
Example of I/O list:



- ② The number of usable keep relays is indicated below.

$$900 \text{ bytes} \times 8 \text{ bits} = 7200 \text{ relays}$$

- ③ Keep relays and contacts are expressed by the following symbol.



(11) Addresses of Keep Memory (#7100 to #9999)

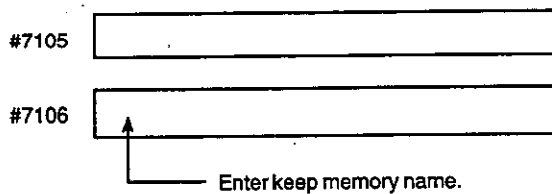
For the one-byte keep memory where the data can be retained after power OFF, addresses #7100 to #9999 are assigned. With the exception that the keep memory can retain the saved data, it can be used in the same manner as with the registers.

Therefore, the keep memory can be used as the object of register instructions or auxiliary data of macro instructions.

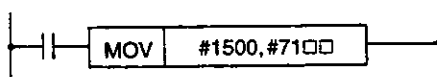
When writing a sequence program for random type ATC memory, a keep memory must be used.

- ① For one-byte (8 bits) keep memory, address number #7100 or above is assigned.

Example of I/O list:



- ② The number of usable keep memory is:
2900 in the range from #7100 to #9999
- ③ For the keep memory, the address number itself is used as the symbol.
See the example below.

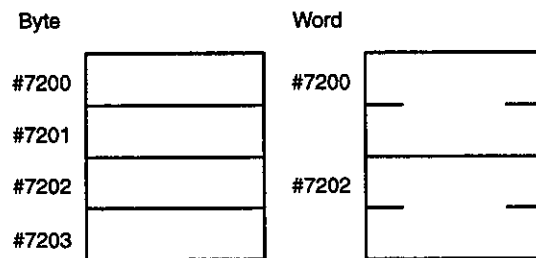


MOV: Contents of register #1500 are transferred to keep memory #71□□.

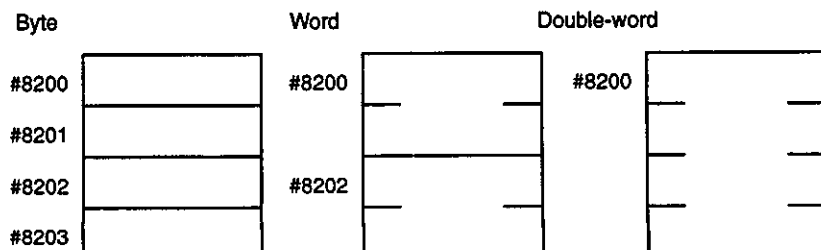
- ④ For the devices that have one-byte or larger information, a four-digit address number is assigned.

Although the data are basically of one-byte construction, the data can be handled as two-byte or four-byte data by the setting for the parameter.

In the case of two- or four-byte data, the address is specified as a one-byte unit address; the address of the least significant byte is used as the address for two- or four-byte data.



Word start number setting parameter: pm3430
 Double-word start number setting parameter: pm3431
 Setting range: 8000 to 9999
 pm3430 < pm3431



(12) Writing the Initial Values for Keep Relays and Keep Memory

When keep relays and keep memory are used in a sequence program, it is necessary to set the initial values for these devices before running the program.

Set the initial values following the procedure indicted below.

- ① Set the system number switch in the NC unit to "1" and turn the power ON.
- ② Press the [MAINT] process key and the [SEQPRM] function soft-key.
- ③ Move the cursor to the required keep memory number.

Use the page keys and the cursor up/down keys, or press the cursor up/down keys after keying in the keep memory number.

- ④ Press the [INSRT] key or the cursor right key.

[INSRT] key: The cursor moves to the decimal number column.

Cursor right key: The cursor moves to the bit data column.

- ⑤ Move the cursor to the bit for which the data should be changed and press the [WRITE] key.

Each time the [WRITE] key is pressed, ON/OFF status of the specified bit is changed.

Data entry by numeric keys is possible only when the cursor is moved to the decimal number column.

Example: Writing a decimal number for bit data

Key Operation	Bit 7 6 5 4 3 2 1 0	Decimal Number
[0] [WRITE]	0 0 0 0 0 0 0 0	0
[8] [WRITE]	0 0 0 0 1 0 0 0	8
[2] [5] [5] [WRITE]	1 1 1 1 1 1 1 1	255

PARAMETER MNT O***** N00000

#7100 7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 8

1 1 1 1 1 1 1 1 255

0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0

0: OFF 1: ON

MEM PARM DIAGN TIME STP LSK

PARM SETTING SEQPRM P*ERR [] []

[<] [1] [2] [3] [4] [5] [>]

- ⑥ Repeat steps ④ and ⑤ to write the initial values for the necessary addresses.
- ⑦ Return the system number switch to "0".

6

PLC INSTRUCTIONS

Chapter 6 describes the PLC instructions. The PLC can use 61 types of basic instructions and 11 types of macro instructions. Explanation is given for the functions and display symbols. The list of coded instructions is also given.

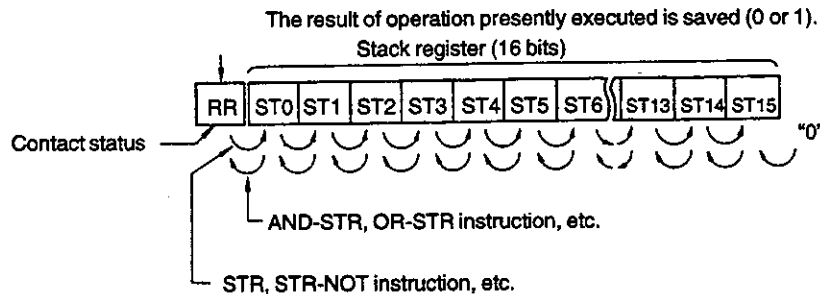
6.1	BASICS OF PLC INSTRUCTIONS	6 - 2
6.2	TYPES AND LIST OF INSTRUCTIONS	6 - 3
6.3	RELAY INSTRUCTIONS	6 - 8
6.4	TIMER INSTRUCTIONS	6 - 15
6.5	REGISTER INSTRUCTIONS	6 - 17
6.6	CONTROL INSTRUCTIONS	6 - 41
6.7	MACRO INSTRUCTIONS	6 - 44



6.1 BASICS OF PLC INSTRUCTIONS

(13) Registers Saving the Intermediate Results during Logical Operation

The PLC has registers where the intermediate results of logical operation in a sequence program are saved. Configuration of these registers is "1 bit + 16 bits".



(14) Result Register (RR)

This is a one-bit register where the result of presently executed operation is set.

Setting the status (0 or 1) of contact to RR by using the LD instruction and outputting the contents of RR to the relay address by using the OUT instruction are possible.

It is also possible to shift the contents of RR to the stack register by one bit or to shift the contents of the stack register to RR by one bit after the completion of operation by using the STR or AND-STR instruction.

(15) Stack register (ST0 to ST15)

When executing a long logical operation, it is possible to save the intermediate result to the stack register by up to 16 bits.

The STR and STR-NOT instructions move the data in RR to ST0 and, sequentially, the data in the stack registers to the right by one bit.

The AND-STR and OR-STR instructions execute the operation between the data in ST0 and RR, set the result to RR and shift the data in stack register to the left by one bit. After the execution of these instructions, "0" is set for ST15. Both the number of STR and STR-NOT instructions, and the number of AND-STR and OR-STR instructions must be the same during the execution of a series of logical operations. Otherwise, an error occurs. In other words, the number of data saving times to the stack register and the number of data fetching times from the stack register must be the same.

6.2 TYPES AND LIST OF INSTRUCTIONS

(1) Types of Instructions

With the PLC, the following types of instructions are provided.

(a) Basic instructions

Relay instruction:	13 types	} Total 61 types
Register instruction:	37 types	
Timer instruction:	2 types	
Control instruction:	9 types	

(b) Macro instructions

Macro instruction:	22 types
Auxiliary instruction:	5 types

(2) List of Relay Instructions

The list of relay instructions is indicated below.

No.	Instruction	Description	RR after Operation
1	LD	Reads the signal status (0, 1) and sets it to RR.	↕
2	LD-NOT	Reads the inversion of signal status (0, 1) and sets it to RR.	↕
3	AND	Executes AND between the contact and RR, and sets the result to RR. (Logical product)	↕
4	AND-NOT	Executes AND between the inversion of signal status and RR, and sets the result to RR. (Reverse logical product)	↕
5	OR	Executes OR between the contact and RR, and sets the result to RR. (Logical sum)	↕
6	OR-NOT	Executes OR between the inversion of signal status and RR, and sets the result to RR. (Reverse logical sum)	↕
7	XOR	Sets "not-coincide" between the signal and RR to RR.	↕
8	XNR	Sets "coincide" between the signal and RR to RR.	↕
9	STR	Enters the content of RR to the stack register and executes the LD instruction.	↕
10	STR-NOT	Enters the content of RR to the stack register and executes the LD-NOT instruction.	↕
11	AND-STR	Executes AND between RR and the stack register and sets the result to RR.	↕
12	OR-STR	Executes OR between RR and the stack register and sets the result to RR.	↕
13	OUT	Writes the result of operation (RR) to the relay (address).	—

Note: Symbol in the column of "RR after Operation"

↕ The content of RR changes before and after the operation of an instruction.

— The content of RR does not change before or after the operation of an instruction.

(3) List of Timer Instructions

The list of timer instructions is indicated below.

No.	Instruction	Description	RR after Operation
1	TIM	Timer processing (fixed timer)	time up = 1
2	TMR	Timer processing (variable timer)	time up = 1

(4) List of Register Instructions

The list of register instructions is indicated below.

No.	Instruction	Description	RR after Operation
1	INR	Adds "+1" to the register content.	—
2	DCR	Adds "-1" to the register content.	—
3	CLR	Clears the content of the register.	—
4	CMR	Inverts the content of the register.	—
5	ADI	Adds a numeric value to the content of the register.	—
6	SBI	Subtracts a numeric value from the content of the register.	—
7	ANI	AND operation between a numeric value and the content of the register.	—
8	ORI	OR operation between a numeric value and the content of the register.	—
9	XRI	XOR operation between a numeric value and the content of the register.	—
10	DEC	Coincidence between a numeric value and the content of the register	—
11	COI	Coincidence between a numeric value and the content of the register	—
12	CMP	Compares a numeric value to the content of the register.	—
13	CPI	Compares a numeric value to the content of the register.	—
14	MVI	Loads a numeric value to the register.	—
15	ADD	Executes addition between register R1 and register R2 and stores the result to R2.	—
16	SUB	Executes subtraction between register R1 and register R2 and stores the result to R2.	—
17	ANR	Executes AND between register R1 and register R2 and stores the result to R2.	—
18	ORR	Executes OR between register R1 and register R2 and stores the result to R2.	—

No.	Instruction	Description	RR after Operation
19	XRR	Executes XOR between registers R1 and R2 and stores the result to R2.	—
20	CPR	Executes comparison between registers R1 and R2 and stores the result to RR.	↕
21	COR	Executes comparison between registers R1 and R2 and stores the “coincide” result to RR.	↕
22	MOV	Transfers the content of the register R1 to register R2.	—
23	DST	Executes AND between a numeric value and the content of the register 1 and transfers the result to register R2.	—
24	DIN	Extracts data.	—
25	ADC	Executes double-length addition.	↕
26	ADDW	Executes addition between double-length register (WR2) and double-length register (WR1) and stores the result to WR2.	—
27	SUBW	Subtracts the content of double-length register (WR1) from the content of double-length register (WR2) and stores the result to WR2.	—
28	MULW	Multiplies the content of double-length register (WR1) and the content of double-length register (WR2) and stores the result to WR2.	RR is set to “1” when overflow occurs.
29	DIVW	Divides content of double-length register (WR1) by the content of double-length register (WR2) and stores the result to WR2.	0
30	INRW	Adds “+1” to the content of the double-length register.	—
31	DCRW	Adds “-1” to the content of the double-length register.	—
32	CLRW	Clears the content of the double-length register to “0”.	—
33	CMRW	Inverts the content of the double-length register.	—
34	CORW	Executes comparison between double-length register R1 and double-length register R2 and stores the “coincide” result to RR.	↕
35	CPRW	Executes comparison between double-length register R1 and double-length register R2 and stores the result to RR.	↕
36	MVIW	Loads a numeric value to the double-length register.	—
37	DSTW	Executes AND between the content of double-length register (WR1) and a numeric value and transfers the result to double-length register (WR2).	—

(5) List of Control Instructions

The list of control instructions is indicated below.

No.	Instruction	Description	RR after Operation
1	NOP	No operation	—
2	MCR	Start of master control relay	—
3	END	End of master control relay	—
4	RET	End of sequence program	—
5	RTI	Executes the RET instruction if "RR = 1".	—
6	SET	Sets "1" to RR.	1
7	RTH	End of high-speed processing sequence program	—
8	JMP	Executes jump to the location indicated by ADR.	—
9	ADR	Indicates the location of destination of jump indicated by JMP.	—

(6) List of Macro Instructions

The list of macro instructions is indicated below.

No.	Instruction	Description	RR after Operation
1	SUBP003	Detects the rising edge of the signal.	⇕
2	SUBP004	Detects the falling edge of the signal.	⇕
3	SUBP005	Counter	⇕
4	SUBP006	Rotation (for the control of rotating object)	⇕
5	SUBP007	Code conversion	⇕
6	SUBP009	Pattern clear	⇕
7	SUBP011	Parity check	⇕
8	SUBP014	Data conversion (binary ⇔ BCD)	⇕
9	SUBP017	Data search	⇕
10	SUBP018	Index data transfer	⇕
11	SUBP023	Message display (option)	⇕
12	SUBP025	Binary decode processing	⇕
13	SUBP027	Binary code conversion	⇕
14	SUBP031	Expansion data transfer	⇕
15	SUBP032	Binary conversion	⇕
16	SUBP034	Binary data search	⇕
17	SUBP035	Binary index modifier data transfer	⇕

No.	Instruction	Description	RR after Operation
18	SUBP036	Binary addition	⇕
19	SUBP037	Binary subtraction	⇕
20	SUBP038	Binary multiplication	⇕
21	SUBP039	Binary division	⇕
22	SUBP040	Binary constant definition	⇕

(7) List of Auxiliary Instructions

The list of auxiliary instructions is indicated below.

No.	Instruction	Description	RR after Operation
1	IPSH	Designation of a numeric value used by SUBP instruction	—
2	APSH	Designation of the address of a register used by SUBP instruction	—
3	PUSH	Designation of the address of a register used by SUBP instruction	—
4	TPSH	Designation of the table number of a PC table used by SUBP instruction	—
5	IPSHD	Designation of the data used by SUBP instruction	—



6.3 RELAY INSTRUCTIONS

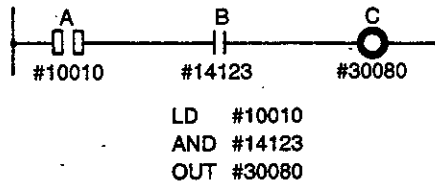
The relay instructions are described below.

(1) LD (Load) RR after operation: RR ⇅

① Format LD #XXXXXX
 ↑
 Internal signal name Example: #10100
 #14312

② Reads the signal status (1 or 0) and sets it to RR.

③ Normally, the instruction is used for NO contact.

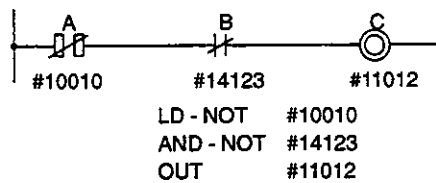


(2) LD-NOT (Load Not) RR after operation: RR ⇅

① Format LD-NOT #XXXXXX
 ↑
 Internal signal name Example: #10100
 #14312

② Reads the signal status (1 or 0) and sets it to RR.

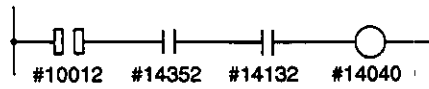
③ Normally, the instruction is used for NC contact.



(3) AND RR after operation: RR ↓

① Format AND #XXXXXX
 Internal signal name

② Executes AND between the contact and RR and sets the result to RR (logical product).

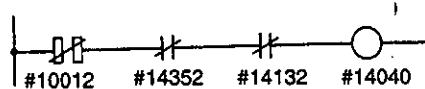


```
LD #10012
AND #14352
AND #14132
OUT #14040
```

(4) AND-NOT RR after operation: RR ↓

① Format AND-NOT #XXXXXX
 Internal signal name

② Executes AND between the inverted contact and RR and sets the result to RR (inverted logical product).

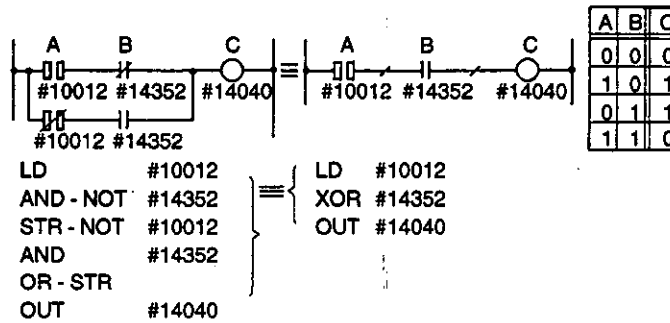


```
LD - NOT #10012
AND - NOT #14352
AND - NOT #14132
OUT #14040
```


(7) XOR (Exclusive OR) RR after operation: RR ⇅

① Format XOR #XXXXXX
 ↑
 Internal signal name

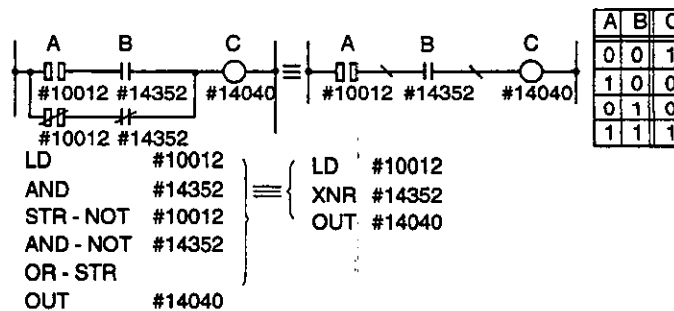
② Sets "not agree" between contact and RR to RR.



(8) XNR (Exclusive NR) RR after operation: RR ⇅

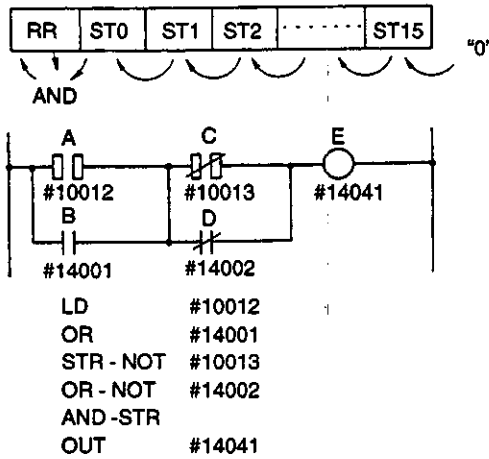
① Format XNR #XXXXXX
 ↑
 Internal signal name

② Sets "agree" between contact and RR to BR.



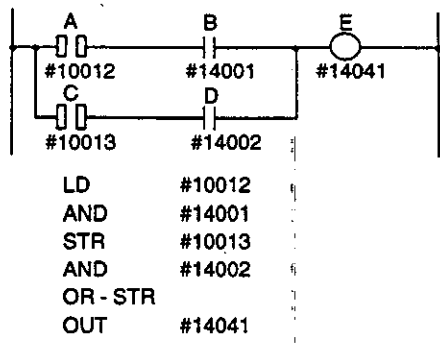
(11) AND-STR (AND Store) RR after operation: RR ↓

- ① Format AND-STR
- ② Executes AND between RR and stack (ST0) and sets the result to RR. Stacks shift to the left by one.



(12) OR-STR (OR Store) RR after operation: RR ↓

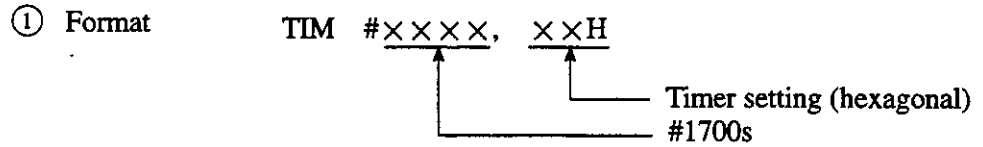
- ① Format OR-STR
- ② Executes OR between RR and stack (ST0) and sets the result to RR. Stacks shift to the left by one.



6.4 TIMER INSTRUCTIONS

The timer instructions are described below.

(1) TIM (Fixed Timer) RR = 1 at time-up



- ② The timer instruction counts the length of time while the ST contact is ON (RR = 1), and turns the TM ON at the preset time.

While the ST contact is OFF (RR = 0), the instruction sets the TM OFF and resets the timer.

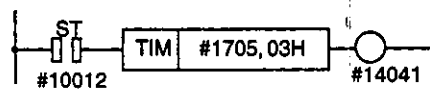
- ③ Setting range is from 0 to 255 in decimal. However, the setting must be made in hexadecimal.

If the setting is "255", the timer does not count up.

- ④ The following five types of timers can be used.

Address No.	Timer Type	Number of Timers
#1700 to #1709, #1760 to #1769 #1300 to #1309, #1360 to #1369	1 = 8 msec	40
#1710 to #1729, #1790 to #1799 #1310 to #1329, #1390 to #1399	1 = 0.1 sec	60
#1730 to #1749, #1780 to #1789 #1330 to #1349, #1380 to #1389	1 = 50 msec	60
#1750 to #1759 #1350 to #1359	1 = 1 sec	20
#1770 to #1773 #1370 to #1373	1 = 1 min	8

- Accuracy of timers depends on the basic unit value.
With a timer of #1770, for example, setting of "2" sets the count-up time in the range of 61 to 120 seconds since the setting unit of this timer is "1 = 1 minute".
- To use the timers of #1300 to #1399, the compiler of Ver. 3.5 or higher is necessary.



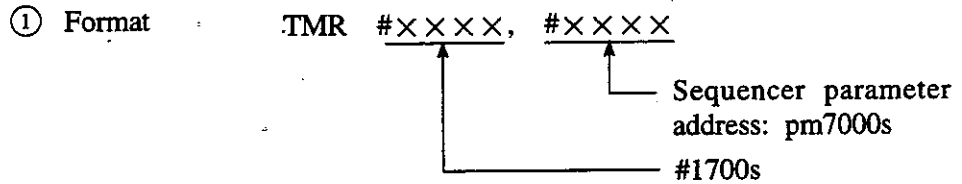
```
LD #10012
TIM #1705,03H
OUT #14041
```



Do not use the same address for both a fixed timer and a variable timer. If used, correct operation cannot be guaranteed.

(2) TMR (Variable Timer)

RR = 1 at time-up



- ② The timer instruction counts the length of time while the ST contact is ON (RR = 1), and turns the TM ON at the preset time.

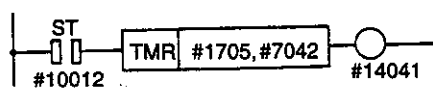
While the ST contact is OFF (RR = 0), the instruction sets the TM OFF and resets the timer.

- ③ Setting range is from 0 to 255 in decimal. However, the setting must be made in hexadecimal.

If the setting is "255", the timer does not count up.

- ④ Write the timer value from the NC keyboard by following the parameter writing procedure. In this case, the timer value can be written in a decimal value.

- ⑤ Five types of timers can be used as with the TIM instruction.



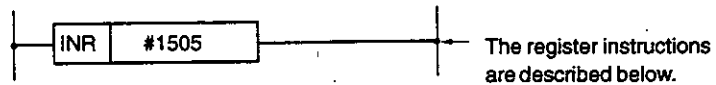
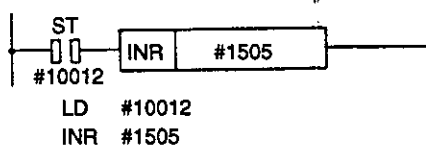
LD #10012
TMR #1705, #7042
OUT #14041

6.5 REGISTER INSTRUCTIONS

The register instructions are described below.

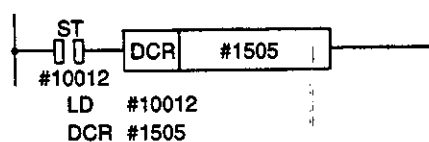
(1) INR (Increment Register) RR after operation: RR —

- ① Format INR #××××
- ② The instruction adds "+1" to the content of the register when the ST contact is ON (RR = 1). If the ST contact is OFF (RR = 0), addition is not executed.
- ③ An ST contact must be entered before the INR instruction.
- ④ The instruction adds "+1" to the content of the register in intervals of "4 × n" msec while the ST contact is ON.
- ⑤ There is no function to detect overflow. If a timer counts to FFH, it returns to 0H.
- ⑥ The following examples show when the register of #1500s is used.



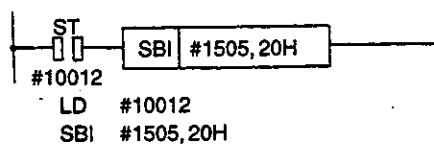
(2) DCR (Decrement Register) RR after operation: RR —

- ① Format DCR #×××××
- ↑
Register
- ② The instruction adds "-1" to the content of the register when the ST contact is ON (RR = 1). If the ST contact is OFF (RR = 0), addition is not executed.
- ③ An ST contact must be entered before the DCR instruction.
- ④ The instruction adds "-1" to the content of the register in intervals of "4 × n" msec while the ST contact is ON.



- ⑤ There is no function to detect underflow. If a timer counts to 0H, it returns to FFH.

- ④ The SBI instruction is executed in intervals of “4 × n” msec while the ST contact is ON.



- ⑤ There is no function to detect underflow. Make sure that “numeric value contents of register”.

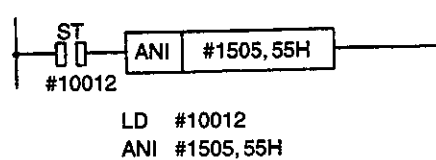
(7) ANI (AND Immediate) RR after operation: RR —

- ① Format
- ANI #XXXX, XXH
-

- ② The instruction executes AND between the specified numeric value and the content of the register and stores the result to the register when the ST contact is ON (RR = 1). If the ST contact is OFF (RR = 0), the instruction is not executed.

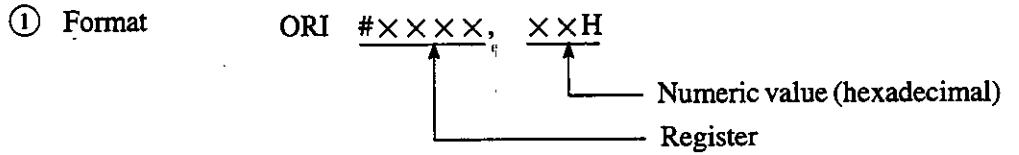
The RR content remains unchanged before and after the execution of the ANI instruction.

- ③ An ST contact must be entered before the ANI instruction.
- ④ The ANI instruction is executed in intervals of “4 × n” msec while the ST contact is ON.



	D7	D6	D5	D4	D3	D2	D1	D0
Register	0	0	1	1	0	0	1	1
Numeric Value	0	1	0	1	0	1	0	1
Result	0	0	0	1	0	0	0	1

(8) ORI (OR Immediate) RR after operation: RR —

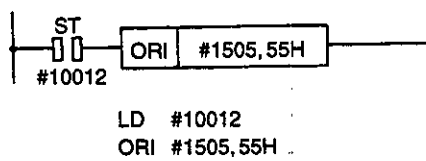


② The instruction executes OR between the specified numeric value and the content of the register and stores the result to the register when the ST contact is ON (RR = 1). If the ST contact is OFF (RR = 0), the instruction is not executed.

The RR content remains unchanged before and after the execution of the ORI instruction.

③ An ST contact must be entered before the ORI instruction.

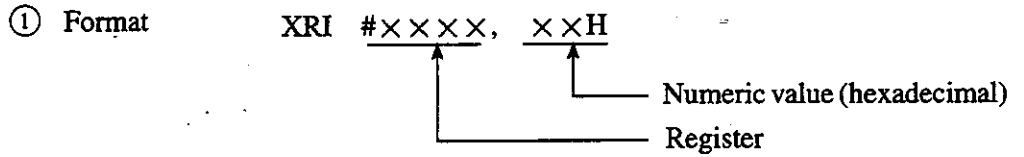
④ The ORI instruction is executed in intervals of “4 × n” msec while the ST contact is ON.



	D7	D6	D5	D4	D3	D2	D1	D0
Register	0	0	1	1	0	0	1	1
Numeric Value	0	1	0	1	0	1	0	1
Result	0	1	1	1	0	1	1	1



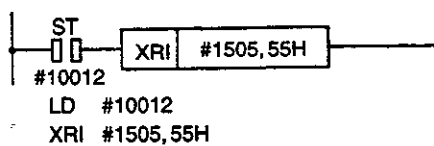
(9) XRI (XOR Immediate)



- ② The instruction executes XOR between the specified numeric value and the content of the register and stores the result to the register when the ST contact is ON (RR = 1). If the ST contact is OFF (RR = 0), the instruction is not executed.

The RR content remains unchanged before and after the execution of the XRI instruction.

- ③ An ST contact must be entered before the XRI instruction.
- ④ The XRI instruction is executed in intervals of "4 × n" msec while the ST contact is ON.

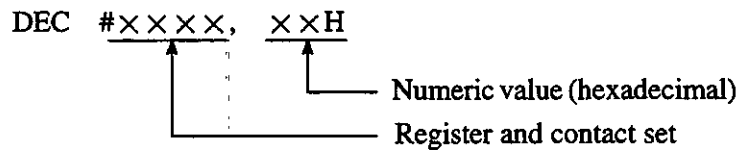


	D7	D6	D5	D4	D3	D2	D1	D0
Register	0	0	1	1	0	0	1	1
Numeric Value	0	1	0	1	0	1	0	1
Result	0	1	1	0	0	1	1	0

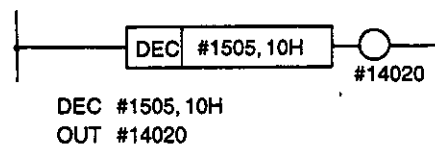
(10) DEC (Decode)

RR after operation: RR \updownarrow

① Format



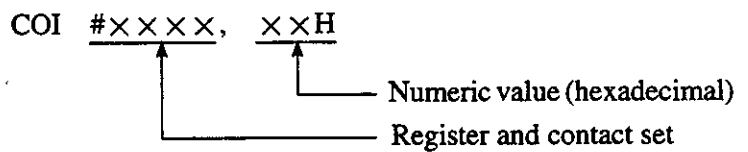
- ② The instruction compares the numeric value to the eight-bit data of the register or contact set and sets "1" to RR (RR = 1) if the result of comparison is "coincide". This instruction is executed independent of RR at the input side.
- ③ It is not allowed to enter a contact before the DEC instruction. If a contact must be entered, use the COI instruction.
- ④ The DEC instruction is executed in intervals of "4 × n" msec.



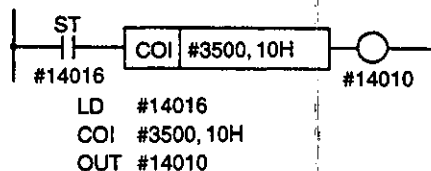
(11) COI (Coincide Immediate)

RR after operation: RR \updownarrow

① Format

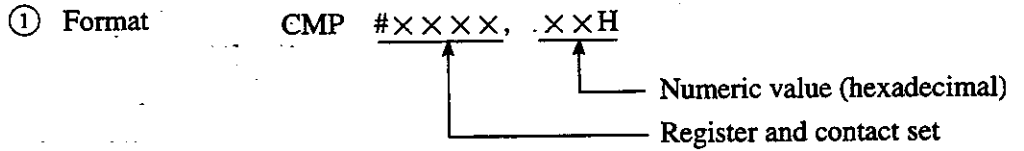


- ② When the ST contact is ON (RR = 1), the instruction compares the numeric value to the eight-bit data of the register or contact set and sets "1" to RR (RR = 1) if the result of comparison is "coincide".
If the ST contact is OFF (RR = 0), the instruction is not executed. RR remains unchanged.
- ③ An ST contact must be entered before the COI instruction.
- ④ The COI instruction is executed in intervals of "4 × n" msec while the ST contact is ON.



(12) CMP (Compare)

RR after operation: RR ↓



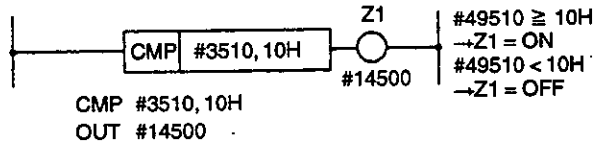
- ② The instruction compares the numeric value to the eight-bit data of the register or contact set and sets "1" or "0" depending on the result of comparison.

Register (contact) \geq Numeric value: RR = 1

Register (contact) < Numeric value: RR = 0

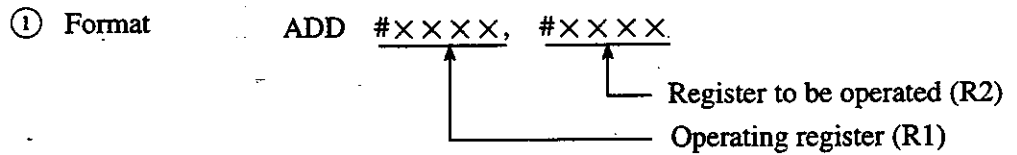
This instruction is executed independent of RR at the input side.

- ③ It is not allowed to enter a contact before the CMP instruction. If a contact must be entered, use the CPI instruction.
- ④ The CMP instruction is executed in intervals of "4 × n" msec.



(15) ADD (Add Register)

RR after operation: RR —

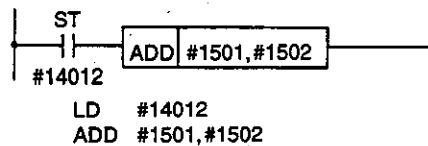


- ② The function executes addition between the content in the register R2 and the content in register R1 when the SR contact is ON (RR = 1) and stores the result to register R2.

The content in register R1 remains unchanged and the status of RR also remains unchanged.

If the ST contact is OFF (RR = 0), the instruction is not executed.

- ③ An ST contact must be entered before the ADD instruction.
- ④ The ADD instruction is executed in intervals of "4 × n" msec while the ST contact is ON.



- ⑤ There is no function to detect overflow. Make sure that the result will not exceed 255 (FFH).

(16) SUB (Sub Register)

RR after operation: RR —

- ① The SUB instruction is basically the same as the ADD instruction with the exception that the SUB instruction executes subtraction.

$$(R2 - R1 \rightarrow R2)$$

- ② There is no function to detect underflow. Make sure that the following is always satisfied: $R1 \leq R2$

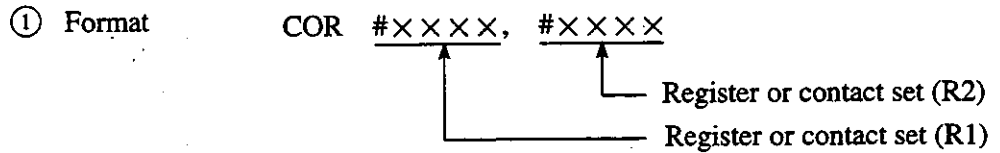
(17) ANR (AND Register)

RR after operation: RR —

- ① The ANR instruction is basically the same as the ADD instruction with the exception that the ANR instruction executes AND operation.

$$(R2 \text{ AND } R1 \rightarrow R2)$$

(21) COR (Coincide Register) RR after operation: RR \updownarrow

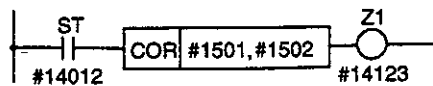


- ② The instruction executes comparison between R1 and R2 when the ST contact is ON (RR = 1), and sets "0" or "1" to Z1 according to the result of comparison.

R1 = R2 Z1 = 1
 R1 \neq R2 Z1 = 0

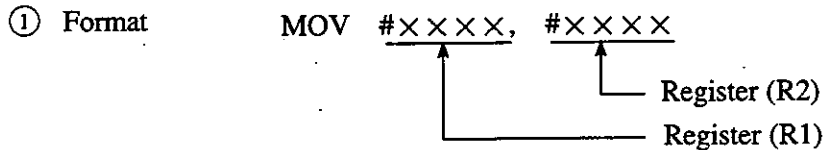
If the ST contact is OFF (RR = 0), the COR instruction is not executed. The content of RR remains unchanged.

- ③ After the execution of the COR instruction, the contents of R1 and R2 remain unchanged.
- ④ An ST contact must be entered before the COR instruction.
- ⑤ The COR instruction is executed in intervals of "4 \times n" msec while the ST contact is ON.



LD #14012 #1501 = #1502 ... Z1 is set.
 COR #1501, #1502 #1501 \neq #1502 ... Z1 is cleared.
 OUT #14123

(22) MOV (Move Register) RR after operation: RR —



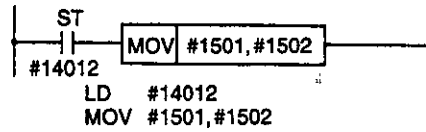
- ② The function transfers the content of the register R1 to register R2 when the ST contact is ON (RR = 1).

The content of the register R1 remains unchanged before and after the execution of the instruction.

If the ST contact is OFF (RR = 0), the MOV instruction is not executed.

- ③ An ST contact must be entered before the MOV instruction.

- ④ The MOV instruction is executed in intervals of "4 × n" msec while the ST contact is ON.



(23) DST (Data Store)

RR after operation: RR —

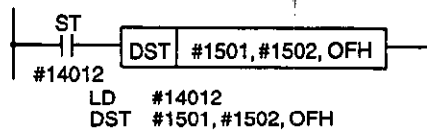
- ① Format

DST	#XXXX,	#XXXX,	XXH	
	↑	↑	↑	
				Numeric value (hexadecimal)
				Register (R2)
				Register (R1)

- ② The instruction executes AND between the content of the register R1 and the numeric value when the ST contact is ON (RR = 1), and stores the result to register R2.

The content of the register R1 remains unchanged before and after the execution of the instruction.

If the ST contact is OFF (RR = 0), the DST instruction is not executed.



	D7	D6	D5	D4	D3	D2	D1	D0
Register R1	B	B	B	B	B	B	B	B
Numeric Value	0	0	0	0	1	1	1	1
Register R2	0	0	0	0	B	B	B	B

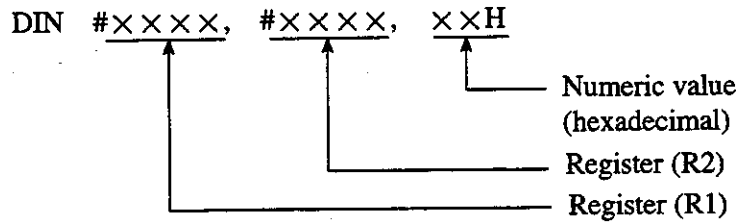
B: "0" or "1"

- ③ An ST contact must be entered before the DST instruction.
- ④ The DST instruction is executed in intervals of "4 × n" msec while the ST contact is ON.

(24) DIN (Data Insert)

RR after operation: RR —

① Format

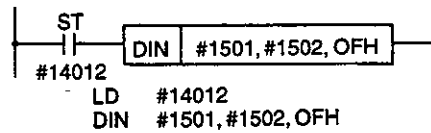


② The instruction executes AND between R1 and the numeric value, and between R2 and the complement of the numeric value, then OR between the results when the ST contact is ON (RR = 1) and stores the result of OR to R2. (Extraction of the data)

If the ST contact is OFF (RR = 0), the DIN instruction is not executed.

③ An ST contact must be entered before the DIN instruction.

④ The DIN instruction is executed in intervals of "4 × n" msec while the ST contact is ON.



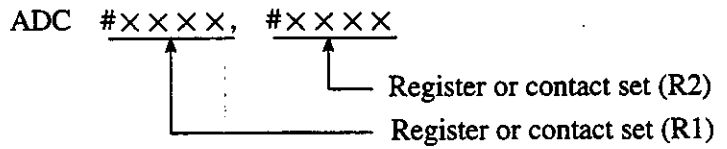
	D7	D6	D5	D4	D3	D2	D1	D0
R1	A	A	A	A	A	A	A	A
R2	B	B	B	B	B	B	B	B
n	0	0	0	0	1	1	1	1
Result	B	B	B	B	A	A	A	A

A, B: "0" or "1"

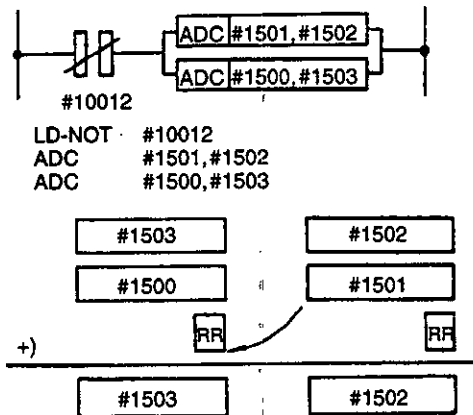
(25) ADC (Add with Carry)

RR after operation: RR \updownarrow

① Format



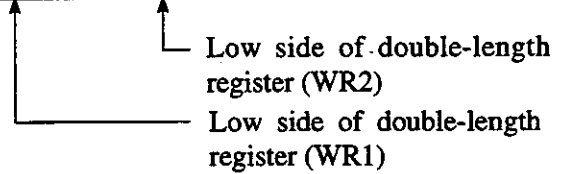
- ② The instruction executes addition between the contents in registers R1 and R2, and the content of RR and stores the result to register R2. If carry occurs "1" is set to RR.
- ③ The ADC instruction is executed in intervals of "4 × n" msec while the ST contact is ON.
- ④ To execute the ADC instruction, the content of RR must be "0".



(26) ADDW (ADD Word Register)RR after operation: RR —

① Format

ADDW #XXXX, #XXXX



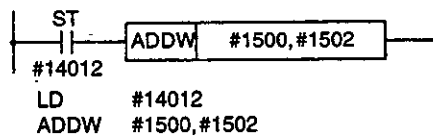
② The instruction executes addition between the contents of double-length register (WR2) and double-length register (WR1) when the ST contact is ON (RR = 1) and stores the result to double-length register (WR2).

$$(WR1) + (WR2) \rightarrow (WR2)$$

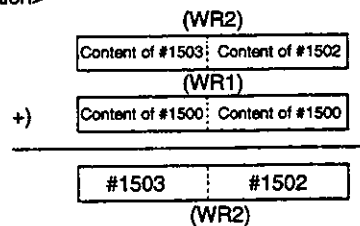
If the ST contact is OFF (RR = 0), the ADDW instruction is not executed.

③ An ST contact must be entered before the ADDW instruction.

④ The ADDW instruction is executed in intervals of "4 × n" msec while the ST contact is ON.

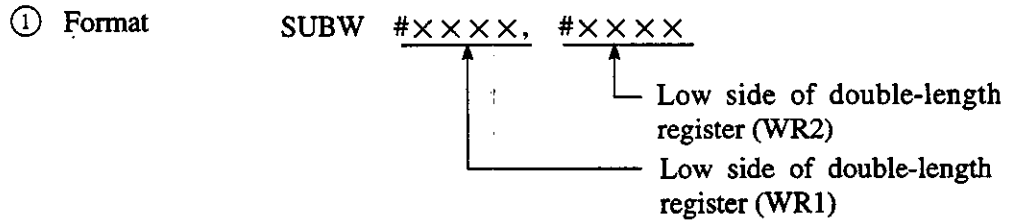


<Description>



⑤ There is no function to detect overflow. Make sure that the result will not exceed FFFFH.

(27) SUBW (SUB Word Register) RR after operation: RR —

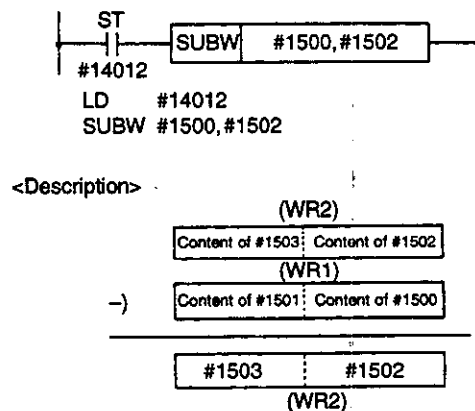


- ② The instruction executes subtraction between the contents of double-length register (WR2) and double-length register (WR1) when the ST contact is ON (RR = 1) and stores the result to double-length register (WR2).

$$(WR2) - (WR1) \rightarrow (WR2)$$

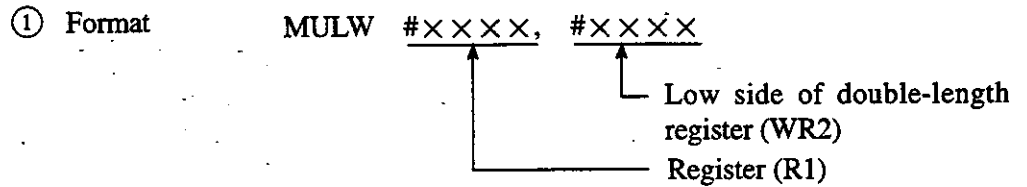
If the ST contact is OFF (RR = 0), the SUBW instruction is not executed.

- ③ An ST contact must be entered before the SUBW instruction.
- ④ The SUBW instruction is executed in intervals of "4 × n" msec while the ST contact is ON.



- ⑤ There is no function to detect underflow. Make sure that the following is always satisfied: $WR1 \leq WR2$

(28) MULW (MUL Word Register) RR after operation: RR \updownarrow



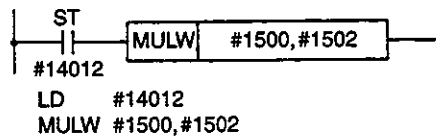
- ② The instruction executes multiplication of the contents of double-length register (WR2) and the register (R1) when the ST contact is ON (RR = 1) and stores the result to double-length register (WR2).

$$(WR1) \times (R2) \rightarrow (WR2)$$

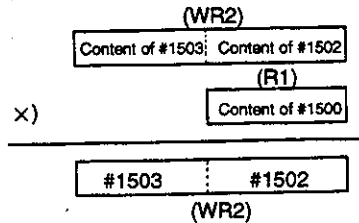
If the ST contact is OFF (RR = 0), the MULW instruction is not executed.

- ③ An ST contact must be entered before the MULW instruction.
- ④ The MULW instruction is executed in intervals of "4 × n" msec while the ST contact is ON.
- ⑤ If overflow occurs, in other words, if the result exceeds FFFH, "1" is set to RR (RR = 1).

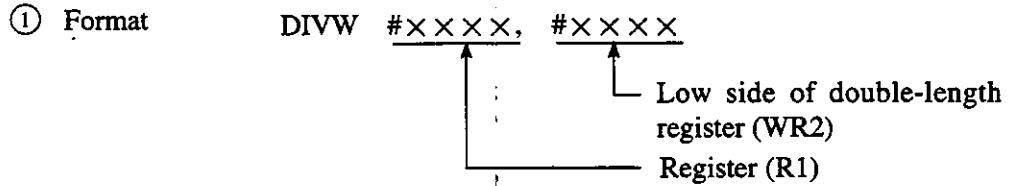
The lower one word is stored to the register.



<Description>



(29) DIVW (DIV Word Register) RR after operation: RR —



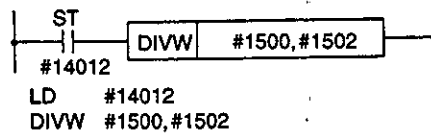
- ② The instruction executes division between the contents of double-length register (WR2) and register (R1) when the ST contact is ON (RR = 1) and stores the result to double-length register (WR2).

The content of R1 remains unchanged before and after the execution of the instruction.

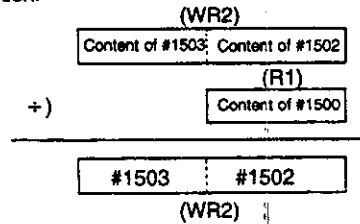
$$(WR1) \div (R2) \rightarrow (WR2)$$

If the ST contact is OFF (RR = 0), the DIVW instruction is not executed.

- ③ An ST contact must be entered before the DIVW instruction.
- ④ The DIVW instruction is executed in intervals of “4 × n” msec while the ST contact is ON.

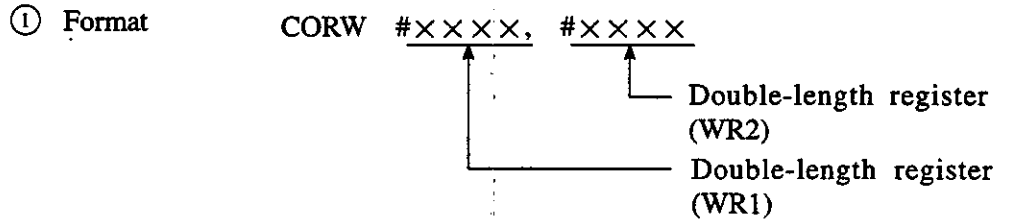


<Description>



- ⑤ The instruction is not executed if the content of R1 is “0”.

(34) CORW (Coincide Word Register) RR after operation: RR \updownarrow

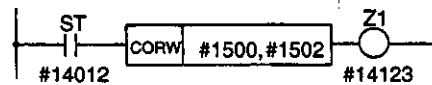


② The instruction executes comparison between WR1 and WR2 when the ST contact is ON (RR = 1), and sets "0" or "1" to Z1 according to the result of comparison.

$$\begin{aligned} \text{WR1} = \text{WR2} \quad \text{Z1} &= 1 \\ \text{WR1} \neq \text{WR2} \quad \text{Z1} &= 0 \end{aligned}$$

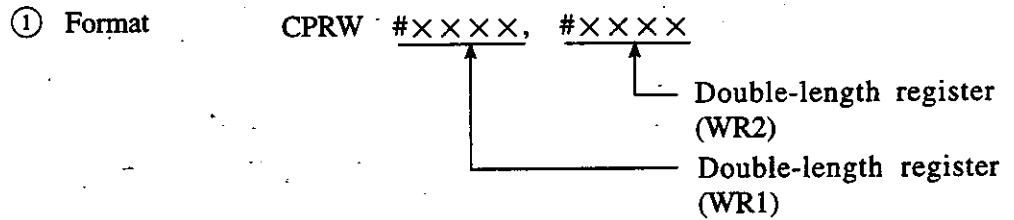
If the ST contact is OFF (RR = 0), the CORW instruction is not executed. The content of RR remains unchanged.

- ③ After the execution of the CORW instruction, the contents of WR1 and WR2 remain unchanged.
- ④ An ST contact must be entered before the CORW instruction.
- ⑤ The CORW instruction is executed in intervals of "4 × n" msec while the ST contact is ON.



LD #14012 #1500 = #1502 ... Z1 is set.
 CORW #1500, #1502 #1500 \neq #1502 ... Z1 is cleared.
 OUT #14123

(35) CPRW (Compare Word Register) RR after operation: RR \updownarrow

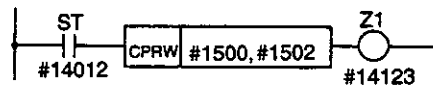


- ② The instruction executes comparison between WR1 and WR2 when the ST contact is ON (RR = 1), and sets "0" or "1" to Z1 according to the result of comparison.

$$\begin{aligned} \text{WR1} < \text{WR2} & \text{Z1} = 0 \\ \text{WR1} \geq \text{WR2} & \text{Z1} = 1 \end{aligned}$$

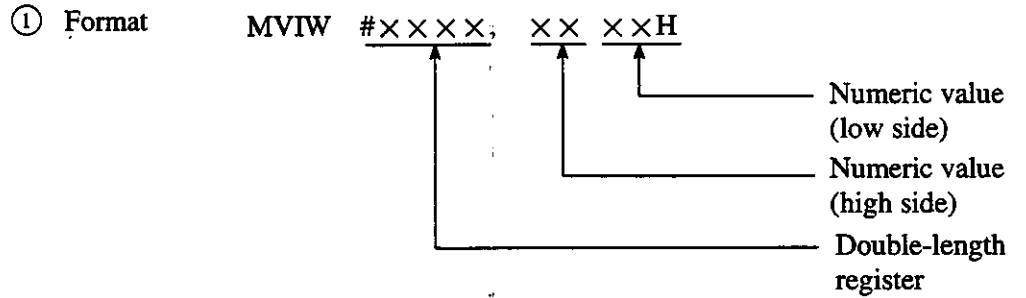
If the ST contact is OFF (RR = 0), the CPRW instruction is not executed. The content of RR remains unchanged.

- ③ After the execution of the CPR instruction, the contents of WR1 and WR2 remain unchanged.
- ④ An ST contact must be entered before the CPRW instruction.
- ⑤ The CPRW instruction is executed in intervals of "4 × n" msec while the ST contact is ON.

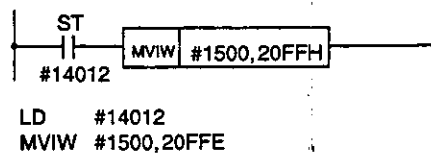


LD	#14012	#1500 < #1502 Z1 is set.
CPRW	#1500, #1502	#1500 ≥ #1502 Z1 is cleared.
OUT	#14123	

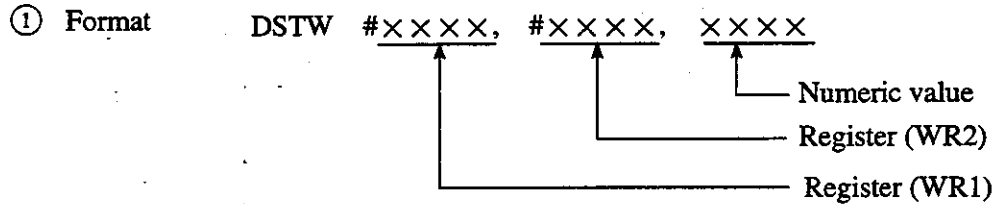
(36) MVIW (Move Immediate Word Register) RR after operation: RR —



- ② The instruction transfers the numeric value to the register when the ST contact is ON (RR = 1).
If the ST contact is OFF (RR = 0), the MVIW instruction is not executed.
- ③ An ST contact must be entered before the MVIW instruction.
- ④ The MVIW instruction is executed in intervals of "4 × n" msec while the ST contact is ON.

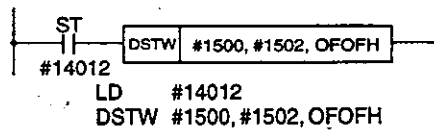


(37) DSTW (Data Store Word Register) RR after operation: RR —



② The instruction executes AND between the content of WR1 and the numeric value when the ST contact is ON (RR = 1) and stores the result to WR2. The content of the register (WR1) remains unchanged before and after the execution of the instruction.

If the ST contact is OFF (RR = 0), the DSTW instruction is not executed.



- ③ An ST contact must be entered before the DSTW instruction.
- ④ The DSTW instruction is executed in intervals of "4 × n" msec while the ST contact is ON.

	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Register WR1	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
Numeric Value	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
Register WR1	0	0	0	0	B	B	B	B	0	0	0	0	B	B	B	B

B: "0" or "1"

6.6 CONTROL INSTRUCTIONS

The control instructions are described below.

(1) NOP (No Operation) RR after operation: RR —

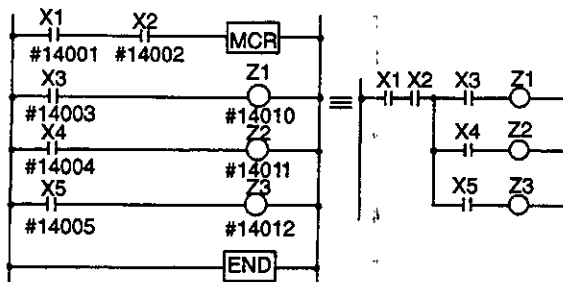
- ① Format NOR
- ② No operation is executed and the program advances to the next step.

The content of RR remains unchanged before and after the execution of the instruction.

(2) MCR (Master Control) RR after operation: RR —

- ① Format MCR
- ② The instruction executes the sequence ladder when the both X1 and X2 contacts are ON (RR = 1).

If the X1 or/and X2 contacts are OFF (RR = 0), the ladder is executed to END in the state of "RR = 0".



```
LD #14001
AND #14002
MCR
LD #14003
OUT #14010
LD #14004
OUT #14011
LD #14005
OUT #14012
END
```

<Description>

If X1 and X2 contacts are OFF, "0" is output to the internal relays Z1, Z2, and Z3.

- ③ It is possible to enter another MCR instruction between the MCR and END instructions (max. 7 levels).
- ④ When a timer instruction is included in the MCR instruction, the timer is cleared when the MCR instruction is OFF.
- ⑤ Even if the self-holding circuit is formed between the MCR and END instructions, the circuit output is OFF when the MCR instruction is OFF.

(3) END (Master Control End) RR after operation: RR —

① Format END

② The instruction indicates the end of the MCR instruction.

(4) RET (Return) RR after operation: RR —

① Format RET

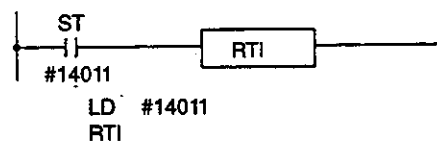
② The RET instruction indicates the end of a sequence program.

(5) RTI (Return Indirect) RR after operation: RR —

① Format RTI

② The instruction executes the RET instruction when the ST contact is ON.

If the ST contact is OFF, the ladder of the next step is executed.



(6) SET (Set Result Register) RR after operation: RR —

① Format SET

② The instruction forcibly sets "1" for "RR".

(7) RTH (Return High Sequence) RR after operation: RR —

① Format RTH

② The instruction indicates the end of a high-speed processing sequence program.

6.7 MACRO INSTRUCTIONS

There are several machine control sequences that cannot be programmed easily if only basic instructions (relay instructions, register instruction, etc.) are used. The macro instructions are provided to simplify programming such sequences:

Macro instructions are written in the following format.

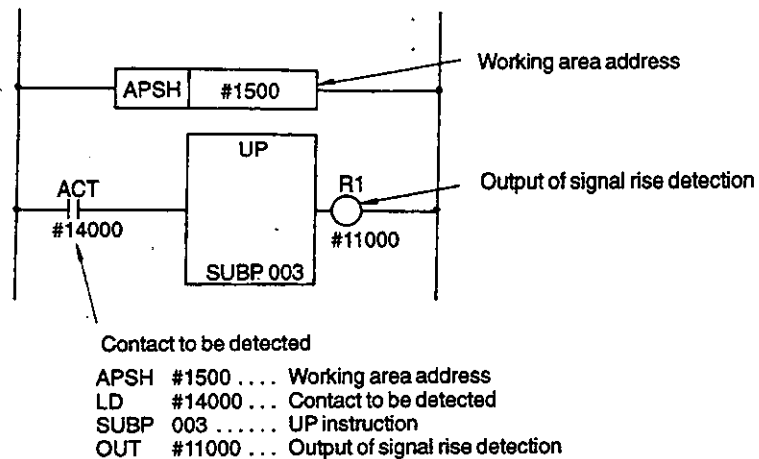
SUBP XXX
 ↑
 Macro instruction number

(1) SUBP 003 (UP: Detecting rising edge of a signal)

(a) Function

The instruction detects the rising edge of a signal.

(b) Format



(c) Control conditions

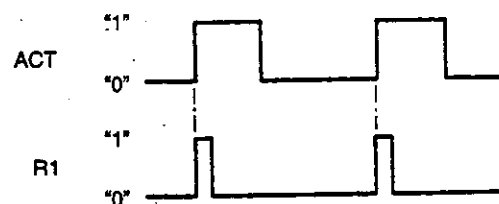
① Working area address (APSH #XXXX)

Designate an address that is not used by other instructions. Prepare one byte for one SUBP 003.

② Contact to be detected (ACT) and output of signal rise detection (R1)

ACT = 0: - Rising edge of a signal is not detected; R1 = 0

ACT = 1: - R1 value changes "0" → "1" → "0" at the detection of the rising edge.



③ If "ACT = 1" when the power is turned ON, it is regarded as the rise.

(3) SUBP 005 (COUNTER)

(a) Function

The counter can be used for the following purposes to control machine tool operation as indicated below according to the applications.

① Ring counter

The counter is a ring counter. Accordingly, the counter value returns to the initial value if a count signal is input after counting up to the preset value.

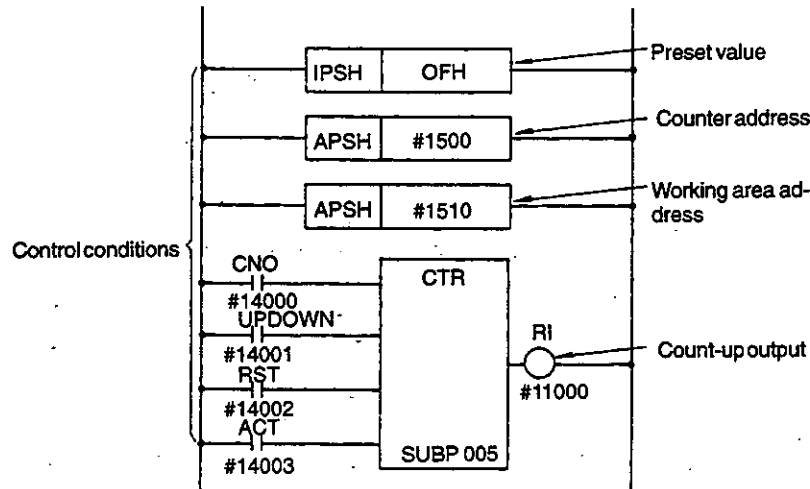
② Preset counter

The count-up signal is output when the count value reaches the preset value.

③ Up/down counter

The counter can be used for both up and down counters.

(b) Format



IPSH	OFH	Preset value
APSH	#1500	Counter address
APSH	#1510	Working area address
LD	#14000	CNO
STR	#14001	UPDOWN
STR	#14002	RST
STR	#14003	ACT
SUBP	005	Counter instruction
OUT	#11000	Count-up output

(c) Control conditions

① Designation of preset value (IPSH $\times \times$)

Designate the preset value directly.

To designate a variable value, use the PUSH instruction instead of the IPSH instruction. If the PUSH instruction is used, the contents of the designated address are used as the preset value.

Example: PUSH #1550

With the designation indicated above, two bytes of #1550 and #1551 are used. Even if only one byte is used, #1551 must not be used for other instructions.

② Designation of counter address (APSH # $\times \times \times \times$)

Designate the counter address.

If "APSH #1500" is designated, continuous two bytes (#1500 and #1501) are used for the counter address.

③ Designation of working area address (APSH # $\times \times \times \times$)

Designate an address that is not used by other instructions. One byte is necessary for one SUBP 005.

If two or more SUBP 005 instructions are used, it is necessary to designate an address for each SUBP 005 instruction.

④ Designation of initial value (CNO)

CNO = 0: Counting begins with "0". (0, 1, 2, n)

CNO = 1: Counting begins with "1". (1, 2, 3, n)

⑤ Designation of up/down counter (UPDOWN)

UPDOWN = 0: Up counter

The initial value is "0" with CNO = 0.

The initial value is "1" with CNO = 1.

UPDOWN = 1: Down counter

The initial value is the preset value.



⑥ Reset (RST)

RST = 0: Reset released

RST = 1: Reset

R1 is cleared to "0".

Counted value is reset to the initial value.

⑦ Count signal (ACT)

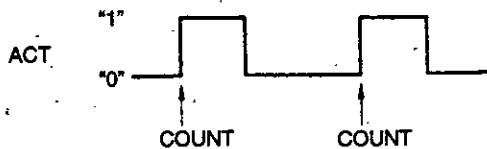
ACT = 0: The counter does not operate. Contents of R1 remain unchanged.

ACT = 1: Counts at the rising edge ("0" → "1").

If the content of the counter is greater than the preset value, the counter operates in the following manner.

UP counter: The value returns to the initial value at the first ACT signal.

DOWN counter: The value is reduced at each input of ACT until the count value is reduced to the preset value. After that the counter operates as a normal counter.



⑧ Count-up output (R1)

Up counter: "1" is set for R1 upon counting up to the preset value.

Down counter: "1" is set for R1 according to the following condition.

CNO = 0

. Upon counting down to "0"

CNO = 1

. Upon counting down to "1"



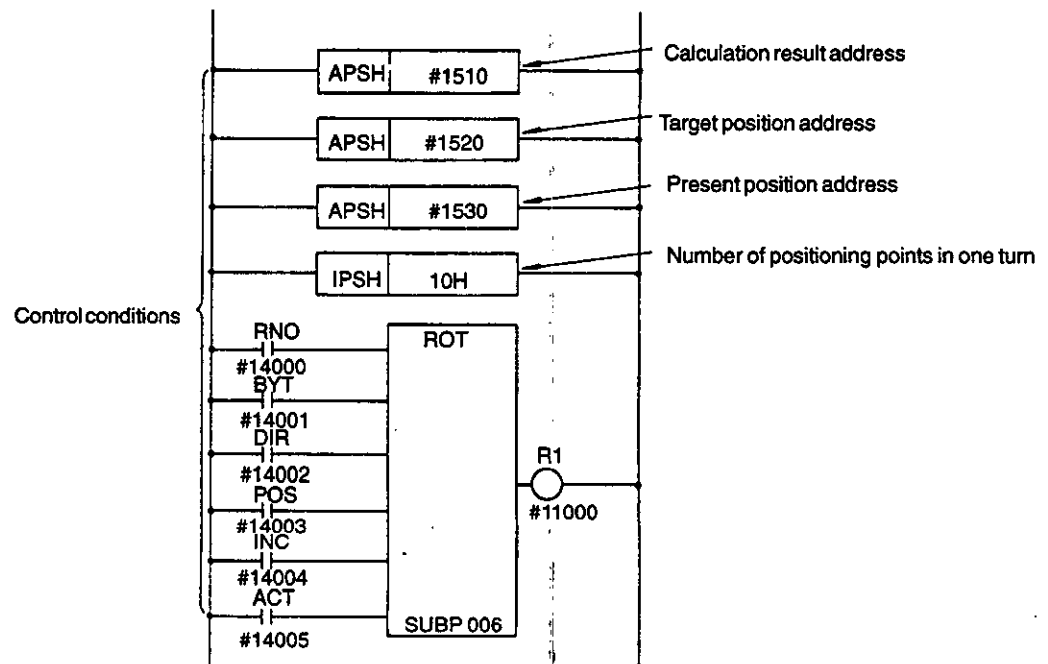
(4) SUBP 006 (ROTATION)

(a) Function

This instruction is used to control rotating units such as turrets, ATCs, and rotary tables. It has the following functions:

- Determination for shorter-path when determining the direction of rotation
- Calculation of the number of steps between the present position and the target position
- Calculation of the position one step before the target position or the number of steps to the position one step before the target position

(b) Format



APSH #1510 Calculation result address
 APSH #1520 Target position address
 APSH #1530 Present position address
 IPSH 10H Number of rotating unit positioning points
 LD #14000 ... Position number; from "0" or from "1"
 STR #14001 ... Position data; 1 byte or 2 bytes
 STR #14002 ... Direction of rotation; fixed or shorter path
 STR #14003 ... Target position or position 1 step before the target position
 STR #14004 ... Position number or the number of steps
 STR #14005 ... Execution
 SUBP 006 ROT instruction
 OUT #11000 ... Output of direction of rotation

(c) Control conditions

① Designation of calculation result storing addresses (APSH # × × × ×)

The ROT instruction calculates the number of steps the rotating unit should rotate, the number of steps of the position one step before the target position, or the position one step before the target position. The result of calculation is stored in the designated address.

② Designation of target position address (APSH # × × × ×)

Designate the address where the target position is stored: for example, the address where the T command is output from the NC.

③ Designation of the present position address (APSH # × × × ×)

Designate the address where the present position data are stored: for example, the address of the counter where the position of the rotating unit is stored.

④ Designation of the initial value of the position number of the rotating unit (RNO)

RNO = 0: Position number of the rotating unit begins with "0".
RNO = 1: Position number of the rotating unit begins with "1".

⑤ Designation of the number of bytes of the position data (BYT)

BYT = 0: Binary 1 byte
BYT = 1: Binary 2 bytes

⑥ Designation of whether or not shorter-path rotation direction is determined (DIR)

DIR = 0: Determination is not made.
Rotation direction is only the FOR direction.
DIR = 1: Determination is made.

⑦ Designation of operation conditions (POS)

POS = 0: The number of steps to the target position is calculated.
POS = 1: The position or the number of steps to the position which is one position or one step before the target position.

⑧ Designation of the position number or the number of steps (INC)

INC = 0: The position number is calculated.
INC = 1: The number of steps is calculated.

⑨ Execution command (ACT)

ACT = 0: ROT instruction is not executed.
R1 is not influenced.

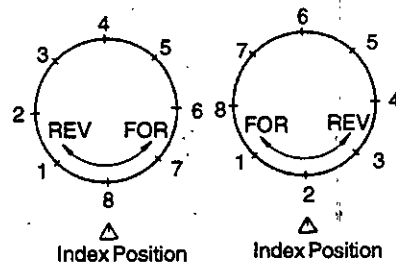
ACT = 1: ROT instruction is executed.
(Not by the rising edge of the signal.)

⑩ Output of rotation direction (R1)

R1 = 0: The rotation direction is "forward".

R1 = 1: The rotation direction is "reverse".

FOR (forward) direction	The direction in which the number increases in reference to the index position.
REV (reverse) direction	The direction in which the number decreases in reference to the index position.

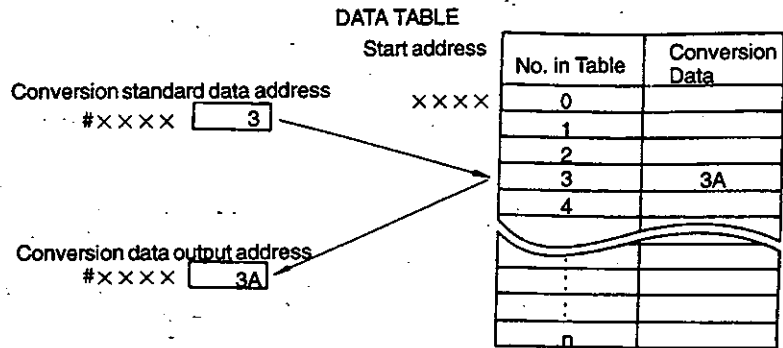


- If the number of steps to the position one step before the target position is calculated while the present position is equal to the target position (POS = 1, INC = 1), the result of calculation is "0".

(5) SUBP 007 (CODE CONVERT)

(a) Function

This instruction converts the data by using the conversion table created on the PLC table.

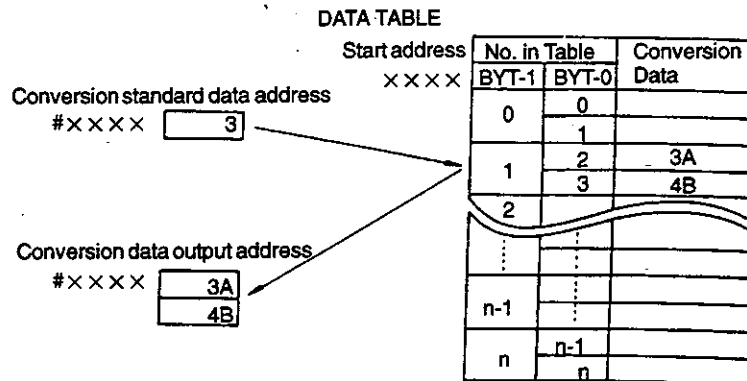


① BYT = 0

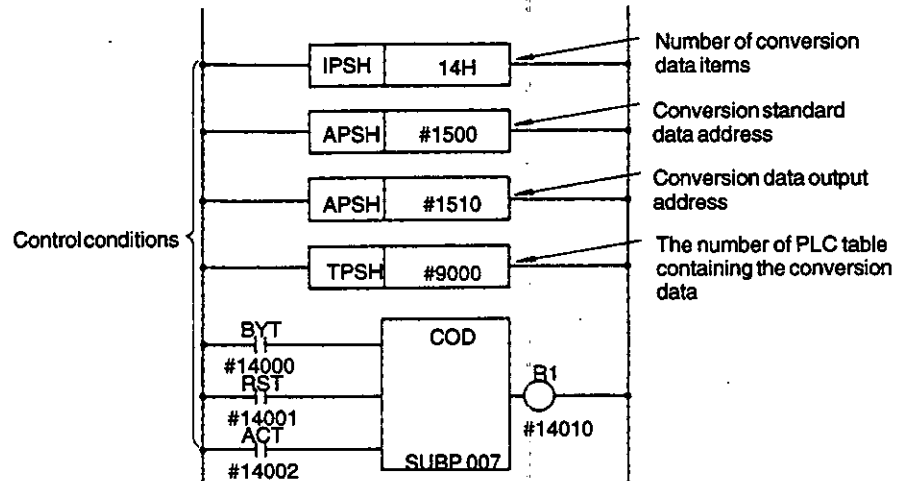
If "3" is specified for the conversion standard data address as shown above, the instruction stores the "third data" from the start of the table to the conversion data output address.
The start of the table is 0th.

② BYT = 1

In this case, the size of the conversion data table should be an even number of bytes.



(b) Format



PLC Table
9000

No. in Table	Conversion Data
0	20H
1	30H
2	40H
17	1AH
18	2BH
19	3CH

- IPSH 14H Size of conversion data table (number of bytes)
- APSH #1500 Conversion standard data address
- APSH #1510 Conversion data output address
- TPSH #9000 PLC table number containing the conversion data
- LD #14000 ... Data in the data table; 1 byte, 2 bytes
- STR #14001 ... Reset
- STR #14002 ... Execution
- SUBP 007 COD instruction
- OUT #14010 ... Error output

- 20H
 - 30H
 - 40H
 -
 - 1AH
 - 12BH
 - 3CH
- } Conversion data table

(c) Control conditions

① Designation of the number of conversion data items (IPSH $\times \times$)

Designate the size of the conversion data table by the number of bytes. The maximum size is 256 bytes.

② Designation of the conversion standard data address (APSH # $\times \times \times \times$)

The data in the conversion data table can be read out by designating the number in the table.

Designate the number in the table.

③ Designation of the conversion data output address (APSH # $\times \times \times \times$)

Designate the address where the data, stored at the number in the table which is specified in item 2 above, should be output.

If "BYT = 1", the upper byte data are output to the address next to the designated address.

④ Designation of the conversion data table (TPSH $\times \times \times \times$)

The size of table differs depending on the PLC table number.

- 9000 to 9007: Max. 256 bytes
- 9008 to 9023: Max. 128 bytes

⑤ Designation of the data size (BYT)

Designate the size of the data in the conversion data table.

BYT = 0: 1 byte

BYT = 1: 2 bytes

⑥ Reset (RST)

Designate whether or not the error output coil R1 is reset.

RST = 0: Not reset

RST = 1: Reset

⑦ Execution command (ACT)

ACT = 0: The COD instruction is not executed. R1 remains unchanged.

ACT = 1: The COD instruction is executed.

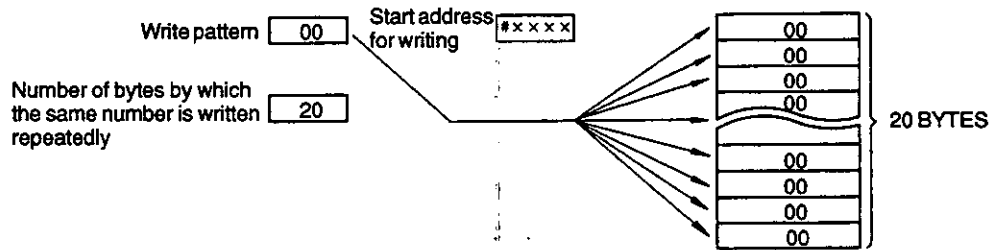
⑧ Error output (R1)

If an error occurs during the execution of the COD instruction (a numeric value greater than the size of the table is set), "1" is set for "R1" (R1 = 1) indicating the occurrence of an error.

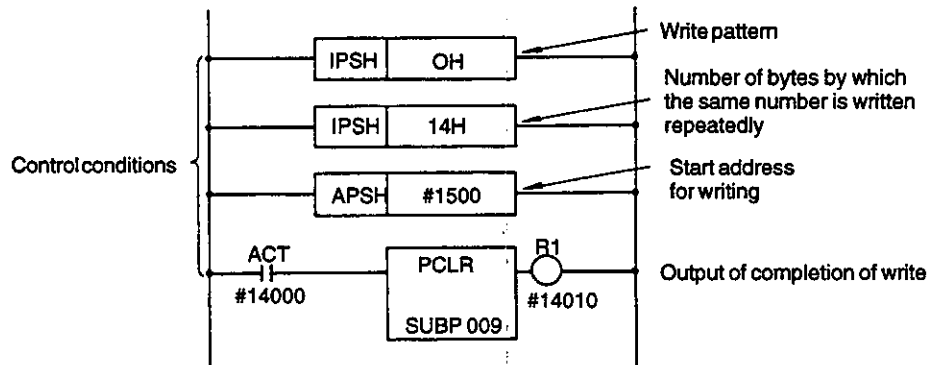
(6) SUBP 009 (PATTERN CLEAR)

(a) Function

The instruction writes the same numeric value repeatedly by the designated number of bytes beginning with the designated address.



(b) Format



- IPSH 0H Write pattern
- IPSH 14H Number of bytes by which the same number is written repeatedly
- APSH #1500 Start address for writing
- LD #14000 Execution
- SUBP #009 PCLR command
- OUT #14010 ... Output of completion of write

(c) Control conditions

① Designation of the write pattern (IPSH $\times \times$)

Designate the pattern to be written.

To designate a variable pattern, use the PUSH instruction to designate the address instead of using the IPSH instruction.

② Designation of the number of bytes by which the same number is written repeatedly (IPSH $\times \times$)

Designate the number of bytes to clear the pattern.

③ Designation of the start address of writing (APSH # $\times \times \times \times$)

Designate the start address of writing.

Pattern clear is executed beginning with this address by the designated number of bytes.

④ Execution command (ACT)

ACT = 0: The PCLR instruction is executed.

ACT = 1: The PCLR instruction is not executed.

⑤ Output of completion of write (R1)

R1 = 0: Writing not completed

R1 = 1: Writing completed

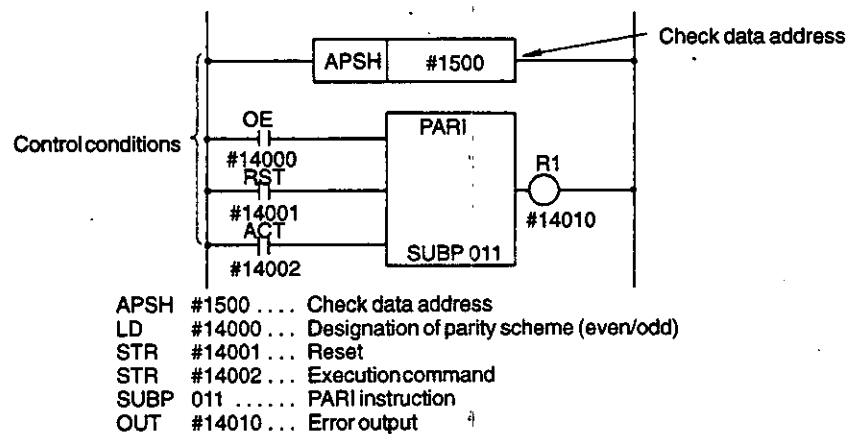
(7) SUBP 011 (PARITY CHECK)

(a) Function

The instruction executes parity check (even parity or odd parity) for the data to be checked (1-byte data).

If an error is detected, an error output is given.

(b) Format



(c) Control conditions

① Designation of check data address (APSH #××××)

Designate the address where the data to be checked are stored.

Parity check is made for 1 byte (8 bits) of data.

② Designation of parity scheme (OE)

OE = 0: Even parity check

OE = 1: Odd parity check

③ Reset (RST)

RST = 0: Error output R1 is not reset.

RST = 1: Error output R1 is reset.

④ Execution command (ACT)

ACT = 0: The PARI instruction is not executed. R1 remains unchanged.

ACT = 1: The PARI instruction is executed.

⑤ Error output (R1)

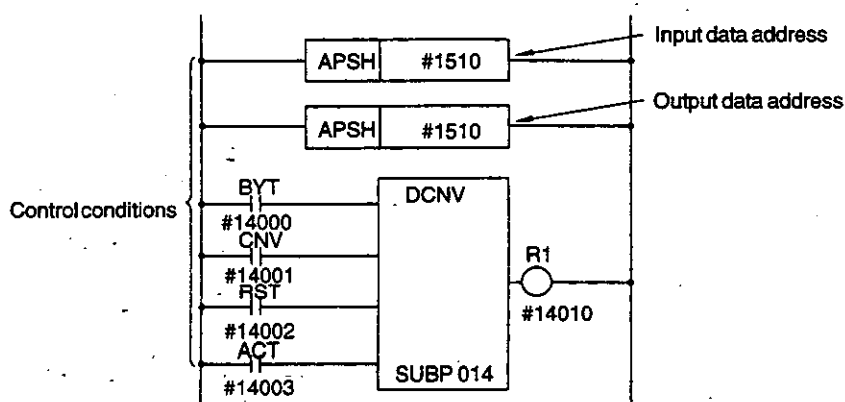
If the result of parity check does not meet the designated parity scheme, "1" is set for "R1" (R1 = 1).

(8) SUBP 014 (DATA CONVERT)

(a) Function

The instruction converts the binary data to the BCD data and the BCD data to the binary data.

(b) Format



APSH #1500 Conversion data address
 APSH #1510 Converted data storing address
 LD #14000 ... 1-byte or 2-byte processing
 STR #14001 ... BCD → BIN or BIN → BCD
 STR #14002 ... Reset
 STR #14003 ... Execution
 SUBP 014 DCNV instruction
 OUT #14010 ... Error output

(c) Control conditions

- ① Designation of the conversion data address (APSH #××××)

Designate the address where the data to be converted are stored.

If "BYT = 1", continuous two bytes are used.

- ② Designation of the converted data address

Designate the address where the result of conversion is stored.

If "BYT = 1", continuous two bytes are used.

- ③ Designation of the number of bytes (BYT)

BYT = 0: The data to be processed are 1-byte data.

BYT = 1: The data to be processed are 2-byte data.

④ Designation of the conversion type (CNV)

CNV = 0: Conversion of binary data to BCD data

CNV = 1: Conversion of BCD data to binary data

⑤ Reset (RST)

RST = 0: Error output R1 is not reset.

RST = 1: Error output R1 is reset.

⑥ Execution command (ACT)

ACT = 0: The DCNV instruction is not executed.

ACT = 1: The DCNV instruction is executed.

⑦ Error output (R1)

R1 = 0: Normal

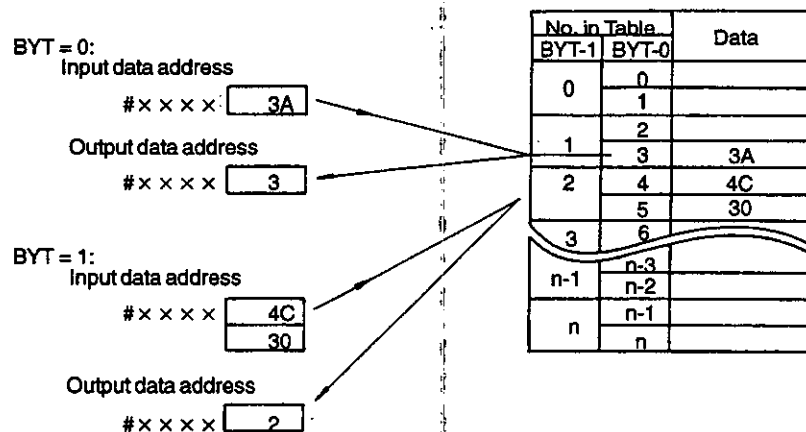
R1 = 1: Error

(An attempt is made to convert the binary data when "CNV = 1", or the byte length is exceeded when "CNV = 0".)

(9) SUBP 017 (DATA SEARCH)

(a) Function

The instruction executes search in the table for the data identical to the input data and stores the address where the identical data are found by the relative address from the start of the table. If the identical data are not found, an error is output.

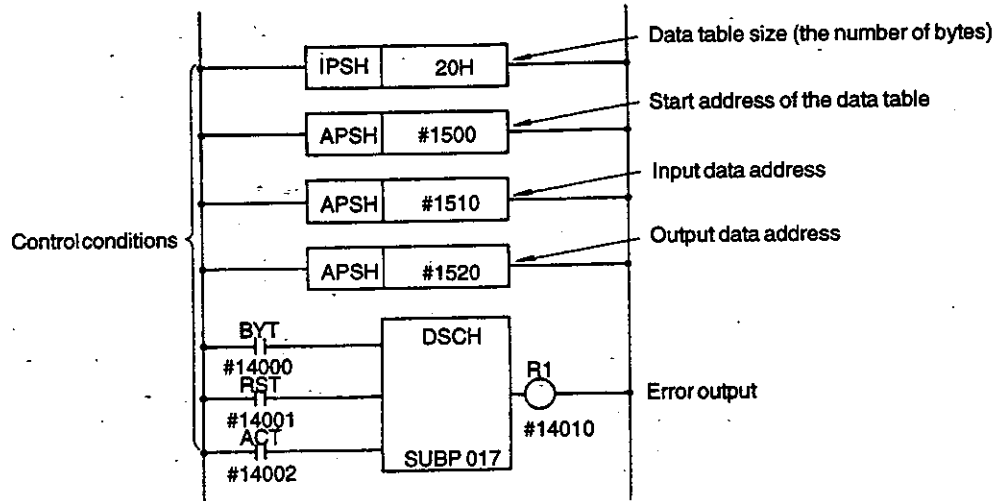


Note 1: When "BYT = 1", the size of the table must be an even number of bytes.

2: If the data to be searched exist at more than one place, the data found first is regarded as the objective data.

3: The data address to be stored is in units of bytes if "BYT = 0" or in units of words if "BYT = 1".

(b) Format



IPSH 20H Data table size (the number of bytes)
 APSH #1500 The start address of the data table
 APSH #1510 Search data address
 APSH #1520 Search result storing address
 LD #14000 ... 1-byte or 2-byte processing
 STR #14001 ... Reset
 STR #14002 ... Execution
 SUBP 017 DSCCH command
 OUT #14010 ... Error output

(c) Control conditions

- ① Designation of the data table size (the number of bytes) (IPSH $\times\times\times\times$)
 Designate the size of the data table by the number of bytes.
- ② Designation of the start address of the data table (APSH $\#\times\times\times\times$)
 Designate the start address of the data table.
 The data table can be created at any place.
- ③ Designation of the input data address (APSH $\#\times\times\times\times$)
 Designate the address where the data to be searched are stored.
- ④ Designation of the output data address (APSH $\#\times\times\times\times$)
 When the specified data are found ($R1 = 0$), the number in the table where the found data are stored is output. Designate the address where that number is stored.

⑤ Designation of the data size (BYT)

BYT = 0: The data stored in the data table are 1-byte data.

BYT = 1: The data stored in the data table are 2-byte data.

⑥ Execution command (ACT)

ACT = 0: The DSCH instruction is executed.

ACT = 1: The DSCH instruction is not executed.

⑦ Reset (RST)

RST = 0: Error output R1 is not reset.

RST = 1: Error output R1 is reset.

⑧ Error output (R1)

R1 = 0: The search data are found.

R1 = 1: The search data are not found.

(10) SUBP 018 (INDEX DATA MOVE)

(a) Function

The instruction reads the data from the data table or rewrites the data in the data table.

① Reading

To read the contents by designating "3" (the number in the table).

Address storing the number in the table

x x x x 3

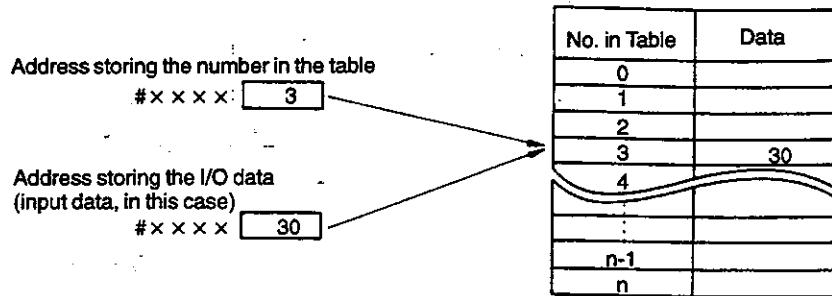
Address storing the I/O data
(output data, in this case)

x x x x 10

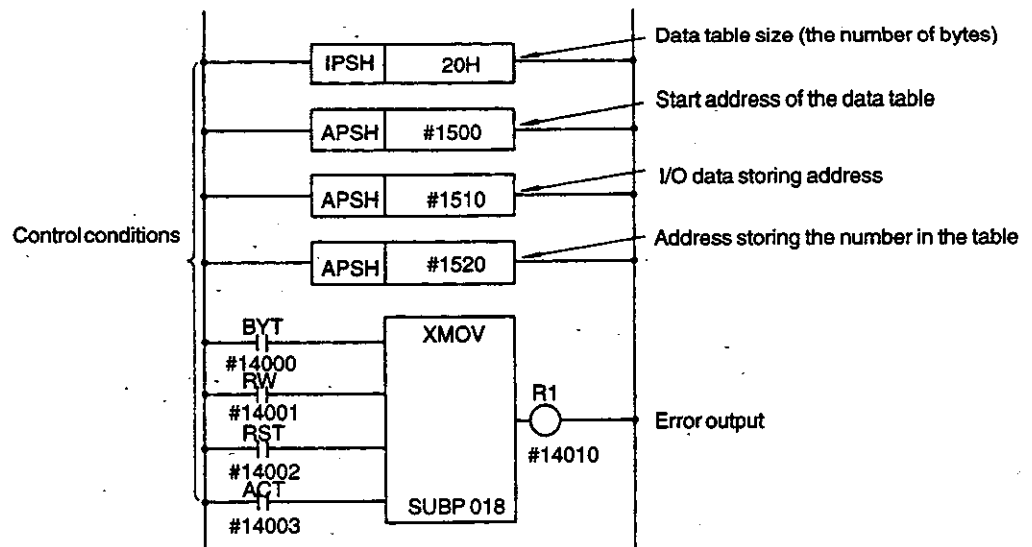
No. in Table	Data
0	
1	
2	
3	10
4	
⋮	
⋮	
n-1	
n	

② Rewriting

To rewrite the contents by designating "3" (the number in the table).



(b) Format



- IPSH 20H Data table size (the number of bytes)
- APSH #1500 The start address of the data table
- APSH #1510 Address storing the I/O data
- APSH #1520 Address storing the number in the table
- LD #14000 ... 1-byte or 2-byte processing
- STR #14001 ... Read or rewrite
- STR #14002 ... Reset
- STR #14003 ... Execution
- SUBP 018 XMOV command
- OUT #14010 ... Error output

(c) Control conditions

- ① Designation of the data table size (number of bytes) (IPSH $\times \times$)

Designate the size of the data table by the number of bytes.

- ② Designation of the start address of the data table (APSH # $\times \times \times \times$)

Designate the start address of the data table.

The data table can be created any place.

- ③ Designation of the address storing the I/O data (APSH # $\times \times \times \times$)

RW = 0: The address where the output data are stored.

RW = 1: The address where the input data are stored.

- ④ Designation of the address storing the number in the table (APSH # $\times \times \times \times$)

The data to be read or rewritten are designated by the number in the table. Designate the address where this number is stored.

- ⑤ Designation of the data size (BYT)

BYT = 0: The data stored in the data table are 1-byte data.

BYT = 1: The data stored in the data table are 2-byte data.

- ⑥ Designation of read/write processing (RW)

RW = 0: Data are read from the data table.

RW = 1: Data in the data table are rewritten.

- ⑦ Reset (RST)

RST = 0: Error output R1 is not reset.

RST = 1: Error output R1 is reset.

- ⑧ Execution command (ACT)

ACT = 0: The XMOV instruction is executed.

ACT = 1: The XMOV instruction is not executed.

- ⑨ Error output (R1)

R1 = 0: Normal

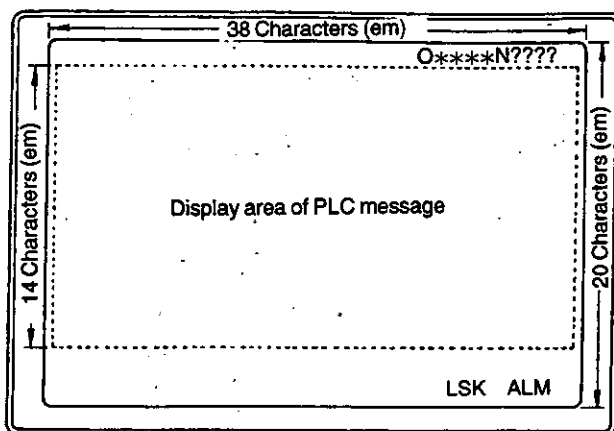
R1 = 1: Error

The address specified for storing the number in the table is outside the allowable range. (Data table size is exceeded.)

(11) SUBP 023 (MESSAGE DISPLAY)

(a) Function

The instruction displays the message on the screen.



- ① The maximum number of characters per line is 38 characters.

Number	Maximum	Quantity	
#9024 to #9323	38 words	300	Message table

- ② If more than 14 message display requests are given for one display screen, 14 lines of messages are displayed in order of priority (lower bit given highest priority).
- ③ The message to be displayed or cleared can be selected by setting "0" or "1" to the corresponding bit. "1" for the message to be displayed and "0" for the message to be cleared.

The correspondence is indicated below.

Display request	7	6	5	4	3	2	1	0	#1500
	15	14	13	12	11	10	9	8	#1501
Display status	7	6	5	4	3	2	1	0	#1502
	15	14	13	12	11	10	9	8	#1503
Display request	23	22	21	20	19	18	17	16	#1504
	31	30	29	28	27	26	25	24	#1505
Display status	23	22	21	20	19	18	17	16	#1506
	31	30	29	28	27	26	25	24	#1507

Note 1: If "1" is set for the bit where no message is stored, blank spaces are displayed.

- 2: This instruction is used to display messages on the screen. It cannot be used to place the NC in the alarm state (1-block stop, stop after deceleration, immediate stop).
- 3: Do not write the data to #1502, #1503, #1506, and #1507, or output the data from these addresses by using the OUT instruction.

- ④ The PLC system has two display screens for display of messages and they are controlled by the DISP (SUBP 023) instruction.

Therefore, if the DISP instruction is specified more than one time for the same message display screen, the display processing is executed more than one time in one scan and the messages cannot be given correctly. (Messages will be written over.)

Even if the DISP instruction is specified more than one time, it will not cause a problem when only one DISP instruction is processed in one scan by designating a JMP or other appropriate instructions.

(b) Format

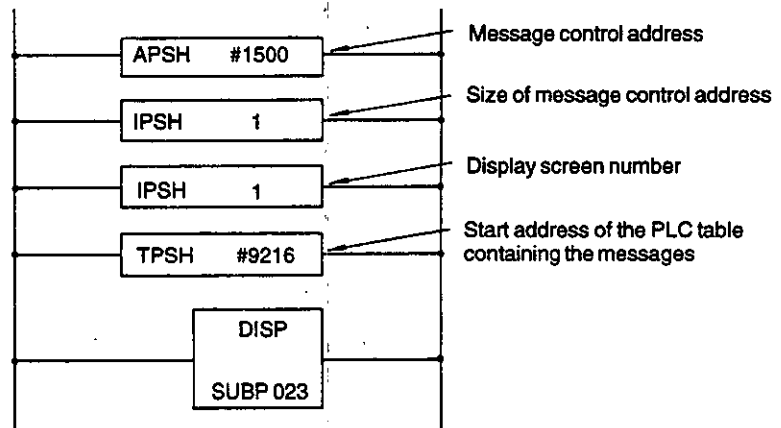


Table Address	Display Request	Message
#9216	#15000	SPINDLE ALARM
#9217	#15001	M06 ERROR
#9218	#15002	TAPPING ERROR
#9219	#15003	
~~~~~		
#9229	#15015	UNUSABLE S-CODE
#9230	#15016	UNUSABLE M-CODE
#9231	#15017	PARAMETER ERROR

APSH #1500 .... Message control address  
 IPSH 1H ..... Size of message control address  
 IPSH 1H ..... Display screen number  
 TPSH #9216 .... Start address of the PLC table containing the messages  
 SUBP 023 ..... DISP instruction

(c) Control conditions

- ① Designation of the message control address (APSH #××××)

Designate the start address of the addresses that request the message.

- ② Designation of the size of message control address (IPSH ××)

Designate the size of the message control address by the number of bytes.

Example: APSH #1500  
IPSH 1H

With the designation indicated above, contiguous four bytes starting from #1500 are used.

If "IPSH 2H" is designated instead of "IPSH 1H", contiguous eight bytes starting from #1500 are used.

Note: If "IPSH 1H" is designated, a maximum of 16 kinds of messages is used.

- ③ Designation of the display screen number (IPSH ×)

Designate the display page number to be used.

A total of two pages (No. 1 and No. 2) can be used for both the high-speed processing and low-speed processing sequence programs.

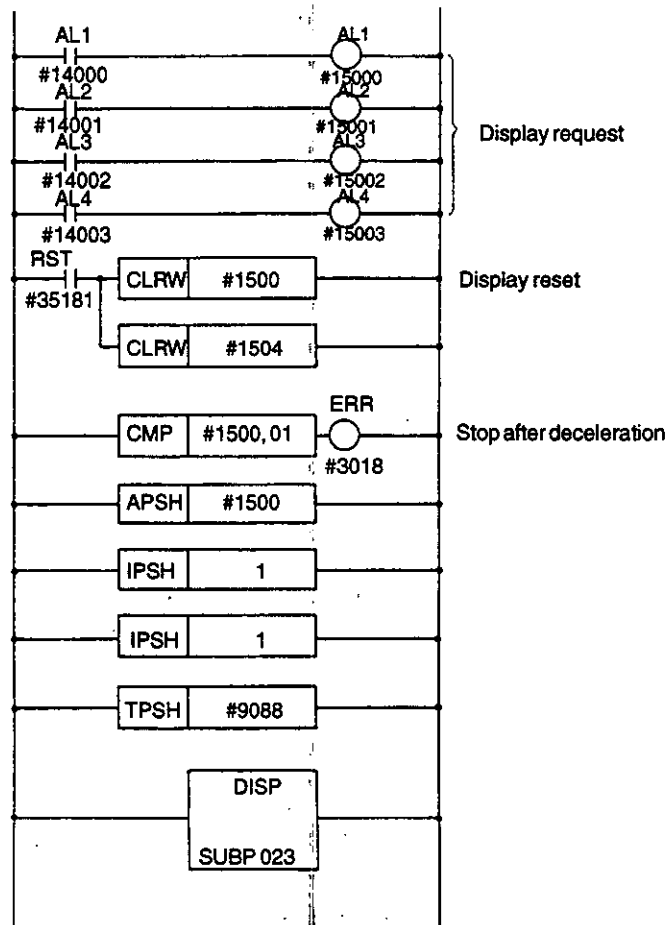
- ④ Designation of the start address of the PLC table containing the messages (TPSH ××××)



## (d) Example of the application of the DISP instruction

When the contact of AL1 to AL4 is turned ON, the message corresponding to the ON bit is displayed on the screen and the machine operation stops after deceleration.

The display is cleared when the reset signal is input.



## (e) Selection of the USER MESSAGE screen

On the USER MESSAGE screen, the message sent from the PLC is displayed.

- ① Press the [COMMON] process soft-key.
- ② Press the [ALM] job soft-key.
- ③ Press the FUNCTION SELECT key.
- ④ User's messages are displayed from the 1st to the 14th line.

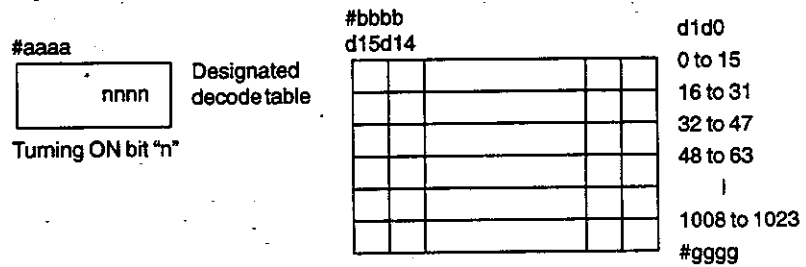
For the display of user's messages, two pages exist. To change the display page between No. 1 and No. 2 pages, use the page keys.

(12) SUBP 025 (Binary Decoding)

(a) Function

The instruction executes decoding of binary data (1 byte or 2 bytes length).

In this decoding, the code data are converted into bits and written to the designated area.



(b) Format

APSH	#aaaa	Code data area
APSH	#bbbb	Start address of the decode table
IPSH	g	Max. decode number
LD	#cccc.c	Selection of byte or word
		0 = Byte
STR	#eeee.e	Reset
STR	#ffff.f	Execution
SUBP	025	
OUT	#dddd.d	Error output

Byte (#cccc.c)

- = 0: Code data . . . . 0 to 255
- = 1: Code data . . . . 0 to 1023

- Decode table start number
- Max. decode number: 1 to 1023

(c) Control conditions

① Designation of the data code area (APSH  $\times \times$ )

Designate the address of the code to be decoded.

Two bytes are used.

② Designation of the start address of the decode table (APSH  $\times \times \times \times$ )

Designate the start address of the table to be decoded.

③ Maximum decode number (IPSH  $\times \times \times \times$ )

Designate the maximum number of the decode bits.

④ Designation of the size of the data in the decode area (LD  $\times \times$ )

LD = 0: The size of the data in the conversion table is 1 byte.

LD = 1: The size of the data in the conversion table is 2 bytes.

⑤ Reset (RST)

RST = 0: Error output R1 is not reset.

RST = 1: Error output R1 is reset.

⑥ Execution command (ACT)

ACT = 0: The binary decode instruction is executed. R1 remains unchanged.

ACT = 1: The binary decode instruction is not executed.

⑦ Error output (R1)

R1 = 0: Normal

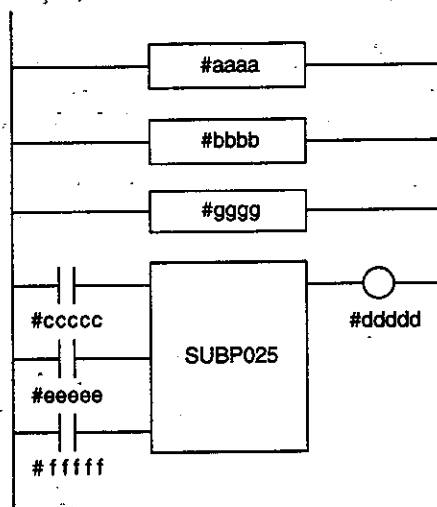
R1 = 1: Error

A numeric value greater than the table size is set.



(d) Example of ladder

APSH	#aaaa	Code data area
APSH	#bbbb	Start address of the decode table
IPSH	#gggg	Max. decode number
LD	#cccc.c	Selection of byte or word 0 = Byte
STR	#eeee.e	Reset
STR	#ffff.f	Execution
SUBP	025	
OUT	#dddd.d	Error output



(13) SUBP 027 (Binary Code Conversion)

(a) Function

The instruction converts the data by using the conversion table created on the PLC table.

From the contents of data "0" to "n" (= "m") of the conversion standard data address, the data in the "m"th line from the start of the table data are read and output to the conversion data output address.

For the type of output data, selection is possible from byte, word, and double-word.

The relationship between the number in the table and the table is as indicated below according to the data length.

Double-word	Word	Byte	Table
0	0	0	
		1	
	1	2	
		3	
1	2	4	
.	.	.	
.	.	.	
n	n	n	

(b) Format

IPSH	a	Size of the conversion table (number of bytes)
APSH	#bbbb	Conversion standard data address
APSH	#cccc	Conversion data output address
TPSH	#9ddd	Table number of the conversion data
IPSH	e	e = 0: Byte 1: Word 2: Double-word
LD	#fff.f	Reset
STR	#ggg.g	Execution
SUBP	027	
OUT	#hhh.h	Error output

(c) Control conditions.

① Designation of the size of the conversion data table (IPSH × ×)

Designate the size of the conversion data table by the number of bytes.

The maximum size is 256 bytes.

② Designation of the conversion standard data table address (APSH × × × ×)

The data in the conversion data table can be read by designating the number in the table.

Designate this number in the table.

③ Designation of the conversion data output address (APSH × × × ×)

Designate the address where the data stored at the number in the table designated in item ② above are output.

④ Designation of the number of conversion table (TPSH × × × ×)

The size of the table differs depending on the PLC number.

9000 to 9007: Max. 256 bytes

9008 to 9023: Max. 128 bytes

⑤ Designation of the size of the data in conversion table (IPSH × ×)

e = 1: 1 byte

e = 2: 2 bytes

e = 3: 4 bytes

⑥ Reset (RST)

RST = 0: Error output R1 is not reset.

RST = 1: Error output R1 is reset.

⑦ Execution command (ACT)

ACT = 0: The binary code conversion instruction is executed.  
R1 remains unchanged.

ACT = 1: The binary code conversion instruction is not executed.

⑧ Error output (R1)

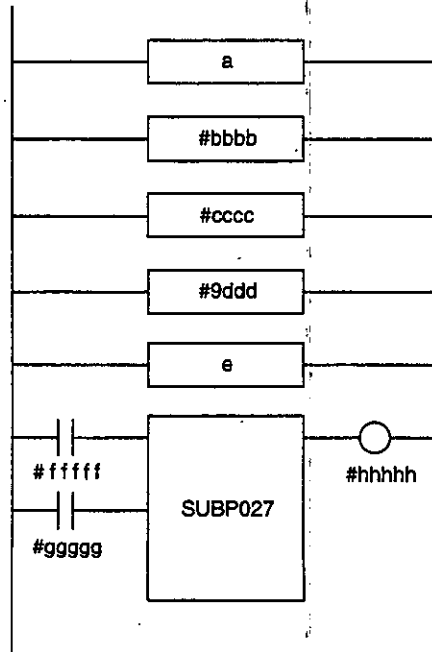
R1 = 0: Normal

R1 = 1: Error

A numeric value greater than the table size is set.

(d) Example of ladder

IPSH	a	Size of the conversion table (number of bytes)
APSH	#bbbb	Conversion standard data address
APSH	#cccc	Conversion data output address
TPSH	#9ddd	Table number of the conversion data
IPSH	e	= 0: Byte = 1: Word = 2: Double-word
LD	#ffff.f	Reset
STR	#gggg.g	Execution
SUBP	027	
OUT	#hhhh.h	Error output



(14) SUBP 031 (Double-word Data Convert)

(a) Function

The instruction converts the binary data to BCD data and the BCD data to binary data.

(b) Format

APSH	#aaaa	Conversion data address
APSH	#bbbb	Converted data address
LD	#cccc.c	= 1: 4 bytes = 0: Skip
STR	#dddd.d	BCD → BIN or BIN → BCD
STR	#eeee.e	Reset
STR	#ffff.f	Execution
SUBP	031	
OUT	#gggg.g	Error output

(c) Control conditions

① Designation of the conversion data address (APSH × × × ×)

Designate the address where the data to be converted are stored.

Both the binary and BCD data use 4 bytes.

The sign of the BCD data is set at the most significant bit position.

Therefore, the expression of a numeric value within the range of ± 9999999 is possible.

② Designation of the converted data address

Designate the address where the result of conversion is stored.

Both the binary and BCD data use 4 bytes.

③ Designation of the number of bytes

LD = 0: No conversion

④ Designation of the conversion type

STR = 0: Conversion of binary data to BCD data

STR = 1: Conversion of BCD data to binary data

⑤ Reset (RST)

RST = 0: Error output R1 is not reset.

RST = 1: Error output R1 is reset.



⑥ Execution command (ACT)

ACT = 0: The double-word data conversion instruction is not executed.

ACT = 1: The double-word data conversion instruction is executed.

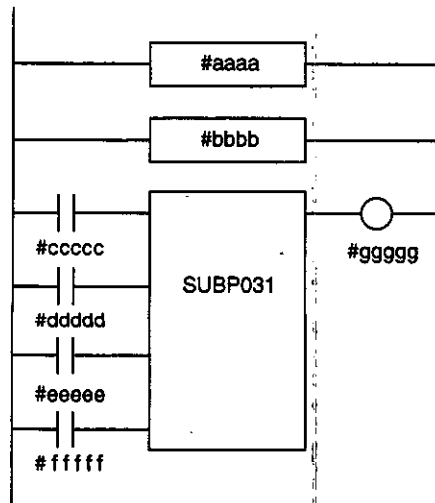
⑦ Error output (R1)

R1 = 0: Normal

R1 = 1: Error

(d) Example of ladder

APSH	#aaaa	Conversion data address
APSH	#bbbb	Converted data address
LD	#cccc.c	= 1: 4 bytes = 0: Skip
STR	#dddd.d	BCD → BIN or BIN → BCD
STR	#eeee.e	Reset
STR	#ffff.f	Execution
SUBP	031	
OUT	#gggg.g	Error output



(15) SUBP 032 (Binary Comparison)

(a) Function

The instruction executes comparison of the 1-, 2-, or 4-byte length binary data and outputs the result of comparison. Both the input data and the data for comparison must be the data of the specified length.

(b) Format

APSH	#aaaa	Input data address
APSH	#bbbb	Comparison data address
APSH	#hhhh	Comparison result output address
IPSH	n	= 0: 1 byte = 1: 2 bytes = 2: 4 bytes
LD	#ffff.f	Execution command
SUBP	032	

(c) Control conditions

① Designation of the input data address (APSH  $\times \times \times \times$ )

Designate the address where the data to be compared are stored.

② Designation of the comparison data address

Designate the address where the comparison data are stored.

③ Designation of the size of the data

= 0: 1 byte  
= 1: 2 bytes  
= 2: 4 bytes

④ Execution command (ACT)

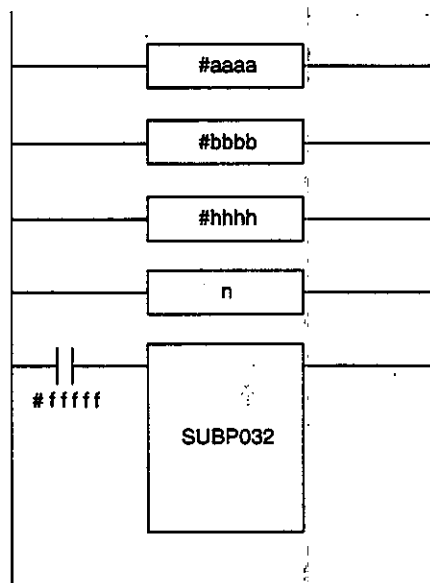
ACT = 0: The binary data comparison instruction is not executed.  
ACT = 1: The binary data comparison instruction is executed.

⑤ Result of comparison

#hhhh  
d0 = 1    aaaa = bbbb  
d1 = 1    aaaa > bbbb  
d2 = 1    aaaa < bbbb

(d) Example of ladder

APSH	#aaaa	Input data address
APSH	#bbbb	Comparison data address
APSH	#hhh	Comparison result output address
IPSH	n	= 0: 1 byte = 1: 2 bytes = 2: 4 bytes
LD	#ffff.f	Execution command
SUBP	032	



(16) SUBP 034 (Binary Data Search)

(a) Function

The instruction executes search in the table for the data identical to the input data and stores the address where the identical data are found by the relative address from the start of the table. If the identical data are not found, an error is output.

The relationship between the numbers in the data table and the designated table is as indicated below according to the data length.

Double-word	Word	Byte	Table
0	0	0	
		1	
	1	2	
		3	
1	2	4	
.	.	.	
.	.	.	
.	.	.	
n	n	n	

(b) Format

IPSH	a	The size of the data table (number of bytes)
APSH	#bbbb	Start address of the data table
APSH	#hhhh	Search data address
APSH	#cccc	Search result storing address
IpsH	e	= 0: Byte = 1: Word = 2: Double-word
LD	#fff.f	Reset
STR	#gggg.g	Execution
SUBP	034	
OUT	#hhhh.h	Error output

(c) Control conditions

- ① Designation of the size of the data table (number of bytes) (IPSH  $\times \times$ )

Designate the size of the data table by the number of bytes.

- ② Designation of the start address of the data table (APSH  $\times \times \times \times$ )

Designate the start address of the data table.

The data table can be created at any place.

- ③ Designation of the input data address (APSH  $\times \times \times \times$ )

Designate the address where the data to be searched are stored.

- ④ Designation of the output data address (APSH #  $\times \times \times \times$ )

When the specified data are found ( $R1 = 0$ ), the number in the table where the found data are stored is output. Designate the address where that number is stored.

- ⑤ Designation of the data size (IPSH  $\times \times$ )

= 0: The data stored in the data table are 1-byte data.

= 1: The data stored in the data table are 2-byte data.

= 2: The data stored in the data table are 4-byte data.

- ⑥ Reset (RST)

RST = 0: Error output R1 is not reset.

RST = 1: Error output R1 is reset.

- ⑦ Execution command (ACT)

ACT = 0: The binary-data search instruction is executed. R1 remains unchanged.

ACT = 1: The binary-data search instruction is not executed.

- ⑧ Error output (R1)

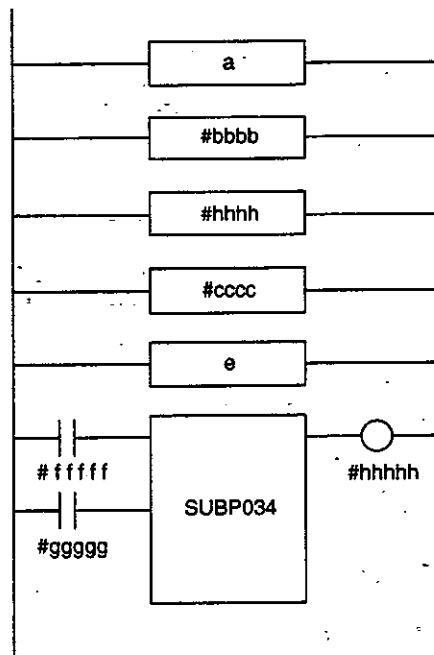
R1 = 0: The search data are found.

R1 = 1: The search data are not found.



(d) Example of ladder

IPSH	a	The size of the data table (number of bytes)
APSH	#bbb	Start address of the data table
APSH	#hhh	Search data address
APSH	#ccc	Search result storing address
Ipsh	e	= 0: Byte = 1: Word = 2: Double-word
LD	#fff.f	Reset
STR	#ggg.g	Execution
SUBP	034	
OUT	#hhh.h	Error output



(17) SUBP 035 (Binary Index Modifier Data Transfer)

(a) Function

The instruction reads the data from the data table or rewrites the data in the data table.

For the output data, selection is possible from byte, word, and double-word.

The relationship between the numbers in the data table and the designated table is as indicated below according to the data length.

Double-word	Word	Byte	Table
0	0	0	
		1	
	1	2	
		3	
1	2	4	
.	.	.	
.	.	.	
.	.	.	
n	n	n	

(b) Format

IPSH	a	The size of the data table (number of bytes)
APSH	#bbbb	Start address of the data table
APSH	#iiii	I/O data storing address
APSH	#cccc	Address storing the number in the table
IPSH	e	= 0: Byte = 1: Word = 2: Double-word
LD	#fff.f	Reset
STR	#ddd.d	Read or rewrite
STR	#ggg.g	Execution
SUBP	035	
OUT	#hhh.h	Error output

(c) Control conditions

- ① Designation of the size of the data table (number of bytes) (IPSH  $\times \times$ )

Designate the size of the data table by the number of bytes.

- ② Designation of the start address of the data table (APSH  $\times \times \times \times$ )

Designate the start address of the data table.

The data table can be created at any place.

- ③ Designation of the address storing the I/O data (APSH  $\times \times \times \times$ )

Designate the address where the data to be searched are stored.

- ④ Designation of the address storing the number in the table (APSH  $\times \times \times \times$ )

The data to be read or rewritten are designated by the number in the table. Designate the address where this number is stored.

- ⑤ Designation of the data size (IPSH  $\times \times$ )

= 0: The data stored in the data table are 1-byte data.

= 1: The data stored in the data table are 2-byte data.

= 2: The data stored in the data table are 4-byte data.

- ⑥ Designation of read/write processing

= 0: Data are read from the data table.

= 1: Data in the data table are rewritten.

- ⑦ Reset (RST)

RST = 0: Error output R1 is not reset.

RST = 1: Error output R1 is reset.

- ⑧ Execution command (ACT)

ACT = 0: The binary-index modifier data transfer instruction is executed.  
R1 remains unchanged.

ACT = 1: The binary-index modifier data transfer instruction is not executed.



⑨ Error output (R1)

If an error occurs when this instruction is executed, "1" is set for "R1" (R1 = 1) to indicate the occurrence of an error.

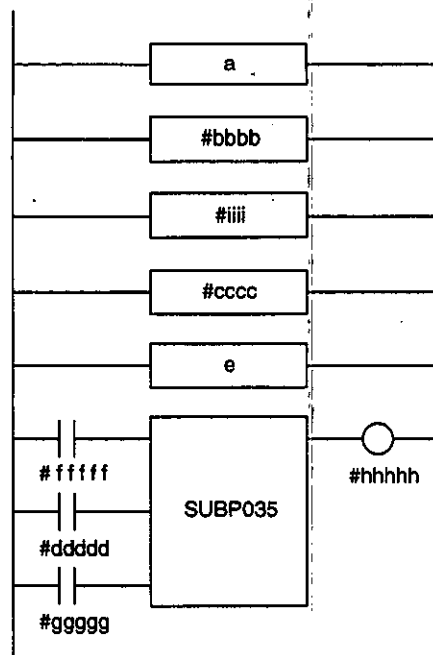
An error occurs in the following cases:

- A numeric value greater than the size of the data table is set.
- The size of the data table is not a multiple of the designated data size.

Example: Data size = Byte       $1 \times N$  bytes  
                                          Word       $2 \times N$  bytes  
                                          Double-word  $4 \times N$  bytes

(d) Example of ladder

IPSH	a	The size of the data table (number of bytes)
APSH	#bbbb	Start address of the data table
APSH	#iiii	I/O data storing address
APSH	#cccc	Address storing the number in the table
IPSH	e	= 0: Byte = 1: Word = 2: Double-word
LD	#ffff.f	Reset
STR	#dddd.d	Read or rewrite
STR	#gggg.g	Execution
SUBP	035	
OUT	#hhhh.h	Error output



(18) SUBP 036 (Binary-data Addition)

(a) Function

The result of operation is set to the registers: the numeric value of operation result is set to the register designated by the operation result output address and the sign information to #2999.

(b) Format

APSH	#aaaa	Augend data address
APSH(ipshd)	#bbbb	Addend data address
APSH	#cccc	Operation result output address
IPSH	n	Operation type
LD	#ddd.d	Reset
STR	#eee.e	Execution
SUBP	036	
OUT	#fff.f	Error output

(c) Control conditions

① Designation of the augend data address

Designate the address where the augend data are stored.

② Designation of the addend data address

Designate the address where the addend data are stored.

③ Designation of the operation result output address

Designate the address where the result of operation is output.

The address is stored as 4-byte data.

④ Designation of the type of operation

Designate the data length and the data type of augend/addend.

1st digit: = 0: 1 byte

= 1: 2 bytes

= 2: 4 bytes

2nd digit: = 0: Constant data

= 1: Address data

⑤ Execution command (ACT)

ACT = 0: The binary-data addition instruction is executed.  
R1 remains unchanged.

ACT = 1: The binary-data addition instruction is not executed.

⑥ Error output

= 0: Normal

= 1: Error

⑦ #2999

Operation status is written.

d0 = 1: 0

d1 = 1: Negative

d5 = 1: Overflow

(19) SUBP 037 (Binary-data Subtraction)

(a) Function

The instruction executes subtraction of the 1-, 2-, or 4-byte binary data.

The result of operation is set to the registers: the numeric value of operation result is set to the register designated by the operation result output address and the sign information to #2999.

(b) Format

APSH	#aaaa	Minuend data address
APSH(ipshd)	#bbbb	Subtrahend data address
APSH	#cccc	Operation result output address
IPSH	n	Operation type
LD	#dddd.d	Reset
STR	#eeee.e	Execution
SUBP	037	
OUT	#ffff.f	Error output

(c) Control conditions

① Designation of the minuend data address

Designate the address where the minuend data are stored.

② Designation of the subtrahend data address

Designate the address where the subtrahend data are stored.

It is also possible to subtract the designated data.

③ Designation of the operation result output address

Designate the address where the result of operation is output.

The address is stored as 4-byte data.

④ Designation of the type of operation

Designate the data length and the data type of minuend/subtrahend.

1st digit: = 0: 1 byte

= 1: 2 bytes

= 2: 4 bytes

2nd digit: = 0: Constant data

= 1: Address data

⑤ Execution command (ACT)

ACT = 0: The binary-data subtraction instruction is executed.  
R1 remains unchanged.

ACT = 1: The binary-data subtraction instruction is not executed.

⑥ Error output

= 0: Normal

= 1: Error

⑦ #2999

Operation status is written.

d0 = 1: 0

d1 = 1: Negative

d5 = 1: Overflow

(20) SUBP 038 (Binary-data Multiplication)

(a) Function

The instruction executes multiplication of 1-, 2-, or 4-byte binary data.

The result of operation is set to the registers: the numeric value of operation result is set to the register designated by the operation result output address and the sign information to #2999.

(b) Format

APSH	#aaaa	Multiplicand data address
APSH(ipshd)	#bbbb	Multiplier data address
APSH	#cccc	Operation result output address
IPSH	n	Operation type
LD	#dddd.d	Reset
STR	#eeee.e	Execution
SUBP	038	
OUT	#ffff.f	Error output

(c) Control conditions

① Designation of the multiplicand data address

Designate the address where the multiplicand data are stored.

② Designation of the multiplier data address

Designate the address where the multiplier data are stored.

③ Designation of the operation result output address

Designate the address where the result of operation is output.

The address is stored as 4-byte data.

④ Designation of the type of operation

Designate the data length and the data type of multiplicand/multiplier.

1st digit: = 0: 1 byte  
          = 1: 2 bytes  
          = 2: 4 bytes

2nd digit: = 0: Constant data  
          = 1: Address data

⑤ Execution command (ACT)

ACT = 0: The binary-data multiplication instruction is executed.  
          R1 remains unchanged.

ACT = 1: The binary-data multiplication instruction is not executed.



⑥ Error output

= 0: Normal

= 1: Error

⑦ #2999

Operation status is written.

d0 = 1: 0

d1 = 1: Negative

d5 = 1: Overflow

(21) SUBP 039 (Binary-data Division)

(a) Function

The instruction executes division of 1-, 2-, or 4-byte binary data.

The result of operation is set to the registers: the numeric value of operation result is set to the register designated by the operation result output address and the sign information to #2999.

(b) Format

APSH	#aaa	Dividend data address
APSH (ipshd)	#bbbb	Divisor data address
APSH	#ccc	Operation result output address
IPSH	n	Operation type
LD	#ddd.d	Reset
STR	#eee.e	Execution
SUBP	039	
OUT	#fff.f	Error output

---

(c) Control conditions

① Designation of the dividend data address

Designate the address where the dividend data are stored.

② Designation of the divisor data address

Designate the address where the divisor data are stored.

③ Designation of the operation result output address

Designate the address where the result of operation is output.

The address is stored as 4-byte data.

④ Designation of the type of operation

Designate the data length and the data type of dividend/divisor.

1st digit: = 0: 1 byte

= 1: 2 bytes

= 2: 4 bytes

2nd digit: = 0: Constant data

= 1: Address data

⑤ Execution command (ACT)

ACT = 0: The binary-data division instruction is executed.  
R1 remains unchanged.

ACT = 1: The binary-data division instruction is not executed.

⑥ Error output

= 0: Normal

= 1: Error

⑦ #2999

Operation status is written.

d0 = 1: 0

d1 = 1: Negative

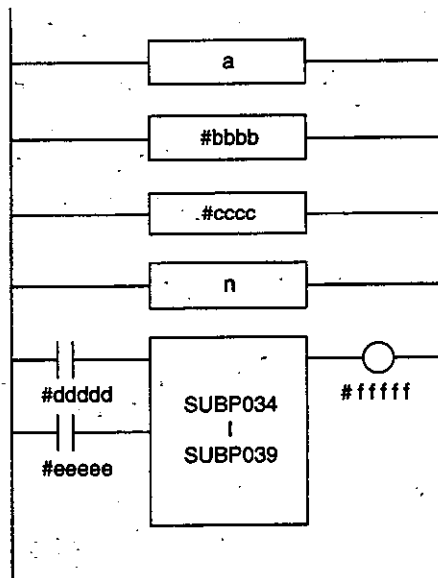
d5 = 1: Overflow

Division by "0" causes overflow.



(d) Example of ladder

APSH	#aaaa	Augend data address
APSH	#bbbb	Addend data address
APSH	#cccc	Operation result output address
IPSH	n	Operation type
LD	#dddd.d	Reset
STR	#eeee.e	Execution
SUBP	036 to 040	
OUT	#ffff.f	Error output





(22) SUBP 040 (Binary Constant Definition)

(a) Function

The instruction defines 1-, 2-, or 4-byte data.

Set a constant (decimal) to the constant output address by the specified number of bytes in the binary number.

(b) Format

APSH	#aaaa	Output address
IPSHD	12345	Setting data
IPSH	n	Byte length
LD	#bbbb.b	Execution
SUBP	40	

(c) Control conditions

① Designation of the constant output address

Designate the address where the data are output in binary format.

② Designation of setting data

Set the constant in decimal.

Data length must be within the specified byte length. The range of setting data is  $\pm 999999999$ .

③ Designation of byte length

= 0: 1 byte

= 1: 2 bytes

= 2: 4 bytes

④ Execution command (ACT)

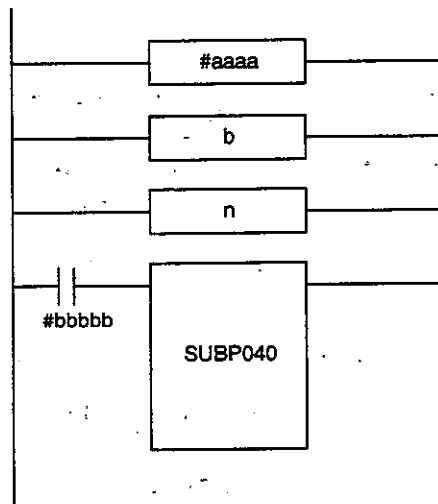
ACT = 0: The binary constant definition instruction is executed.  
R1 remains unchanged.

ACT = 1: The binary constant definition instruction is not executed.



(d) Example of ladder

APSH	#aaaa	Output address
IPSHD	b	Setting data
IPSH	n	Byte length
LD	#bbbb.b	Execution
SUBP	40	







# 7

---

## JXSD OFFLINE SYSTEM

Chapter 7 describes the JXSD offline system.

7.1	OUTLINE OF THE JXSD OFFLINE SYSTEM .....	7 - 2
7.2	SOURCE FILE .....	7 - 6
7.3	COMPILER .....	7 - 18
7.4	LINKER .....	7 - 21
7.5	REMOTE CONTROLLER OPERATION .....	7 - 24
7.6	LIST OF ERROR MESSAGES AND WARNING MESSAGES .....	7 - 33



---

## 7.1 OUTLINE OF THE JXSD OFFLINE SYSTEM

The JXSD offline system is used to create the sequence ladder by using the compiler, linker, and download tools among the utilities provided for the development of PLC sequence programs. These tools run on the MS-DOS.

### 7.1.1 Operating Environment

Hardware: IBM PC compatibles  
OS: MS-DOS Ver. 6.2 or above  
Memory: 400K bytes minimum

### 7.1.2 Execution Files

The JXSD offline systems consists of the following software packages.

- JXLCOMP.EXE Ladder language compiler
- JXLLINK.EXE Linker
- JXPCCOM.EXE JXSD remote controller

### 7.1.3 Outline of the Execution Files

#### (1) Ladder Language Compiler

The compiler compiles the source file, coded using the ladder language, to generate the object file.

The processing objective data by the compiler is indicated below.

- Version No.
- High-speed scan ladder program
- Low-speed scan ladder program
- Low-speed ladder stop count
- Conversion data table
- Message data table
- Symbol data

#### (2) Linker

The linker generates the binary file in the executable format from the object file which is output by the compiler.

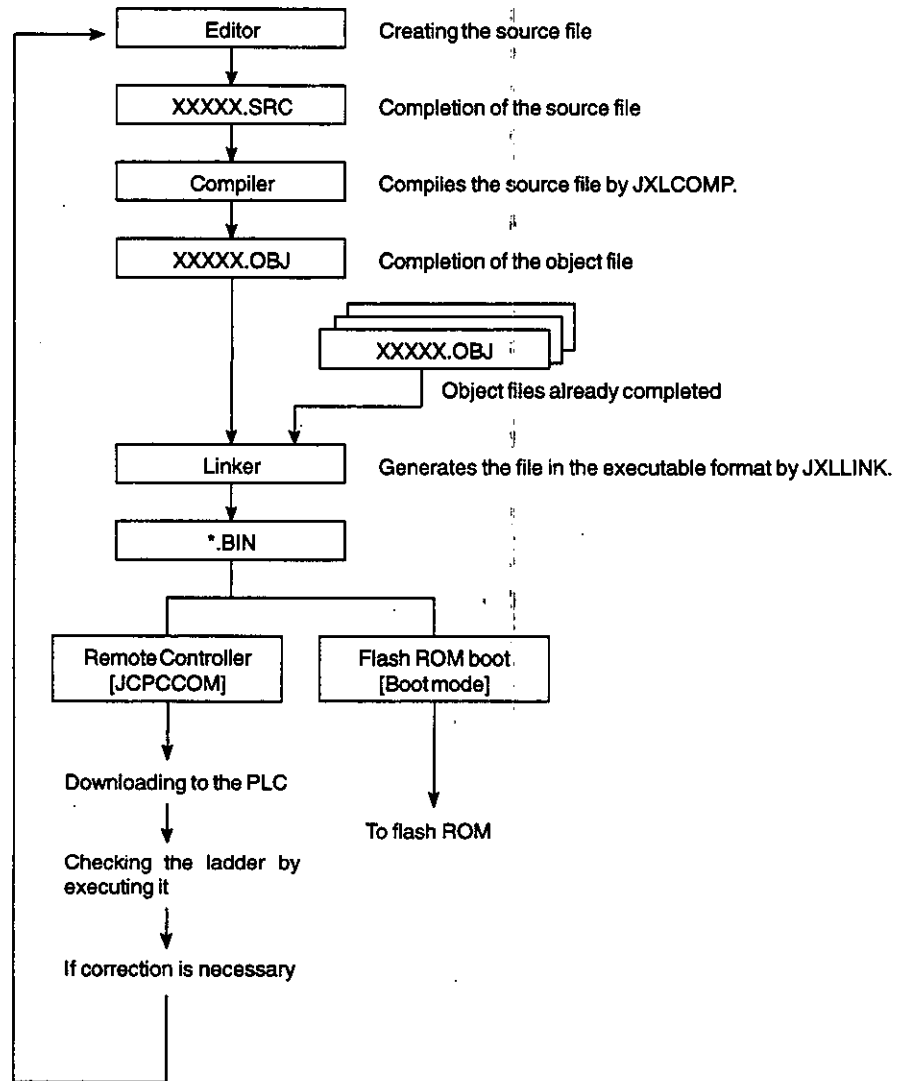
#### (3) JXSD Remote Controller

By connecting the PLC and the IBM PC compatible personal computer with the RS-232C interface, the JXSD remote controller executes the following processing.

- Display of the PLC's ladder execution status
- Downloading the binary file to the PLC (PC → PLC)
- Execution/stop control of the ladder

### 7.1.4 Ladder Program Development Procedure

The following chart shows the procedure for developing the ladder program.



- ① Create the source file in the ladder language.

Any editor that can create a DOS file can be used for creating the source file in the ladder language.

Create the source file by using an appropriate editor.

For details of the ladder language format, refer to the explanation on the compiler.

#### YELADDER.SRC

```
*****
*           YELADDER.SRC           *
*****
VERSION JXSD LADDER
LOWSTOPCOUNT1      ; Low-speed scan ladder program stop count

HIGHSEQUENCE        ; High-speed scan ladder program
INCLUDE LAD.HI
ENDP

LOWSEQUENCE         ; Low-speed scan ladder program
INCLUDE LAD.LO1
INCLUDE LAD.LO2
INCLUDE LAD.LO3
ENDP

CONVERSION          ; Conversion data
INCLUDE CONV.DAT
ENDP

MESSAGE            ; Message data
INCLUDE MESSAGE.DAT
ENDP

SYMBOL             ; Symbol data
INCLUDE SYMBOL.DAT
ENDP
```

- ② Compile the created or modified source file.

Generate the object file by using the JXLCOMP.

For the operation procedure, refer to the explanation on the compiler.



- ③ By consolidating the object files into a single file, generate the executable file. Use the JXLLINK to generate the executable file.

For the operation procedure, refer to the explanation on the linker.

This link processing is always necessary even if only one object file has been generated.

The executable file (*.BIN) generated by the linker is the binary file having the same configuration as the file written to the PLC's flash ROM.

- ④ Download the executable file (*.BIN) to the PLC.

Connect the Personal Computer to the JXSD with an RS-232C cable and start the JXPCCOM. Use the object download function of the JXPCCOM to download the generated executable files to the PLC.

For the operation procedure of the JXPCCOM, refer to the explanation on the remote controller.

- ⑤ Execute and check the ladder.

After downloading the ladder, execute it to check the contents.

If an error is found in the ladder, correct the error by returning to step ① above.

To reduce this debugging time, it is recommended to divide the source file into several files so that the compiling time can be reduced.

- ⑥ Boot the sequence ladder to the flash ROM.

After finishing debugging, boot the correct sequence ladder by using the NC boot mode.



---

## 7.2 SOURCE FILE

The format of the source file input to the compiler is described below.

### 7.2.1 Source File Format

#### (1) Definition of Character Codes

- All data including the comments and character data must be ASCII. Although upper case and lower case characters can be used, they are not distinguished for the internal processing. When entering characters in ladder programs, pay attention to this point:
- Note that all characters are processed in upper case characters.

#### (2) Definition of Numeric Values

- Decimal number           9,1234
- Hexadecimal number       1234H, 0ab12H, 0FFH (note)
- Characters                aBc, a, Z
- Contact/ladder table number   #1000, #10012, #9024

Note: Place "0" at the beginning of a hexadecimal number which begins with A to F.

#### (3) Pseudo Instructions

The following characters are processed as pseudo instructions. These pseudo instructions can be used only once in one source file.

- version
- lowsequence
- message
- include
- highsequence
- lowstopcount
- endp
- conversion

#### (4) Definition of Version Number

For one object file where high-speed scan ladder, low-speed scan ladder, tables and symbols are consolidated, one version number is assigned.

### (5) Nesting in the Source File

The source file of a ladder program will usually be a very large file and editing is not a simple task.

In compilation, the included file function allows the several divided files to be compiled in one file.

[Main][High-speed scan sequence][Low-speed scan sequence 1][Low-speed scan sequence 2]

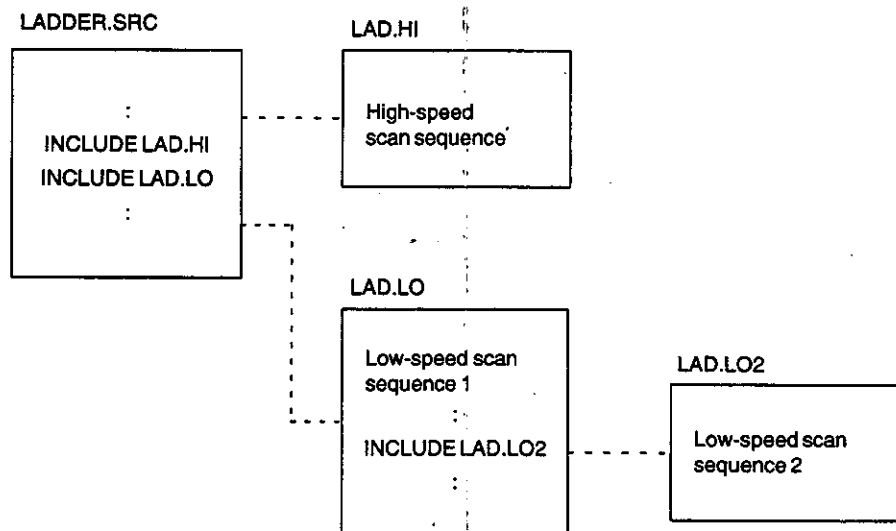


Fig. 7.1

- As illustrated above, nesting of the files is possible up to two levels.
- Pseudo instructions for the start and end of a high-speed scan/low-speed scan ladder (HIGHSEQUENCE, LOWSEQUENCE, ENDP) must always be written in a main file.

(a) Main file

The format of a source file is described below using this as an example.

YELADDER.SRC (main file)

```
*****  
* YELADDER.SRC *  
*****  
① VERSION JXSD LADDER  
② LOWSTOPCOUNT2 ; Low-speed scan ladder program stop count  
③ HIGHSEQUENCE ; High-speed scan ladder program  
④ INCLUDE LAD.HI  
⑤ ENDP  
⑥ LOWSEQUENCE ; Low-speed scan ladder program  
INCLUDE LAD.LO1  
INCLUDE LAD.LO2  
INCLUDE LAD.LO3  
ENDP  
⑦ CONVERSION ; Conversion data  
INCLUDE CONV.DAT  
ENDP  
⑧ MESSAGE ; Message data  
INCLUDE MESSAGE.DAT  
ENDP  
⑨ SYMBOL ; Symbol data  
INCLUDE SYMBOL.DAT  
ENDP
```

Fig. 7.2

1) Source file name

Source file name can be assigned as required by using an extension of ".SRC".

_____.SRC

2) Source file format

- There are no restrictions on start position, the number of lines and the number of columns for entering the pseudo instructions, sequence program, and data.
- Characters appearing after ";" in a line are regarded as a comment.

### 3) Pseudo instructions

#### ① VERSION

- Set a version number.
- A version number should be set in up to 20 characters using the following format.
- VERSION AAAAAAAAAAAAAAAAAAAAAA
- If no entry is made, spaces occupy 20 columns.

#### ② LOWSTOPCOUNT

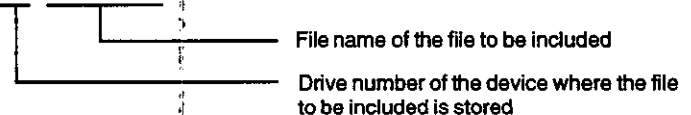
- Set the low-speed scan ladder sequence processing stop count.
  - 0: The compiler sets "1".
  - 1 to 127: Stop count (4 msec per count)
- If no entry is made, the default value of "1" is set.

#### ③ HIGHSEQUENCE

- This indicates the start of a high-speed scan ladder sequence.
- An object file is generated as a high-speed scan ladder up to the ENDP instruction.
- Format: HIGHSEQUENCE ..... ENDP
- If no entry is made, a high-speed scan ladder is not generated.
- This pseudo instruction must always be written in a main file.

#### ④ INCLUDE

- This instruction calls up the files to be included.
- Format: INCLUDE B : LAD.L01



- Entry of a path name preceding the file name of an included file is possible.

```
INCLUDE B:\LPROG\LOWLAD.L01
```

#### ⑤ ENDP

- This indicates the end of a high-speed scan ladder sequence, low-speed scan ladder sequence, conversion data, and message data.
- Format: ENDP
- This pseudo instruction must always be written in a main file.

#### ⑥ LOWSEQUENCE

- This indicates the start of a low-speed scan ladder sequence.
- An object file is generated as a low-speed scan ladder up to the ENDP instruction.
- Format: LOWSEQUENCE ..... ENDP
- If no entry is made, a low-speed scan ladder is not generated.
- This pseudo instruction must always be written in a main file.

⑦ CONVERSION

- This instruction generates the object file as the conversion data in the table.
- Format: CONVERSION ..... ENDP
- If no entry is made, it is regarded as there being no message data.

⑧ MESSAGE

- This instruction indicates the start of setting of message data in the ladder table.
- The instruction generates the object file regarding the data up to ENDP as the message data.
- Format: MESSAGE ..... ENDP
- If no entry is made, it is regarded as there being no message data.

⑨ SYMBOL

- This defines the names for individual coils.
- Definition is possible in up to 8 characters.
- In the display of ladder, the first 5 characters of the specified symbol name are displayed.
- Registration capacity of the symbol names is 5000.
- Format: SYMBOL ..... ENDP

(b) Included files

Files shown in ① to ⑦ below are examples of high-speed scan ladder, low-speed scan ladder, conversion data, and message data that are included in the main file (YELADDER.SRC). Pseudo instructions such as HIGHSEQUENCE, LOWSEQUENCE, CONVERSION, MESSAGE, SYMBOL and ENDP must not be written in the source files ① to ⑦. They must be written in a main file.

① LAD.HI

```
*****  
* High Speed Ladder *  
*****  
LD #10000  
OUT #11000  
  
RTH
```

## ② LAD.L01

```

:*****
:*      Low Speed Ladder (Lead)      *
:*****
LOWSEQUENCE
LD      #14000
INR     #1500
-
OUT     #11010

```

## ③ LAD.L02

```

:*****
:*      Low Speed Ladder2 (Intermediate)  *
:*****
LD      #14056
DST     #1552, #1532,0FFH
-
OUT     #14033

```

## ④ LAD.L03

```

:*****
:*      Low Speed Ladder3 (End)          *
:*****
LD      #10012
OUT     #14500
-
OUT     #14010
RET

```

## ⑤ CONV.DAT

```

:*****
:*      Conversion Table Data          *
:*****
N9000   0H, 1H, 2H, 3H, 4H, 5H, 6H, 7H
-
N9023   0FAH, 0FBH, 0FCH, 0FDH, 0FEH, 0FFH

```

⑥ MESSAGE.DAT

```
*****  
*           Message Data           *  
*****  
N9024 'SPINDLE ALARM'  
N9323 'TROUBLE IN EXTERNAL DEVICE'
```

⑦ SYMBOL.DAT

```
*****  
*           Symbol Name            *  
*****  
#10000 JOG ; Jog  
#79990 OIL ; Coolant
```



## (c) Source files

## ① High-speed scan ladder, low-speed scan ladder source files

- Write the sequence source ladder programs which should be processed at high- or low-speed.
- Although there are no restrictions on start position, the number of lines or the number of columns for entering characters, at least one space must be placed between an instruction and address.

## ② Conversion data source file

- Write the conversion table which is used by macro instruction SUBP007.
- Although there are no restrictions on the data start line or column, at least one space must be placed between the table number and data.
- A table number must begin with "N".
- Delimiter "," is used between data.
- The table numbers that can be used are indicated below.
  - #9000 to #9007: 256 bytes
  - #9008 to #9023: 128 bytes
- In normal format, data are stored in the ladder table as byte data.

N9000 1, 2, 3, 4, 5

N9000[0]	1
[1]	2
[2]	3
[3]	4
[4]	5

- To store word data, place an underscore preceding the numeric values.

N9000 _1, _2, _3

N9000[0]	<u>1</u>
[1]	
[2]	<u>2</u>
[3]	
[4]	<u>3</u>
[5]	

- To store double-word data, place two underscores preceding the numeric values.

N9000    __1,__2

N9000[0]	
[1]	
[2]	1
[3]	
[4]	
[5]	2
[6]	
[7]	

- Entry of the data should be only the necessary data.  
If the number of data to be converted is specified as "5" using the SUBP007 instruction (N9000 '1, 2, 3, 4, 5'), entry of 5 data is necessary, and it is not necessary to enter 256 data.  
Omission of entry is treated as 0H.

### ③ Message data source file

- Write the message data to be used by the macro instruction SUBP023.
- The message data must be within 40 characters.
- The message data must be enclosed by "".
- Although there are no restrictions on the data start line or column, at least one space must be placed between the table number and "".
- A table number must begin with "N".
- The allowable range of the table numbers is indicated below:
  - #9024 to #9323: 40 words
  - #9024: 'Spindle error occurred'

## (d) Recommended source file format

The explanation has been given using examples in which source files are consolidated in one file using the include function. In practical programming steps, the source file is created in several sections to reduce ladder execution/check cycle time. By creating the source files in several sections, necessary correction should be made only for the source file which is involved with errors, thus compiling time can be reduced.

Examples of source file division are indicated below.

## KANKYOU.SRC

```

:*****
:      EnvironmentSetting      *
:*****
VERSION JXSD LADDER

LOWSTOPCOUNT1

```

## LAKDHI.SRC

```

:*****
:      High Speed Ladder      *
:*****
HIGHSEQUENCE
LD      #14000
INR
-
OUT     #11010
ENDP

```

## LADLOW1.SRC

```

:*****
:      Low Speed Ladder (Lead) *
:*****
LOWSEQUENCE
LD      #14000
INR
-
OUT     #11010
ENDP

```

LADLOW2.SRC

```
*****  
* Low Speed Ladder2 (Intermediate) *  
*****  
LOWSEQUENCE  
LD      #14056  
DST     #1530, 1532, 0F F H  
  
OUT     #14033  
ENDP
```

LADLOW3.SRC

```
*****  
* Low Speed Ladder3 (End) *  
*****  
LOWSEQUENCE  
LD      #10012  
OUT     #14500  
  
OUT     #1410  
RET  
ENDP
```

CONV.SRC

```
*****  
* Conversion Table Data *  
*****  
CONVERSION  
N9000   0H, 1H, 2H, 3H, 4H, 5H, 6H, 7H  
  
N9023   0FAH, 0FBH, 0FCH, 0FDH, 0FEH, 0FFH  
ENDP
```

MESSAGE.SRC

```
*****  
* Message Data *  
*****  
MESSAGE  
N9024   'SPINDLE ALARM'  
  
N9323   'TROUBLE IN EXTERNAL DEVICE'  
ENDP
```

SYMBOL.SRC

```
*****  
* Symbol Name *  
*****  
SYMBOL  
N10000 JOG ; Jog  
-  
N19990 OIL ; Coolant  
ENDP
```

---

## 7.3 COMPILER

### 7.3.1 Compiler Operation

By executing the JXLCOMP instruction, the source file which has been created or edited is compiled to generate the object file.

#### (1) Starting the JXLCOMP

The JXLCOMP is started by the following procedure.

```
JXLCOMP file-1 [.SRC]
[file-2 [.OBJ]]
[file-3 [.ERR]] [ENTER]
```

#### (2) Description of Parameters

file-1: Source file name (input)  
file-2: Object file name (output)  
file-3: Error file name (output)  
Entry for the items in [ ] can be omitted.

- If the entry is omitted for file-2 and file-3, a default file name is set.
- If only "JXLCOMP" is input, the guide messages for inputting the parameters are displayed.
- Example: JXLCOMP B : LADTEST [ENTER]  
The LADTEST.SRC file is input and compiled. If an error occurs, LADTEST.ERR file is created. When the source is compiled without errors, LADTEST.OBJ file is output.
- When the include function is used, compilation is required only for the main file. The files included in the main file are compiled automatically.

### 7.3.2 Compiler Error List

If an error occurs during compilation, the compiler outputs the error list file having the extension of ".ERR" with the same main file name as the input file.

It is also possible to designate the file name of the error list file at the start of the JXLCOMP. In this case, the error list is output in the designated file name.

The compiler error information is stored to the error list file.

If the file has the same name as the error list file name, the existing file is deleted when the error list file is generated.

Error list file:

LAD.HI	40	Illegal characters are used
LAD.LO1	33	Invalid operator
LAD.LO1	56	Insufficient number of operands

----- Error message  
 ----- Error line number  
 ----- Error file name

### 7.3.3 Compiler Check Items

The compiler checks the source file for the format whether it is written in the processing permitted format. In addition to this check, it also checks the following items.

#### (1) Instruction Check

##### ① Operand code check

Permitted: LD, LD-NOT, AND ...  
 Not permitted: ABS, XOR-NOT ...

##### ② Operand number check

Permitted: DEC #1001, OFFH ...  
 Not permitted: DEC #1001 ...

##### ③ Operand address designation range check

Permitted: LD #10001 ...  
 Not permitted: LD #10 ...

##### ④ Operand constant designation range check

Permitted: MV1 #1405, 55H  
 Not permitted: MV1 #1405, OFFFFH

---

(2) The number of Setting Characters

For the characters to be set to the ladder table, the compiler checks the number of characters whether it is within the upper and lower limits.

(3) Output Contact Check

- The compiler checks the output addresses of the OUT instruction whether they are all unique.
- It also checks the output contact addresses whether they are within the specified range.

(4) Check on MCR/END and Nest Level

The compiler checks the correspondence between MCR and END, and also the nesting level.

(5) Timer Check

- The compiler checks the range of registers used for timers.
- It also checks the interference in the addresses of the timers (#1700s and #1300s).

(6) Label Check

- The compiler checks the ADR label names for overlapped definition.
- The correspondence between the JMP and ADR is also checked.

(7) STR and AND-STR Check

The compiler checks the correspondence between the STR (STR-NOT) and AND-STR (OR-STR).

(8) SUBP Calling Sequence Check

- The compiler checks the correspondence between the SUBP and PUSH (APSH, TPSH, IPSHD).
- It also checks the correspondence between the SUBP and STR.

(9) Existence Check for RTH and RET

The compiler checks the RTH and RET for the following:

- Only one RTH exists.
- Either RET or RTI exists.



## 7.4 LINKER

The linker reads the object files in the order in which they are designated in the link module designation file and maps the data contained in these files in the executable format that is the same format as in the flash ROM.

### 7.4.1 Object Data and Linker Processing

Linker processing for the data contained in the object files is described below.

#### (1) High-speed Scan Ladder Data (HIGHSEQUENCE Data)

- Execution order of the high-speed scan ladder is determined in the order of link object.  
If the object data are divided into multiple objects, they are stored additionally starting from the first address of the ladder storage area in the order of link object.
- If the high-speed scan ladder data appear after the low-speed scan ladder data, it causes an error.
- The linker executes the max. check for the ladder storage area.
- An error occurs if there is no RTH.
- The linker checks the ADR label names for overlapped definition.

#### (2) Low-speed Scan Ladder Data (LOWSEQUENCE Data)

- Execution order of the low-speed scan ladder is determined in the order of link object.  
If the object data are divided into multiple objects, they are stored additionally starting from the first address of the ladder storage area in the order of link object.
- The linker executes the max. check for the ladder storage area.
- An error occurs if there is neither RET nor RTI.
- The linker checks the ADR label names for overlapped definition.

#### (3) Conversion Table Data (CONVERSION Setting Data)

- The designated message data are stored to the address (N9000 to N9023) corresponding to the variable number.
- An error occurs if the same variable data exist in more than one object file.

#### (4) Message Table Data

- The designated message data are stored to the address (N9024 to N9323) corresponding to the variable number.
- An error occurs if the same variable data exist in more than one object file.

(5) Version Number Data (VERSION Setting Data)

- The linker stores the version number data to the designated address.
- An error occurs if a version number is defined in more than one object file.

(6) Low-speed Scan Ladder Stop Count (LOWSTOPCOUNT Setting Data)

- The linker stores the low-speed scan ladder stop count to the designated address.
- An error occurs if the low-speed scan ladder stop count is defined in more than one object file.

## 7.4.2 Linker Operation

The linker generates the link binary file from the object file output from the compiler by using the JXLLINK instruction.

(1) Link Module File

It is necessary to create the link module file before starting the JXLLINK. The object files to be linked are designated by this file.

(a) Link module file name

FILE1.LNK

File name can be assigned as required. However, the extension must be ".LNK".

(b) Link module file format

- Designate all object files to be linked as indicated below.
- There are no restrictions on the start line/column for the entry of characters. (The maximum number of characters per line is 80 including the path name.)
- Designation of the link module file must be made in one line, within 80 characters including path name.
- The high-speed and low-speed scan ladders are executed in the order they are designated in this file.

YELAD.LNK

```
KANKYOU.OBJ
LADHI.OBJ
LADLOW1.OBJ
LADLOW2.OBJ
LADLOW3.OBJ
MESSAGE.OBJ
DATA.OBJ
SYMBOL.OBJ
```

## (2) Starting the JXLLINK

```
JXLLINK FILE1.LNK [FILE2] [ENTER]
```

- Description of parameters:

FILE1: Link module designation file name (input)

FILE2: Binary file name (output)

Entry for the items in [ ] can be omitted.

If the entry is omitted for FILE2, the same file name as FILE1 is assigned.

- If only "JXLLINK" is input, the guide messages for inputting the parameters is displayed.

### 7.4.3 Linker Output File

The result of link by the execution of "JXLLINK" is generated in one output file.

Example: JXLLINK YELAD.LNK[ENTER]

Output file

YELAD.BIN      Ladder execution file

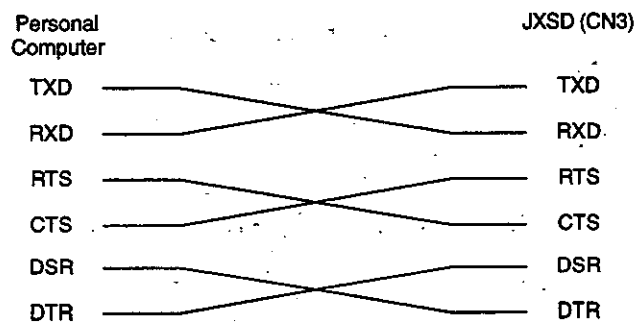
## 7.5 REMOTE CONTROLLER OPERATION

By using the JXPCCOM, communication between the personal computer and PLC is possible.

### 7.5.1 Connecting the JXSD to PLC

Connect the CN3 port in the JXSD to the standard RS-232C port in the PLC. It is not necessary to use the SWITCH command to set the communication parameters; they are automatically set at the start up of the JXPCCOM.

Connection between Personal Computer and JXSD (CN3):



## 7.5.2 Starting the Remote Controller

JXPCCOM [ENTER]

By the entry as indicated above, the start-up screen is displayed and the JXPCCOM waits for the key entry.

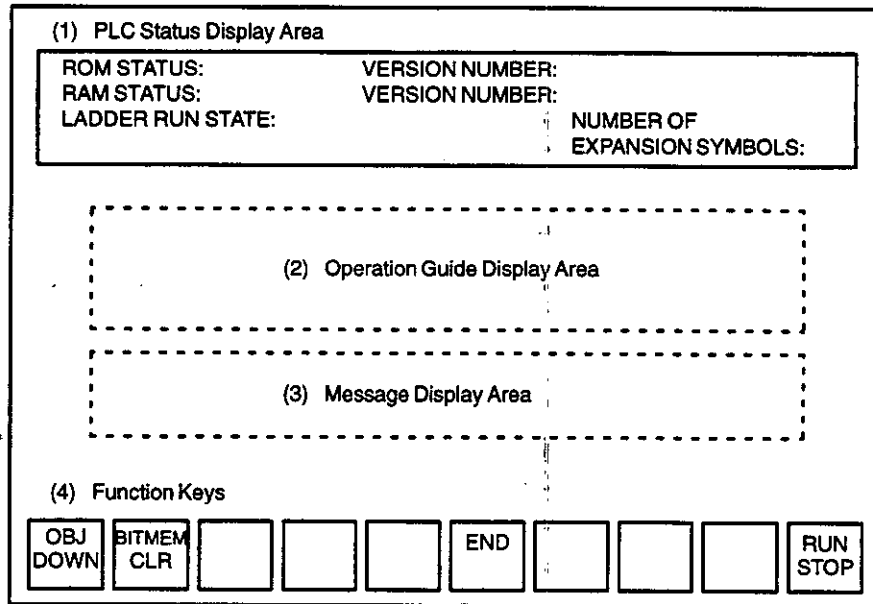


Fig. 7.3

---

### 7.5.3 Description of Screen Display Information

#### (1) PLC Status Display Area

In this area, the status of PLC is displayed at the start-up of JXPCCOM and at the completion of operation.

##### ① ROM STATUS

VALID: Ladder program exists in the flash ROM.

INVALID: Ladder program does not exist in the flash ROM.

##### ② RAM STATUS

VALID: Ladder program exists in the RAM area used for editing.

INVALID: Ladder program does not exist in the RAM area used for editing.

##### ③ VERSION NUMBER

The version number of the stored data is displayed only when the contents of editing RAM are correct.

##### ④ LADDER RUN STATE

STOP: Ladder program is stopped.

RUN: Ladder program is run.

##### ⑤ NUMBER OF EXPANSION SYMBOLS

The number of expansion symbol cases stored in the ladder edit RAM area.

#### (2) Operation Guide Display Area

Guide messages for key entry are displayed when key operation becomes necessary due to pressing a function key.

#### (3) Message Display Area

The results of operation having been conducted according to the guide messages are displayed.

#### (4) Function Keys

The function keys are used to select the operation.

① OBJ TRANSFER

The operation mode enters the downloading of the ladder object files to the PLC.

② BITMEM CLR

The bit memory is cleared to "0".

③ END

The operation quits the PCCOM.

④ EDIT

The operation mode enters the ladder edit mode.

⑤ RUN STOP

The operation mode enters the mode for switching run/stop of the ladder.

## 7.5.4 Operation of Remote Controller

### (1) Downloading the Object Files

Use the procedure indicated below to download the object files.

- ① Press the [OBJ DOWN] function-key.

The following screen is displayed.

ROM STATUS: Mounted	VERSION NUMBER: JXSD Ladder 1
RAM STATUS: Valid	VERSION NUMBER: JXSD Ladder 2
LADDER RUN STATE: Stopped	NUMBER OF EXPANSION SYMBOLS: 0

Input Binary File Name

Name:

Execute with [ENTER] (Cancel with [ESC])

(4) Function Keys

OBJ DOWN	BITMEM CLR				END				RUN STOP
-------------	---------------	--	--	--	-----	--	--	--	-------------



- ② Designate the file name of the object file to be downloaded.

The file which has been output by the linker can be designated.

Example: B : LADTEST.BIN

It is not allowed to omit the extension “.BIN” in the designation of a file name.  
Press the [ESC] key to return the screen to the start-up screen.

ROM Status: Mounted	Version Number: JXSD Ladder 1
RAM Status: Valid	Version Number: JXSD Ladder 2
Ladder Run State: Stopped	Number of Expansion Symbols: 0

Select the write data

Ladder Program  
Ladder Table  
Symbol Table

Select and specify [SPACE] with the cursor keys [↑] and [↓].  
Execute with [ENTER] and cancel with [ESC].

Function Keys

OBJ DOWN	BITMEM CLR				END				RUN STOP
-------------	---------------	--	--	--	-----	--	--	--	-------------

- ③ Designate the data to be downloaded.

Move the cursor to the data to be downloaded by using the cursor up/down keys and press the space bar.

The data displayed in highlighted characters are downloaded.

Selection by the space bar is toggle.

The data actually downloaded in response to the selection of the data name on the screen are indicated below.

- Ladder Program Specification
  - New/old ladder classification
  - Low-speed scan ladder stop count
  - High-speed scan ladder
  - Low-speed scan ladder
  - Conversion ladder
  - Label ladder
  - Version number
- Ladder Table Specification
  - Conversion table
  - Message table
- Symbol Table Specification

- 
- ④ In the first downloading operation, all of the data that are used for the operation must be transmitted. If conversion data and message data are used, it is necessary to transmit the ladder table in addition to the ladder program.

The operation procedure to be followed when transmitting the ladder table in addition to the ladder program is indicated below.

Move the cursor to "Ladder Program"  
(The "Ladder Program" will be highlighted.)



Move the cursor to "Ladder Table".



Press the space bar.  
(Both the "Ladder Program" and "Ladder Table" are highlighted.)



Press the [ENTER] key.  
(Download starts. The following message appears in the message display area.  
"SENDING OBJECT")



Downloading is complete if the following message is displayed.

"ENDED NORMALLY"

- ⑤ If the ladder is modified after it has been downloaded, only the modified data require downloading.

If the contents of the ladder program are modified while the conversion data and the message data remain unchanged, only the ladder program should be downloaded and it is not necessary to download the ladder table.

- ⑥ After the completion of downloading, RAM Status information and the information of Number of Expansion Symbols will be changed.

## (2) Run/Stop of Ladder

By pressing the [RUN STOP] function key, operation starts. If this function key is pressed after the execution of the ladder has started, it is stopped. Conversely, when it is pressed while the ladder execution is stopped, it starts.

After the execution, the Ladder Run State information is changed.

The procedure to run the ladder while the ladder stop status is indicated below.

- ① Press the [RUN STOP] function-key.

The following screen is displayed.

ROM STATUS: Mounted	VERSION NUMBER: JXSD Ladder 1
RAM STATUS: Valid	VERSION NUMBER: JXSD Ladder 2
LADDER RUN STATE: Stopped	NUMBER OF EXPANSION SYMBOLS: 0

Run ladder

Run with [ENTER] (Cancel with [ESC])

OBJ DOWN

BITMEM CLR

END

RUN STOP

- ② Press the [ENTER] key.

Operation is complete if "ENDED NORMALLY" is displayed in the message display area. The Ladder Run State information changes from "Stopped" to "Run".

The screen returns to the start-up screen when the [ESC] key is pressed.

### (3) Clearing the Bit Memory

The operation starts in response to pressing the [BITMEM CLR] function-key. After the execution, the bits of the I/O signals and the internal relays are cleared to "0". The operation procedure is indicated below.

- ① Press the [BITMEM CLR] function-key.

The following screen is displayed.

ROM STATUS: Mounted		VERSION NUMBER: JXSD Ladder 1	
RAM STATUS: Valid		VERSION NUMBER: JXSD Ladder 2	
LADDER RUN STATE: Stopped		NUMBER OF EXPANSION SYMBOLS: 0	

Clear bit memory to 0.

Execute with [ENTER] (Cancel with [ESC])

OBJ DOWN	BITMEM CLR				END				RUN STOP
-------------	---------------	--	--	--	-----	--	--	--	-------------

- ② Press the [ENTER] key.

Operation is complete if "ENDED NORMALLY" is displayed in the message display area.

## 7.6 LIST OF ERROR MESSAGES AND WARNING MESSAGES

### 7.6.1 Error Messages

1-line characters over  
 Illegal character is used.  
 Over the nest of source-file.  
 Illegal character is used instead of pseudo-instruction.  
 A pseudo-instruction is used duplicatedly.  
 'ENDP' cannot be found.  
 Characters of a word is too long.  
 Invalid operator.  
 Object-file memory size over.  
 Operand of an instruction is not enough.  
 Operand-address is not correct.  
 Operand-byte-data is not correct.  
 Operand-word-data is not correct.  
 Label define error.  
 SUBP number is not correct.  
 Label define error.  
 Total number of defined-label exceeds 256.  
 Stop-count-setting-range is not correct.  
 Table-number define error.  
 Table-number-setting-range is not correct.  
 Character data define error.  
 Character data range define error.  
 Character data lines over.  
 Variable number error.  
 Out instruction address range over.  
 Timer-register range error.  
 Number of MCR & END is unmatch.  
 Byte data define error.  
 Word data define error.  
 Data range define error.  
 Number of Operands are too large, or Include valid characters.  
 Nest of MCR over.  
 There is no version number character.  
 Duplicatedly define of label-characters.  
 JUMP & CALL are used too much.  
 Duplicatedly use of variable number.  
 Stop-count of low-speed-scan must be defined.  
 Symbol-case-number exceed 6500.  
 SUBP calling sequence error.  
 Number of SUBP & PUSH is unmatch.  
 Nest of STR over.  
 Number of stack instruction by STR is not correct.  
 SUBP parameter error.  
 Operand double word data error.  
 Nesting filr open error.

### 7.6.2 Warning Messages

Output contact of OUT-instruction is defined duplicatedly.  
 Symbol-case-number exceed 5000.



# 8

---

## ONLINE EDITING

Chapter 8 describes online editing operation.

8.1	OUTLINE OF ONLINE EDITING .....	8 - 2
8.2	FUNCTION TREE AND DISPLAY SCREENS .....	8 - 4
8.3	LADDER DISPLAY FUNCTION .....	8 - 6
8.4	NET EDITING FUNCTION .....	8 - 10
8.5	TABLE EDIT FUNCTION .....	8 - 38
8.6	INPUT/OUTPUT FUNCTION .....	8 - 43
8.7	SEQ STS (SEQUENCE STATUS) FUNCTION .....	8 - 47
8.8	LIST OF MESSAGES .....	8 - 51

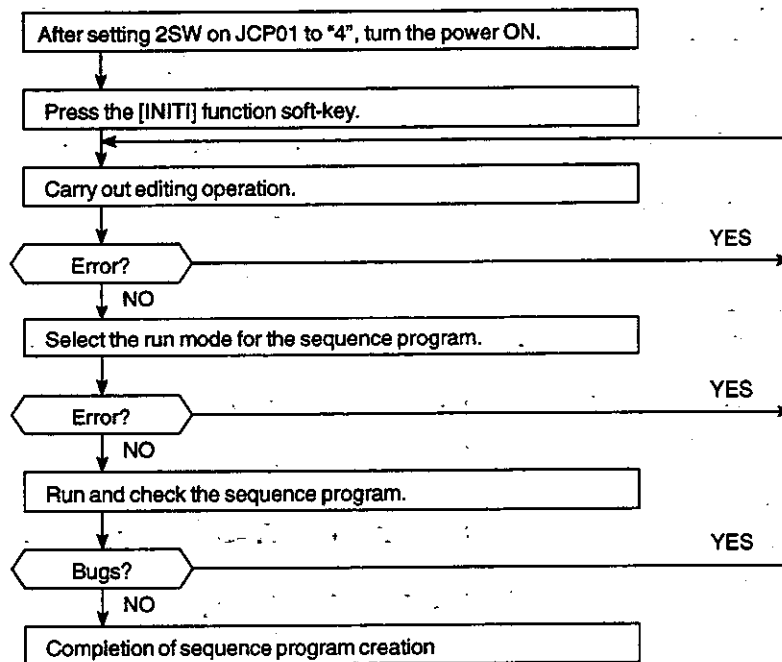


## 8.1 OUTLINE OF ONLINE EDITING

It is possible to edit the sequence ladder directly at the NC operation panel instead of using a personal computer. Sequence ladder edit operation procedure differs slightly depending on whether the sequence ladder is newly created or the sequence ladder is created based on the existing sequence ladder.

### 8.1.1 Creating a Sequence Program Newly

The sequence program development flow chart for newly creating a sequence program is described below.



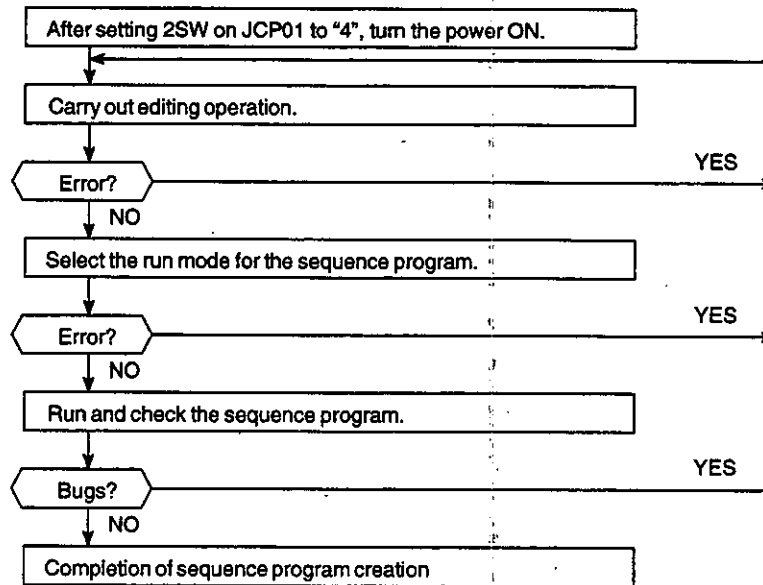
The following processing is executed when the [INITI] function soft-key is pressed.

- ① "TEST" is set for "VERSION".
- ② "3" is set for "LOW-SPEED STOP COUNT".
- ③ Sequence area is cleared and the following program is inserted automatically from the beginning.
  - HIGHSEQUENCE
  - RTH
  - ENDP
  - LOWSEQUENCE
  - RET
  - ENDP
- ④ The message data area is cleared.
- ⑤ The conversion data area is cleared.
- ⑥ The symbol data area is cleared.



### 8.1.2 Creating a Sequence Program by Modifying the Existing Sequence Program

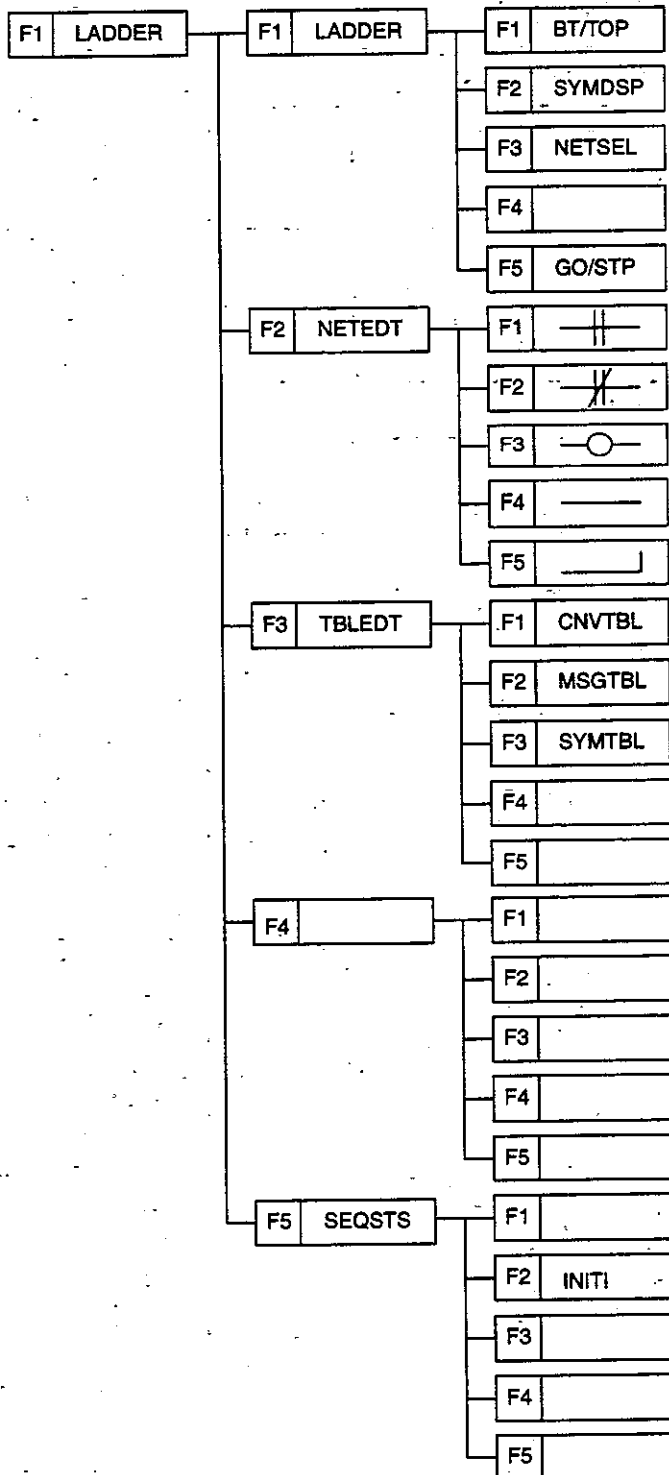
The operation steps to be followed when creating a sequence program by modifying the existing sequence program are described below.



## 8.2 FUNCTION TREE AND DISPLAY SCREENS

### 8.2.1 Function Tree

The tree of functions called by selecting the ladder job are indicated below.



Function Display	Conditions for Calling Up the Function
Always displayed	In sequence program debug mode
While the JPXCCOM is not connected.	While the JPXCCOM is not connected with sequence program stopped.

### 8.2.2 Ladder Display Screen

An example of the ladder display screen is shown in Fig. 8.1.

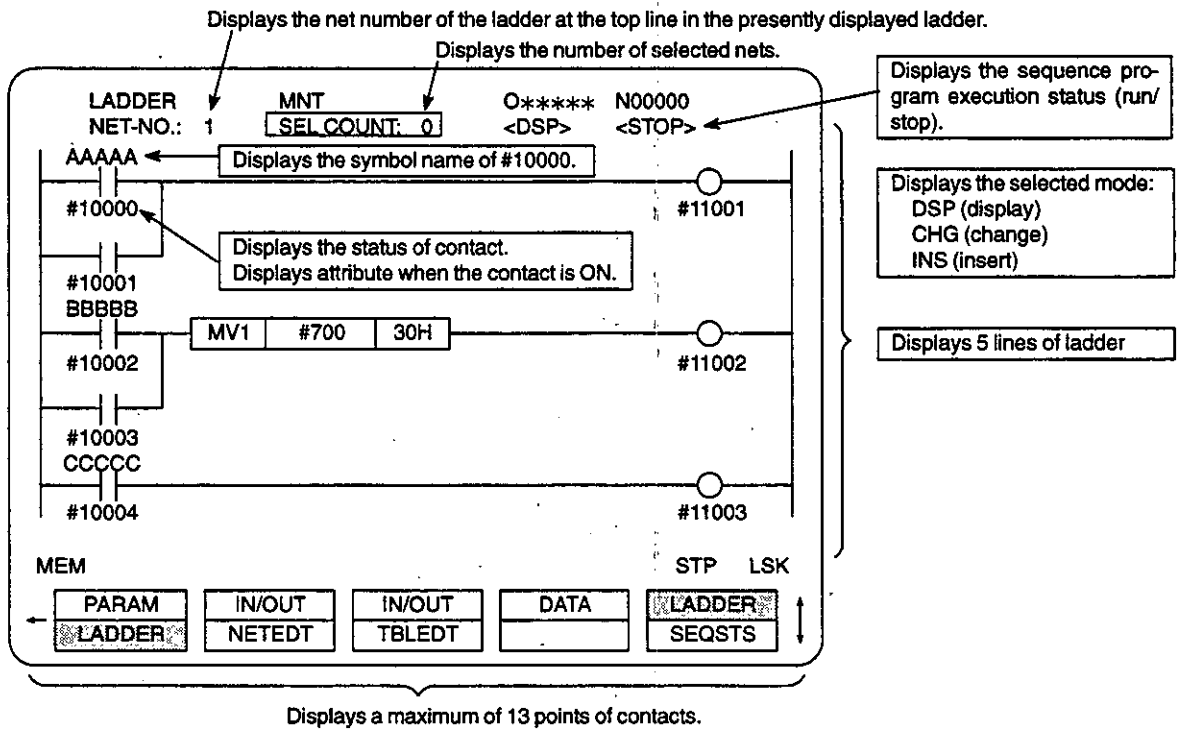


Fig. 8.1 Ladder Display Screen

## 8.3 LADDER DISPLAY FUNCTION

The ladder display function displays the sequence program stored in the NC in ladder form. One line of ladder can contain a maximum of 13 contacts; the maximum number of contacts that can be displayed in a sequence ladder is 100. There are no limits on the number of lines as long as the maximum number of contacts is within the limit.

Each instruction indicated in Table 8.1 is counted to have the specified number of contacts.

Table 8.1 List of Instructions That Have Multiple Number of Contacts

Instruction	Number of Contacts	Instruction	Number of Contacts	Instruction	Number of Contacts		
INR	2	ANR	3	CMRW	2		
DCR	2	ORR	3	CORW	3		
CLR	2	XRR	3	CPRW	3		
CMR	2	CPR	3	MVIW	3		
ADI	3	COR	3	DSTW	3		
SBI	3	MOV	3	JMP	2		
ANI	3	DST	3	ADR	2		
ORI	3	DIN	4	IPSH	2		
XRI	3	ADC	3	APSH	2		
DEC	3	ADDW	3	PUSH	2		
COI	3	SUBW	3	TPSH	2		
CMP	3	MULW	3	IPSHD	3		
CPI	3	DIVW	3	TMR	3		
MVI	3	INRW	2	TIM	3		
ADD	3	DCRW	2	SUBP	3		
SUB	3	CLRW	2				

### 8.3.1 BT/TOP (Bottom/Top) Function

The function searches and displays the top line or the bottom line of the sequence ladder. This key is a toggle.

### 8.3.2 SYM DIS (Symbol Display) Function

This function displays the symbol name of the contact which is set by the symbol pseudo instruction "SYMBOL".

This key is a toggle - each time the key is pressed, the symbol display is given and cleared.

### 8.3.3 NET SEL (Net Selection) Function

The function displays only the selected nets. This function is used when the nets to be referenced are separated from each other so that displaying them on one display page is impossible.

This function is executed in the following procedure.

- ① On the sequence ladder display screen, press the [WR] key when the net number to be selected is displayed.
- ② The selection symbol (see Fig. 8.2) is displayed and the net is set in the selected status.

Selection is possible for up to ten nets. If an attempt is made to select the net exceeding this limit, a warning message is displayed.

“SELECTION OVER”

- ③ Press the [NET SEL] function soft-key.
- ④ While the selected nets are collected, the message “COLLECTING” is displayed.

After the completion of net selection, the ladder of the collected nets is displayed and the message is cleared.

- ⑤ Press the [NET SEL] function soft-key once again, and the screen returns to the normal sequence ladder display screen.



The selected status of the nets is cleared when the power is turned OFF.

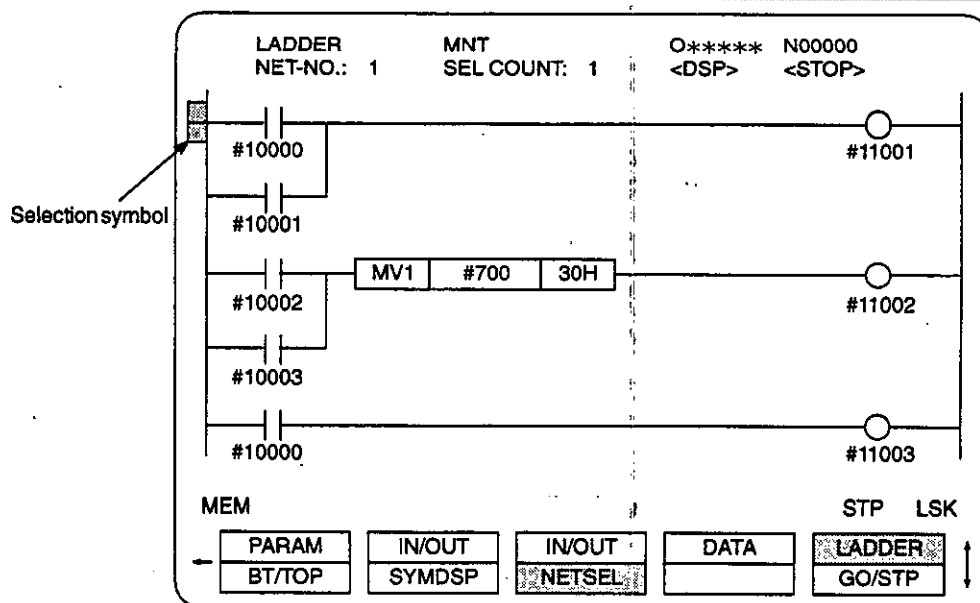


Fig. 8.2

### 8.3.4 GO/STP (Run/Stop) Function

By using the [GO/STP] function soft-key, the sequence program execution (run, stop) can be controlled.

At the start of sequence program execution, correctness of the sequence program is checked. During this check, the message "LADDER CHECKING" is displayed. If an error is found during this check, the sequence program cannot be executed even if the [GO/STP] function soft-key is depressed. The content of the error found during the check is displayed by the corresponding warning message.

Note that the [GO/STP] function soft-key is not valid while the NC is running.

After the start of the sequence program; message <EXEC> is displayed on the screen.

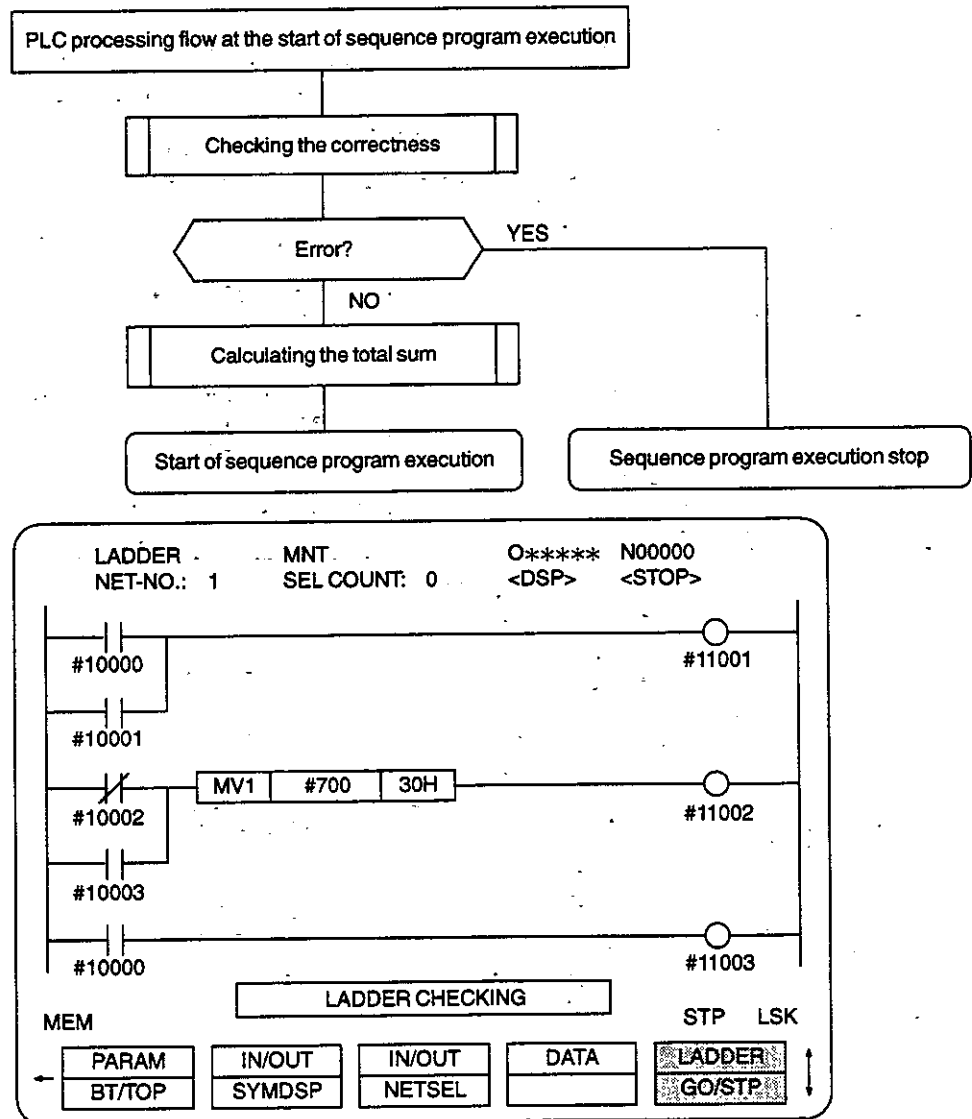


Fig. 8.3

## (1) Checking the Correctness of Sequence Program

Before the execution of the sequence program, its correctness is checked.

Table 8.2 Sequence Program Correctness Check

No.	Check Items	Description
1	JMP - ADR correspondence check Warning message: NO JMP-ADR	There must be an ADR entry corresponding to JMP instruction. A pair of JMP and ADR designation must exist within the program of the same processing type (high-speed scan or low-speed scan).
2	SUBP calling up sequence check Warning message: SUBP CALL ERROR	The order and argument of the following are checked: APSH, TPSH, IPSH, IPSHD, and PUSH The third argument of SUBP023 must be either "1" or "2".

If an error is found in the correctness check, the sequence program is not executed.

When the execution status shifts from "stop" to "run", the total sum value is created in addition to the correctness check. The total sum value can be confirmed by the SEQ information function.



## 8.4 NET EDITING FUNCTION

When the [NET EDT] function soft-key is pressed after placing the sequence program in the stop status, the pop-up menu showing the net edit items is displayed as shown in Fig. 8.8.

The objective of the net edit function is the net number (NET-NO) displayed at the upper left part in the screen. The net number indicates the net presently displayed on the screen.

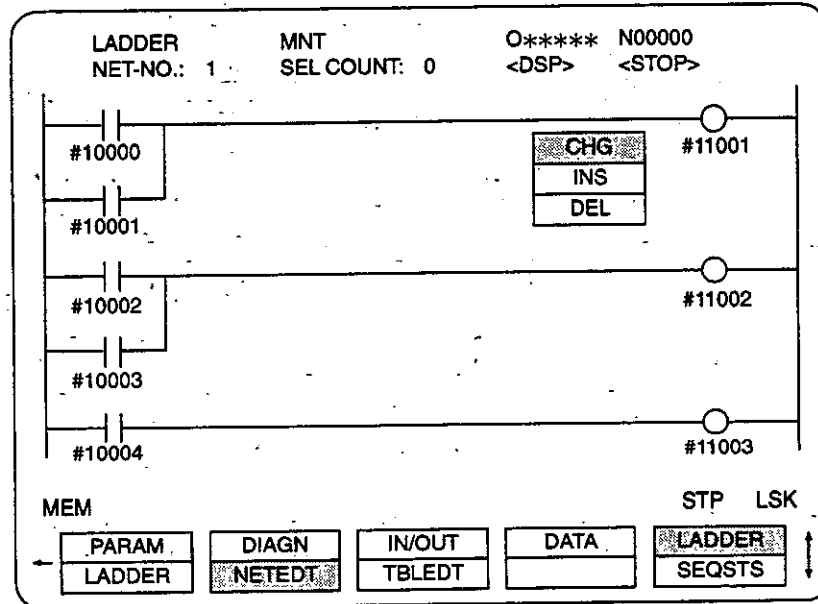


Fig. 8.4 Net Edit Function Screen



1. During the execution of a sequence program, if the objective of editing is either the RTH or RET instruction, editing (change, insert, delete) is not allowed. If the [NET EDT] function soft-key is depressed during the execution of a sequence program, the following warning message is displayed.

"LADDER CHECKING"

2. The contact status is not displayed during net editing.



### 8.4.1 Selection of Edit Mode

When selecting an edit item, either a net edit key or an action key on the NC operation panel can be used.

#### (1) Selecting the Edit Mode by Net Edit Key

Select the net edit mode from the menu items in the pop-up menu.

Net edit items:

CHG (change): This mode is used to change the net.

INS (insert): This mode is used to insert a new net.

DEL (delete): This mode is used to delete an existing net.

#### (2) Selecting the Edit Mode by Action Key on the NC Operation Panel

Selection of an edit mode is possible without displaying the pop-up menu screen. To clear the pop-up menu from the screen, press the [NET EDT] function soft-key once again.

Without using the pop-up menu, change, insert, and delete modes can be selected by using the action keys on the NC operation panel.

[ALT] key: This selects the change mode.

[INS] key: This selects the insert mode.

[ERASE] key: This selects the delete mode.



(3) Description of the Edit Modes

(a) Change mode

In this mode, a selected net can be changed. When the change mode is selected, change is possible for one selected net. If the change mode is selected while the net number "1" is displayed on the screen (NET-NO: 1) as in Fig. 8.4, the screen displays only the ladder of net number "1" as shown in Fig. 8.5. The shaded block in this screen indicates the edit cursor (blinking). When the change mode screen is displayed first, the cursor appears at the contact displayed in the upper left area and the mode indication changes to <CHG>.

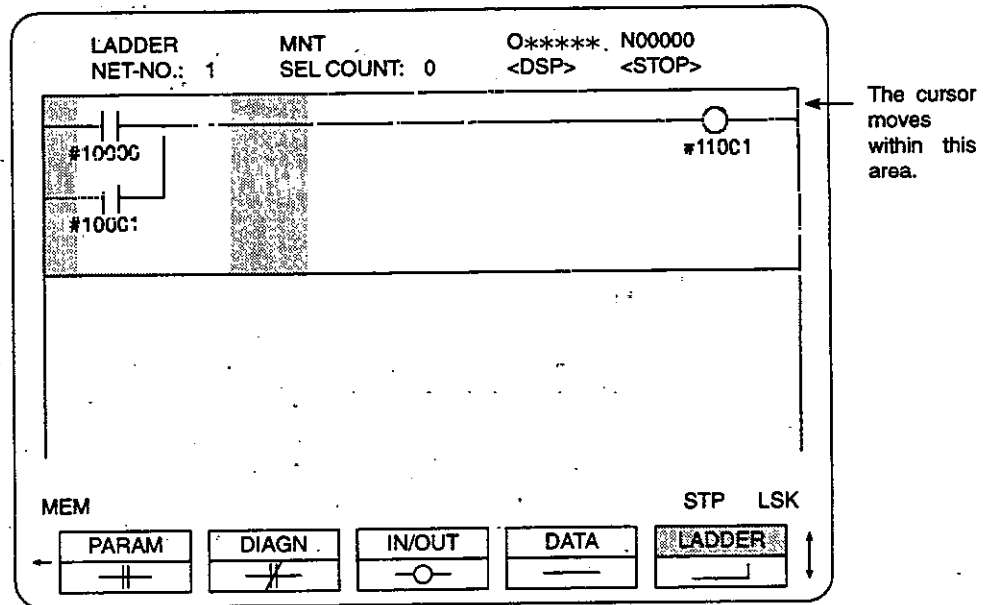


Fig. 8.5

## (b) Insert mode

In this mode, a new net is inserted preceding the selected net. When the insert mode is selected, the edit screen is opened for insertion operation. Since a new net is inserted, the opened screen does not show the ladder and the mode indication changes to <INS>.

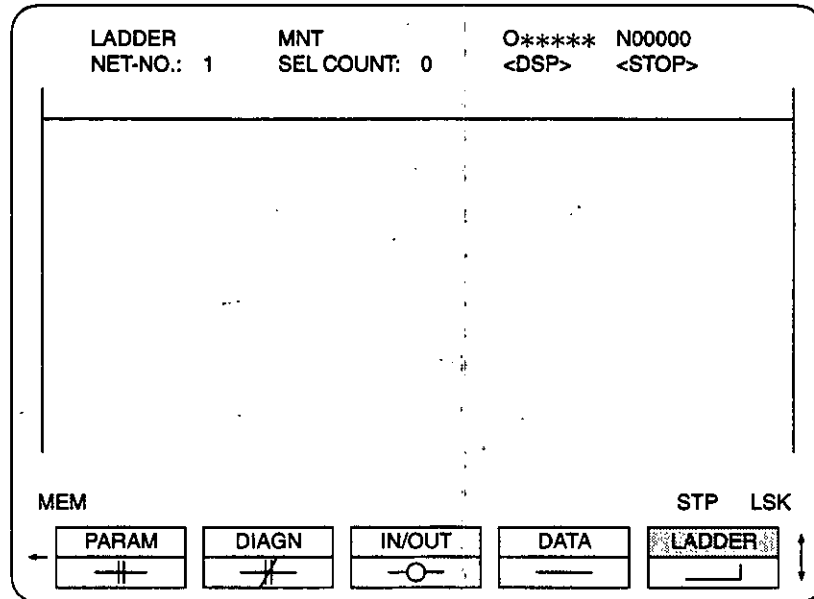


Fig. 8.6

(c) Delete mode

In this mode, the selected net is deleted. At the mode display area, <DEL> is displayed.

The "DELETING" is displayed.

Figs. 8.7 and 8.8 show the delete mode screens before and after the deletion of a net. Upon completion of deletion, the new ladder chart is displayed with the following message displayed.

"DELETION COMPLETED"

Operation for deletion:

Select the contact to be deleted by moving the cursor onto it, select DEL pop-up menu and press the [WR] key. This deletes the selected contact.

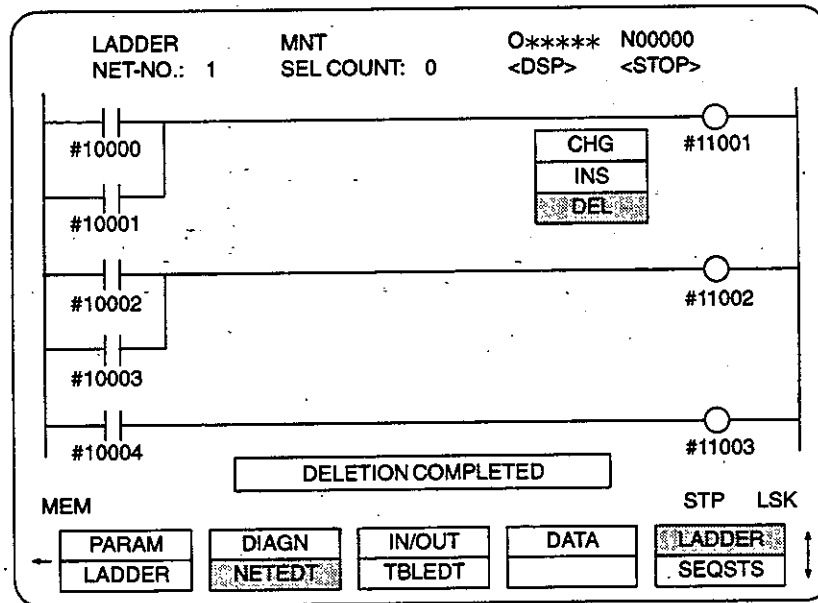


Fig. 8.7 Delete Mode Screen (Before Deletion)

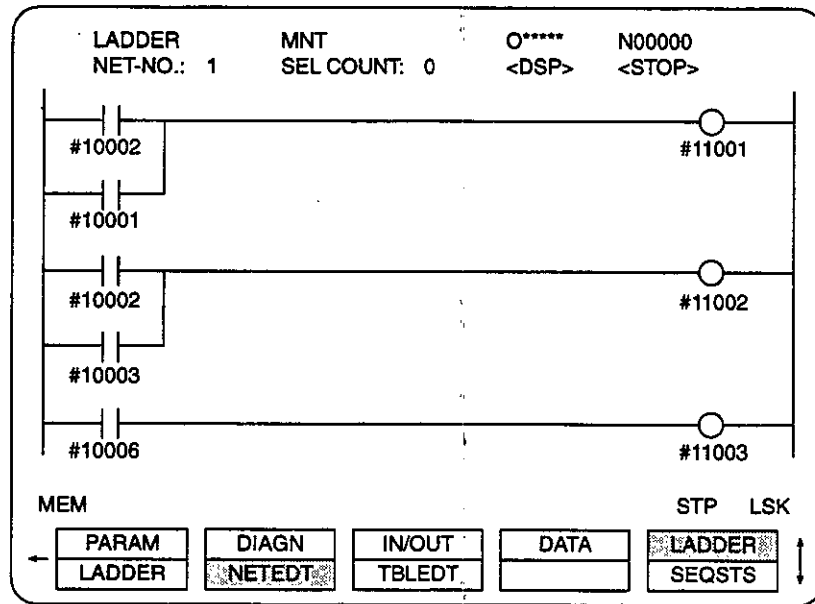


Fig. 8.8 Ladder Display Screen (After Deletion)

#### 8.4.2 Keys Used for Editing the Ladder

The net edit screen is displayed when either the insert or the delete mode is selected. On the net edit screen, editing is possible for one net.

A contact instruction can be input by using a function soft-key. The same restrictions as applied for displaying the ladder are also applied for editing the ladder: a maximum of 13 contacts in one line, no limitations on the number of lines in the vertical direction, and the maximum of 100 contacts. If the number of contacts exceeds 100, the following warning message is displayed.

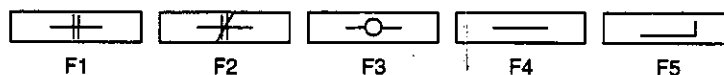
“CONTACT OVER”

If the number of contacts exceeds the limit, reduce the number by dividing the net into two or more nets or some other appropriate method and create a net again.

Register instructions other than contacts can be input in the conversational mode by entering the first character of the instruction. To input contacts and register instructions, move the cursor to the position where the input is possible. If an attempt is made to input them at a position where input is not possible, the following warning message is displayed. In this case, move the cursor to the position where the input is permitted and input them again.

“INPUT ERROR”

In the net edit mode (change or insert), the following function soft-keys are provided as the secondary function soft-keys.




---

The functions of the keys are indicated below.

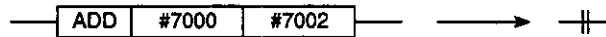
(1) Cursor Keys


- ① Cursor up key: Moves the cursor up by one line.  
If the cursor is in the top line in the net, the cursor does not move even if the cursor up key is pressed.
- ② Cursor down key: Moves the cursor down by one line.  
If the cursor is in one line below the bottom line in the net, the cursor does not move even if the cursor up key is pressed.
- ③ Cursor right key: Moves the cursor right to the next device.  
If the cursor right key is pressed when the cursor is at the right hand end position, the cursor moves to the left hand end position.
- ④ Cursor left key: Moves the cursor left to the next device.  
If the cursor left key is pressed when the cursor is at the left hand end position, the cursor moves to the right hand end position.

## (2) Function Soft-keys

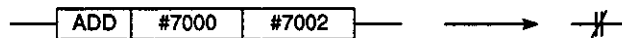
- ①  : Replaces the block where the cursor is positioned with the NO contact.


If this key is pressed when the cursor is positioned on the block which includes a register instruction, the remaining block is deleted as indicated below.



- ②  : Replaces the block where the cursor is positioned with the NC contact.

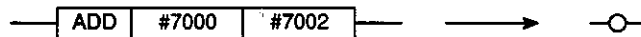
If this key is pressed when the cursor is positioned on the block which includes a register instruction, the remaining block is deleted as indicated below.




- ③  : Replaces the block where the cursor is positioned with the OUT coil.

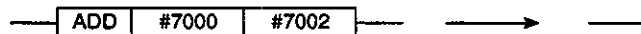
If this key is pressed when the cursor is positioned on the block which includes a register instruction, the remaining block is deleted as indicated below.

Note: An OUT coil can be input only at the 13th contact. If it is input at any other position, an error occurs and "INPUT ERROR" message is displayed.




- ④  : Replaces the block where the cursor is positioned with a horizontal line "——".

If this key is pressed when the cursor is positioned on the block which includes a register instruction, the remaining block is deleted as indicated below.



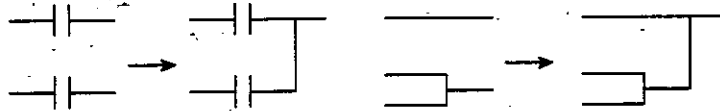
If this key is pressed when the cursor is positioned on the block of the horizontal line "——", the horizontal line is deleted.



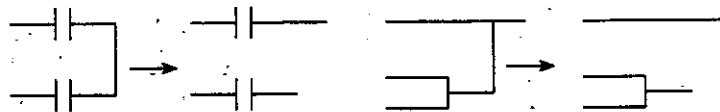
- ⑤  : Draws a vertical line on the block where the cursor is positioned and also on the block above this block.

How the vertical line is drawn is explained below:

If a vertical line does not exist in the upper right of the cursor, a vertical line is drawn.



If a vertical line exists in the upper right of the cursor, the vertical line is deleted.



Display method of a vertical line varies depending on the cursor position and the information of the lines drawn above and below the cursor.

For details, refer to 8.4.4, "Inputting Vertical and Horizontal Lines".

### (3) Other Key

- ① ERASE: Deletes the instruction on which the cursor is positioned.



### 8.4.3 Inputting Contacts

#### (1) Inputting Contacts

- ① When a function soft-key representing a contact is pressed, the message asking the input of the contact (switch) number is displayed as shown in Fig. 8.9.

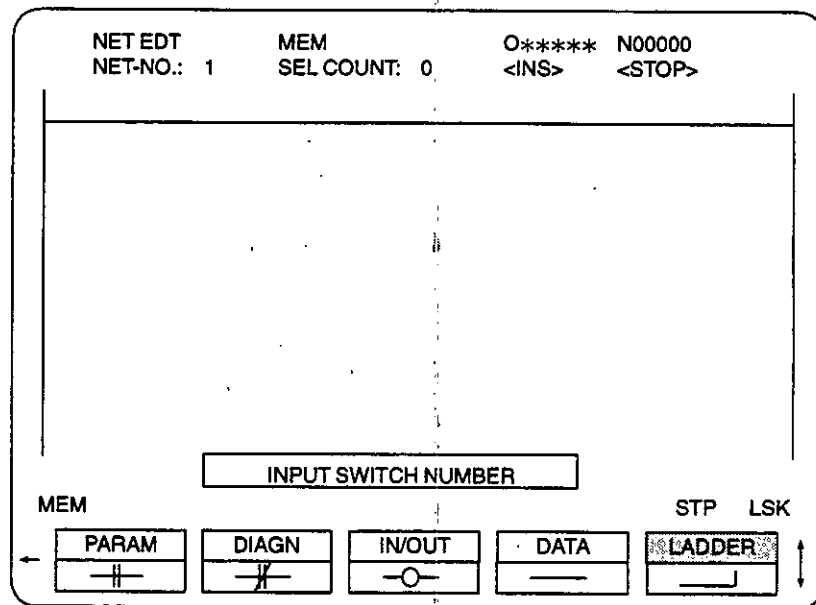





Fig. 8.9 Contact Input Screen

- ② Input the contact number.
- ③ The relationship between the contact instruction and the registers that can be input is indicated below.

	: #1000 to #1063, #3500 to #3699, #7000 to #7999
	: #1400 to #1699, #1800 to #2999
	: #3000 to #3159, #7000 to #7999, #1100 to #1163 #1200 to #1299, #1400 to #1699, #1800 to #2999

If a contact number which does not correspond to the designated contact instruction is input, an error occurs and the following warning message is displayed.

“INPUT ERROR”

If this warning message is displayed, check the contact number again and input the correct one.

(2) Example of Contact Input

- ① When any of function soft-keys f1, f2, or f3 is pressed, the screen gives the message requiring the input of the contact number. Key-in a contact number and press the [WR] key to input the contact number.

When inputting a contact number, it is not necessary to input “#”.

Examples:

10000 [WR]: Correct input

#10000 [WR]: Incorrect input (input error)

- ② When a contact number is input correctly, the highlighted function soft-key returns to the normal display.

To cancel the input during key operation, press the [RST] key.

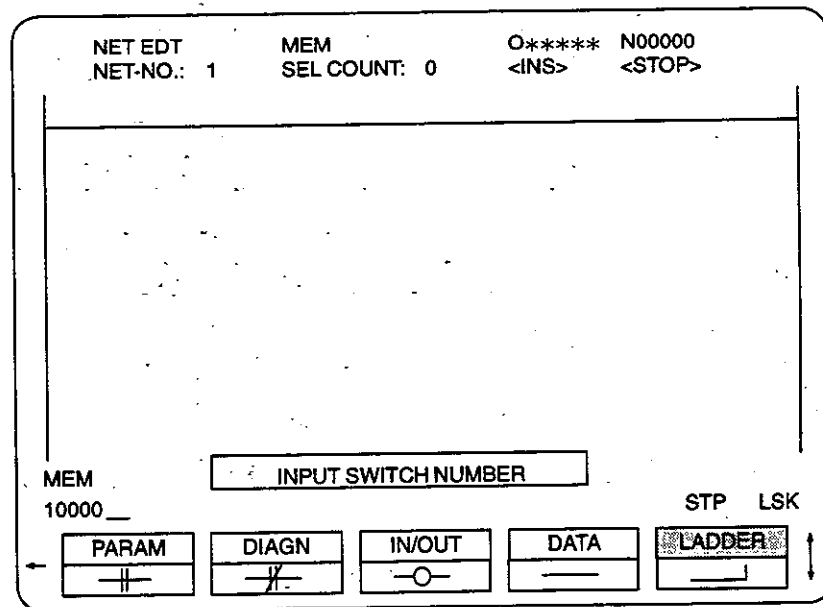


Fig. 8.10 Contact Input Screen

- ③ By the key operation of [1][0][0][0][0][WR], the LD instruction is input and the contact number is input correctly. The cursor position is not changed before or after the input of the LD instruction.

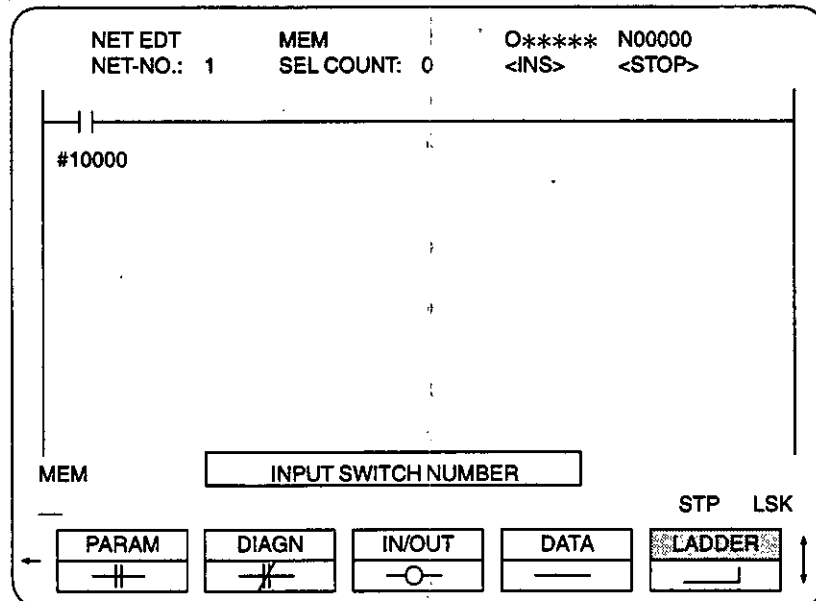


Fig. 8.11 Screen Display after Inputting LD Instruction (#10000)

#### 8.4.4 Inputting Vertical and Horizontal Lines

By using the function soft-keys [ — ] or [ — ], a vertical line or a horizontal line can be input at the cursor position.

When inputting a vertical line, the content of line display varies depending on the cursor position and the instructions existing around the cursor. How the vertical line is input is shown below.

After the correct input of the vertical or horizontal line, the input instruction is reflected to the ladder chart. The cursor position is not changed before or after the input of the vertical/horizontal line.

The example of the screen below shows the vertical line input operation.

Since a horizontal line does not exist in the upper right of the cursor, the vertical line is drawn upward.

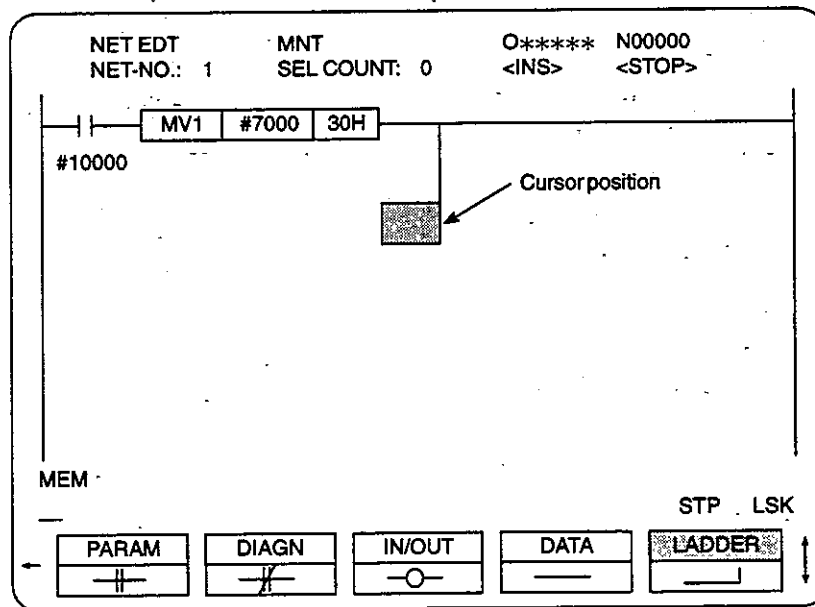


Fig. 8.12 Drawing a Vertical Line

### 8.4.5 Inputting Register Instructions

By keying in the first letter of a register instruction, pop-up menu for the selection of the register instruction appears on the screen.

#### (1) Items Given in Pop-up Menu

The items displayed in the pop-up menu are indicated in Table 8.3. In response to the character keyed-in to the key buffer area, the corresponding list of register instructions is displayed in the pop-up menu. Select the required instruction by moving the cursor on it and press the [WR] key. The required register instruction can be written in this manner.

Table 8.3 List of Register Instructions Displayed in Pop-up Menu

Keyed-in Character	Corresponding Register Instructions Displayed in Pop-up Menu	Quantity
A	ANI, ADI, ADD, ANR, ADC, ADDW, ADR, APSH	8
C	CLR, CMR, COI, CMP, CPI, CPR, CIR, CLRW, CMRW, CORW, CPRW	11
D	DCR, DEC, DST, DIN, DIVW, DSTW, DCRW	7
I	INR, INRW, IPSH, IPSHD	4
M	MVI, MOV, MULW, MVIW, MCR	5
S	SUB, SBI, SUBW, SET, SUBP3, SUBP4, SUBP5, SUBP6, SUBP7, SUBP9, SUBP11, SUBP14, SUBP17, SUBP18, SUBP23, SUBP25, SUBP27, SUBP31, SUBP32, SUBP34, SUBP35, SUBP36, SUBP37, SUBP38, SUBP39, SUBP40	26
T	TIM, TMR, TPSH	3
O	ORR, ORI	2
X	XRI, XRR, XOR, XNR	4

(2) Contents of Pop-up Menu

The pop-up menu displayed in response to the keyed-in character is indicated below. In the pop-up menu, it is not possible to move the cursor to the empty area.

[A]	[C]	[D]	[I]	[M]
ANI	CLR	DCR	INR	MVI
ADI	CMR	DEC	INRW	MOV
ADD	COI	DST	IPSH	MULW
ANR	CMP	DIN	IPSHD	MVLW
ADC	CPI	DIVW		MCR
ADDW	CPR	DSTW		
ADR	COR	DCRW		
APSH	CLRW			
	CMRW			
	CORW			
	CPRW			

[S]	[T]	[O]	[X]
SUB	TIM	ORR	XRI
SBI	TMR	ORI	XRR
SUBW	TPSH		XOR
SET			XNR
SUBP3			
SUBP4			
SUBP5			
SUBP6			
SUBP7			
SUBP9			
SUBP11			
SUBP14			
SUBP17			
SUBP17			
SUBP18			

If the following characters are keyed-in, the pop-up menu is not displayed but the corresponding instruction is input.

N: NOP  
E: END  
J: JMP  
P: PUSH  
R: RTI

### (3) Register Instruction Inputting Procedure

- ① Key-in the first character of the register instruction to be input.

The pop-up menu screen is displayed.

- ② Move the cursor to the required register instruction and press the [WR] key.

When the register instruction is input, the next pop-up menu screen is displayed for the input of the necessary operand. Input the contact number, register number, or numeric value.

Note that it is not necessary to input “#” when inputting a register number.

- ③ Press the [INS] key to insert the input register to the ladder.



(4). Example of Input – MVI Instruction

The operation and screen display are explained below using the input of MVI instruction as an example.

① Key-in “M” to the key buffer area.

The pop-up menu showing five instructions beginning with “M” is displayed. The character “M” which has been keyed-in is not displayed in the key buffer display area.

To cancel the pop-up menu, press the [RST] key.

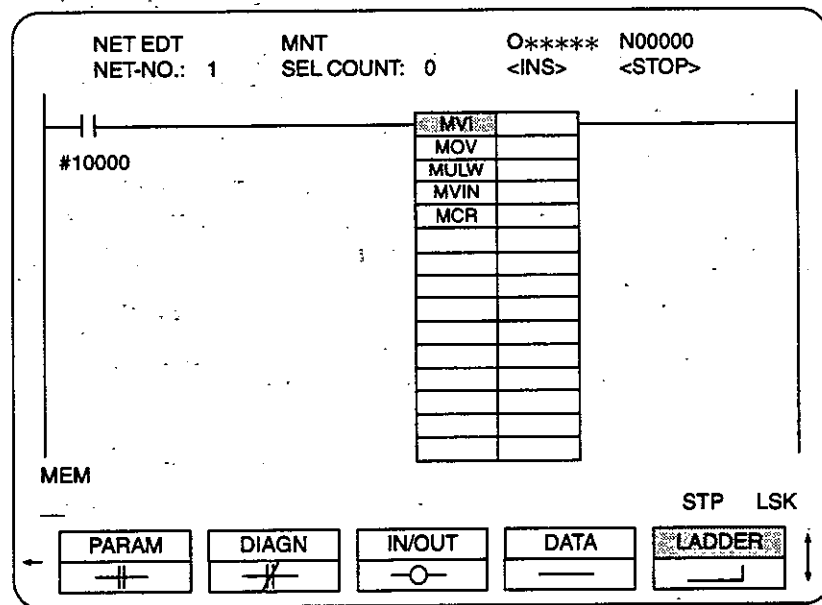


Fig. 8.13 Pop-up Menu Screen after Keying-in “M”



- ② The MVI instruction format is displayed in the pop-up screen when "M" is input.

Select the instruction to be input from the pop-up menu by moving the cursor onto it and then input the numbers by using the keyboard.

The screen given below is an example of a screen when the MVI instruction is selected. If another instruction is selected, the pop-up screen meeting the input instruction is displayed.

After inputting the operand, depress the [INS] key to insert the input instruction to the ladder. If the operand has been input correctly, the input instruction is inserted to the ladder and displayed in the manner as shown in Fig. 8.20. However, if the operand has not been input correctly, the following warning message is displayed.

"INPUT ERROR"

The system enters the state where the input of operands is waited for.

To cancel this operand input waiting status, depress the [RST] key.

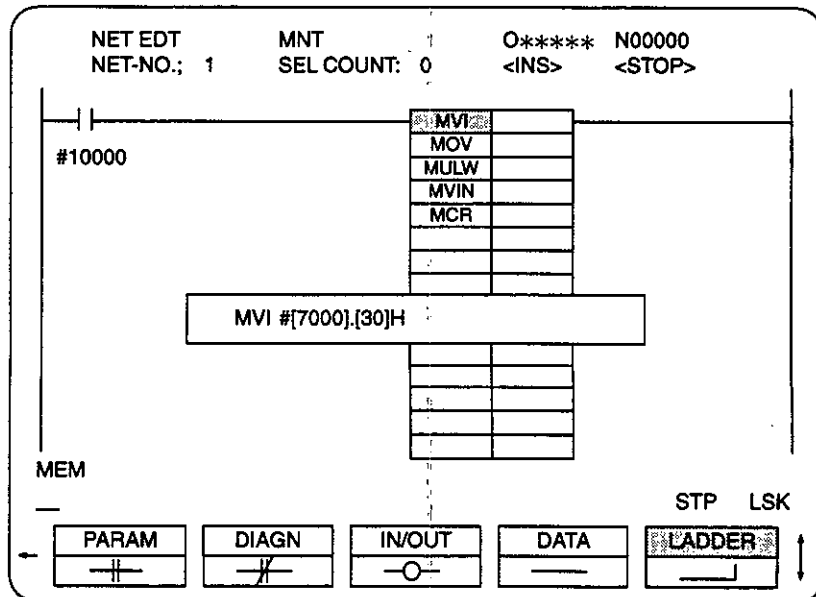


Fig. 8.14 MVI Instruction Input Screen

- ③ Press the [INS] key.

The input instruction is inserted to the ladder in the manner as shown in Fig. 8.15. The cursor position remains unchanged before or after the input of the MVI instruction.

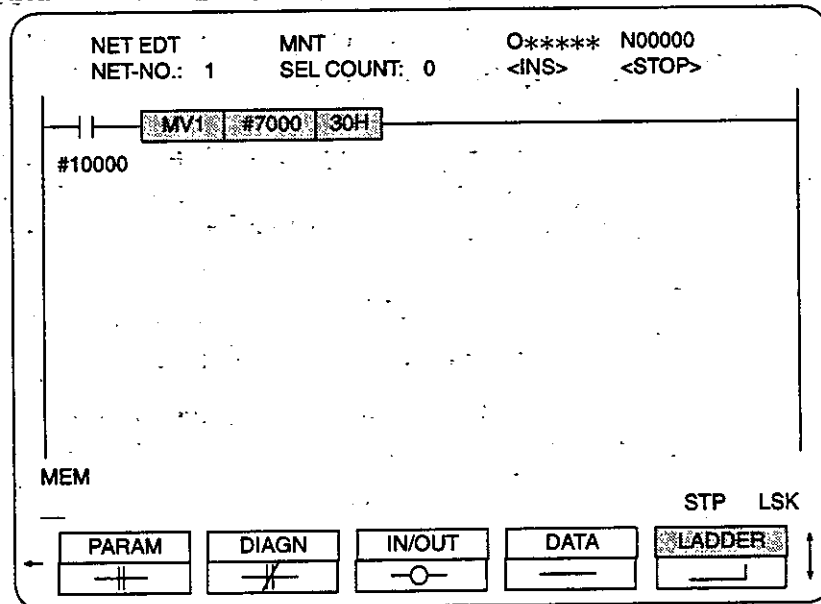


Fig. 8.15 Ladder Display after Inputting MVI Instruction.

## (5) Operand Input Pop-up Box Screen

## (a) Types of operand input pop-up boxes

Depending on the input register instruction, the corresponding operand input pop-up box appears on the screen.

For the operands, check is made whether or not the input is allowed.

① JMP [1]	JMP, ADR
② IPSH [7000]	IPSH, IPSHD, SUBP
③ INRW N [7000]	INR, DCR, CLR, CMR, INRW, DCRW, CLRW, APSH, PUSH, XOR, XNR
④ INRW N [9023]	TPSH
⑤ CPI # [7000], # [ ]H	TMR, ADI, SBI, ANI, ORI, XRI, DEC, COI, CMP, CPI, MVI
⑥ ADDW # [7000], # [ ]	TIM, ADD, SUB, ANR, ORR, XRR, CPR, COR, MOV, ADC, ADDW, SUBP, MULW, DIVW, CORW, CPRW, MVIW
⑦ DIN # [7000]. # [7000], [ ]H	DST, DIN, DSTW

## (b) Registers that can be input

The registers that can be input in the operand input pop-up box for the individual register instructions are indicated in Table 8.4.

If a register outside the allowable range is input, the following warning message is displayed.

“INPUT ERROR”

Table 8.4 Range of Registers that can be Input for Individual Register Instructions

Register Instruction	Input Permitted Registers	Range Group
TIM TMR	#1300 to #1399, #1700 to #1799 #1300 to #1399, #1700 to #1799	A
TPSH	N9000 to N9323	B
Other instructions	#1000 to #1063, #1100 to #1163, #1200 to #1299 #1400 to #2999, #3000 to #3159, #3500 to #3699	C

---

(c) **Input range**

With each register instruction, the input range is determined for individual operands. If a value outside the allowable range is input, the following warning message is displayed.

**“INPUT ERROR”**

The ranges of registers and reals that can be input for individual registers are indicated in Table 8.5.



Table 8.5 Range of Registers and Reals of Register Instructions

Instruction	No. 1 Operand Check Range (A, B, C = Range Group in Table 8.5)	No. 2 Operand Check Range	No. 3 Operand Check Range	
TIM	A	0 - FFH		
TMR	A	C		
INR	C			
DCR	C			
CLR	C			
CMR	C			
ADI	C	0 - FFH		
SBI	C	0 - FFH		
ANI	C	0 - FFH		
ORI	C	0 - FFH		
XRI	C	0 - FFH		
DEC	C	0 - FFH		
COI	C	0 - FFH		
CMP	C	0 - FFH		
CPI	C	0 - FFH		
MVI	C	0 - FFH		
ADD	C	C		
SUB	C	C		
ANR	C	C		
ORR	C	C		
XRR	C	C		
CPR	C	C		
COR	C	C		
MOV	C	C		
DST	C	C		0 - FFH
DIN	C	C		0 - FFH
ADC	C	C		
ADDW	C	C		
SUBW	C	C		
MILW	C	C		
DIVW	C	C		
INRW	C			
DCRW	C			
CLRW	C			
CMRW	C			
CORW	C			
XOR	C			
XNR	C			
CPRW	C	C		
MVIW	C	0 - FFFFH		
DSTW	C	C	0 - FFFFH	
JMP	Numeric value of 1 to 256			
ADR	Numeric value of 1 to 256			
IPSH	0 - FFFFH			
APSH	C			
PUSH	C			
TPSH	B			
IPSHD	-999999999 to 999999999			
SUBP	5, 6, 7, 9, 11, 14, 17, 18, 23, 31, 32, 34, 35, 36, 37, 38, 39			

(d) Operand input procedure (decimal, hexadecimal)

In the operand input pop-up box, a real can be input in either decimal or hexadecimal. If "H" is entered after entering a real, it is regarded as a hexadecimal number.

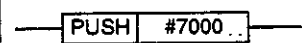
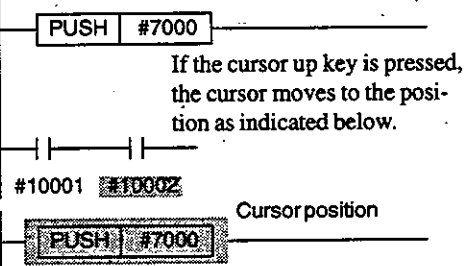


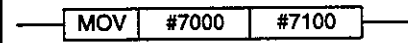
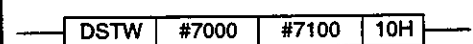
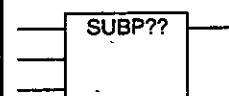
Examples: 64H_ Regarded as "100"  
 64_ Regarded as "64"

(6) Moving the Cursor on the Register and SUBP Instructions

There are cases in which the cursor movement is not allowed when the register or SUBP instruction is on the edit screen.

The cursor movement on the register or SUBP instruction is executed in the following manner.

Table 8.6

Cursor Position	Description on Cursor Display Position
	<p>When the cursor is positioned in the area within a frame, cursor movement is possible only within the instruction display area.</p>  <p>If the cursor up key is pressed, the cursor moves to the position as indicated below.</p> <p>#10001 #10002</p> <p>Cursor position</p>  <p>#10001 #10002</p>
	
	
	
	

## (7) Patterns that does not Allow Register Instruction

Input of a register instruction is not allowed in the positions indicated below.

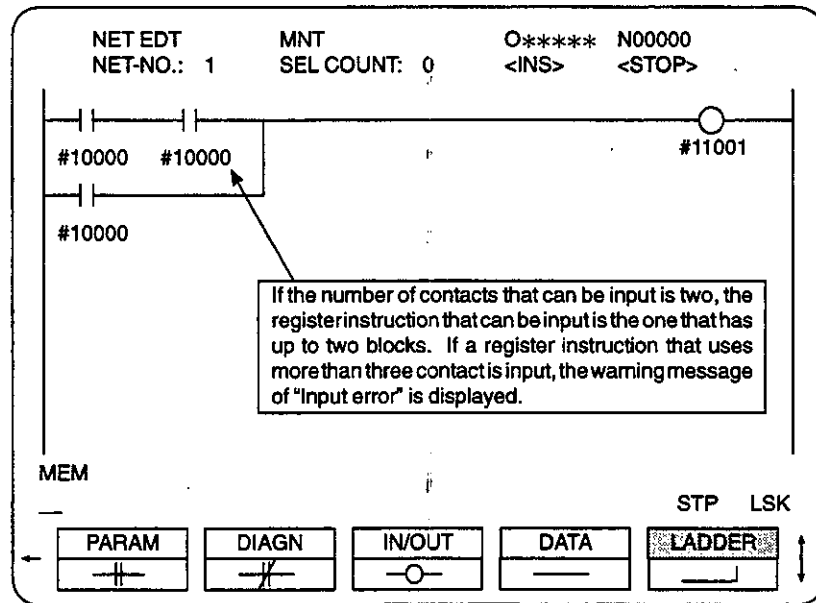


Fig. 8.16

## 8.4.6 Canceling the Net Edit Function

It is possible to cancel (quit) the contents of net edit during editing. Key-in "Q" and press the [WR] key.

## 8.4.7 Exiting the Edit Function

### (1) Operation Procedure

- ① Press a process, job, or function soft-key other than [NET EDT] to exit the net edit function.

When exiting the net edit function, the NC executes the check on the following six items.

- ② Check 1: Checking the connection status in one net

The NC checks whether all lines are connected. If the edit has been finished with an open line remaining as shown below, the "NOT CONNECTED" error occurs.

Example 1: Net is not connected.

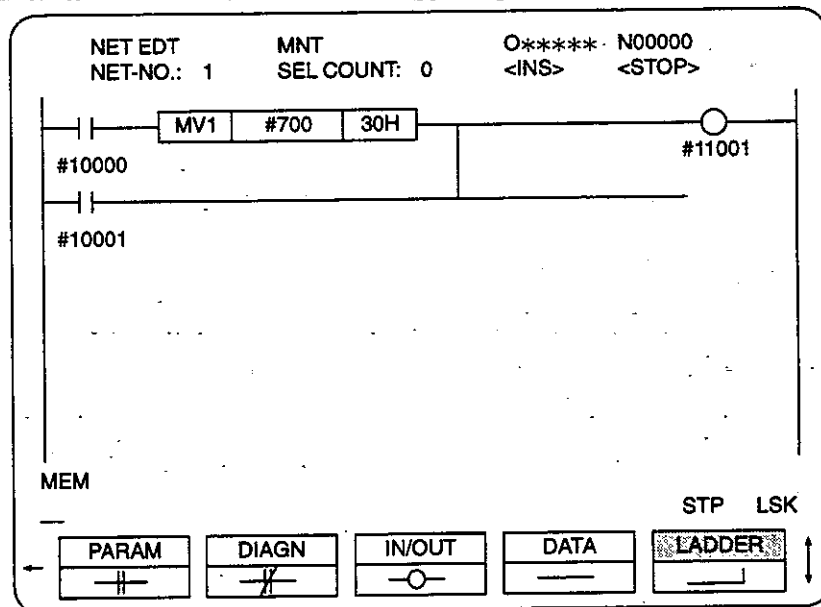


Fig. 8.17 Line Disconnection Error



Example 2: In the case of the ladder shown in Fig. 8.18, the ladder is not one net and therefore connection error occurs.

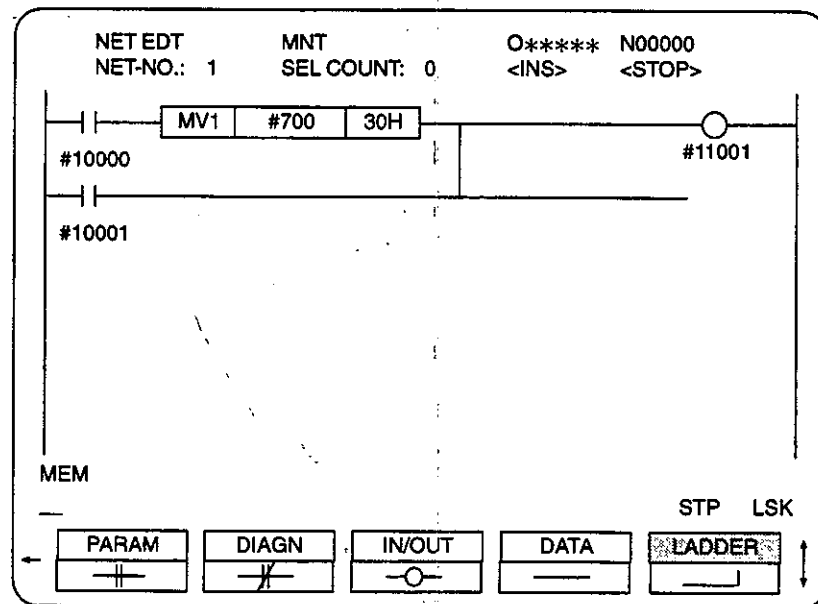


Fig. 8.18 Incorrect Net (More than One Ladder)

③ Check 2: Start instruction check

The NC checks whether any of the instructions indicated below is designated at the start of a net. If not, the "START INST ERR" is displayed.

CMP, DEC, PUSH, ADR, SET, RET, END, POP, IPSH, NOP, LD, LD-NOT, TPSH, APSH, SUBP23, IPSHD

④ Check 3: Double-instruction check

The NC checks that no instruction is designated with any of the following instructions. If designated, the "DOUBLE INST ERR" is displayed.

END, RET, RTH, IPSH, APSH, PUSH, TPSH, ADR, IPSHD

⑤ Check 4: Single-instruction check

The NC checks whether some instruction is designated with any of the following instructions. If not designated, the "SINGLE INST ERR" is displayed.

TIM, TMR, INR, DCR, CLR, CMR, ADI, SBI, ANI, ORI, XRI, DEC, COL, CMP, CPI, MVI, ADD, SUB, ANR, ORR, XRR, CPR, COR, MOV, DST, DIN, ADC, ADDW, SUBW, MULW, DIVW, INRW, DCRW, CLRW, CMRW, CORW, CPRW, MVIW, DSTW, MCR, RTI, JMP

⑥ Check-5: Analysis possibility check

The NC checks if the edited net can be converted into a sequence program. If conversion is not possible, the "INVALID NET" is displayed.

⑦ Check 6: Sequence program size check

The NC checks that a sequence program is within the allowable size.

The maximum size of a sequence program is 131,055 bytes which are approximately 32,700 steps (1 step = 4 bytes). If the size of the sequence exceeds 131,055 bytes, the "SIZE OVER" is displayed.

⑧ After the completion of the check indicated above, the screen changes.

During net check, the "NET CHECKING" is displayed on the screen as shown in Fig. 8.19.

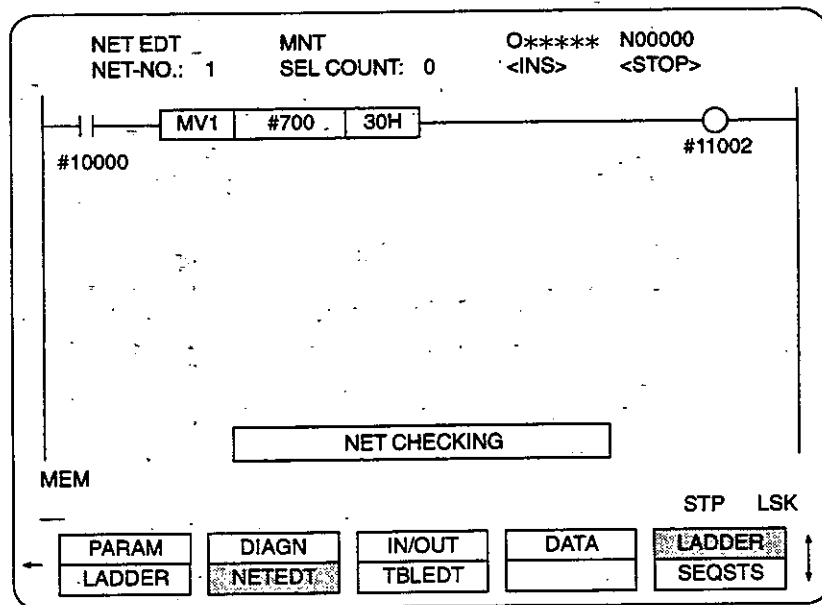


Fig. 8.19 Screen during Net Check

## (2) Warning Messages

If an error is found in net check, a corresponding warning message is displayed.

Table 8.7

Warning Message	Description	Corrective Action
NOT CONNECTED	The net has not been completed.	Check the created or edited net and connect the line.
SIZE OVER	The size of sequence exceeds the allowable limit.	Review the program and delete several codes.
START INST ERR	An instruction that must be placed at the beginning of a net is not designated.	Designate an instruction that must be placed at the beginning of a net.
SINGLE INST ERR	An instruction that must be designated in a net with another instruction is designated without other instruction.	Refer to the PLC instruction manual and make corrections.
DOUBLE INST ERR	In one net, some instruction is designated with an instruction that must be designated in a net without another instruction.	Delete an instruction that is designated with the instruction that must be designated in a net without other instruction.

## (3) Forced stop of net check

If the power is turned OFF during net check, the information might not be updated entirely. Therefore, the sequence is automatically initialized (booting from flash ROM to CMOS) when the power is turned ON next.

Edit the sequence again.



---

## 8.5 TABLE EDIT FUNCTION

The table data editor is provided to edit the table data.

The data of the following three tables can be edited.

- Message table
- Conversion table
- Symbol table



### 8.5.1 Editing the Data in the Conversion Table

The conversion data that have been set by using the pseudo instruction "CONVERSION" can be edited.

The data can be edited in units of bytes. The data for which word or double-word designation has been made in the conversion table of the source program are also displayed in units of words in this screen.

Conversion data are set in hexadecimal.

The conversion data numbers are indicated below:

N9000 - N9007 256 bytes

N9008 - N9023 128 bytes

"FFH" is displayed for undefined conversion data.

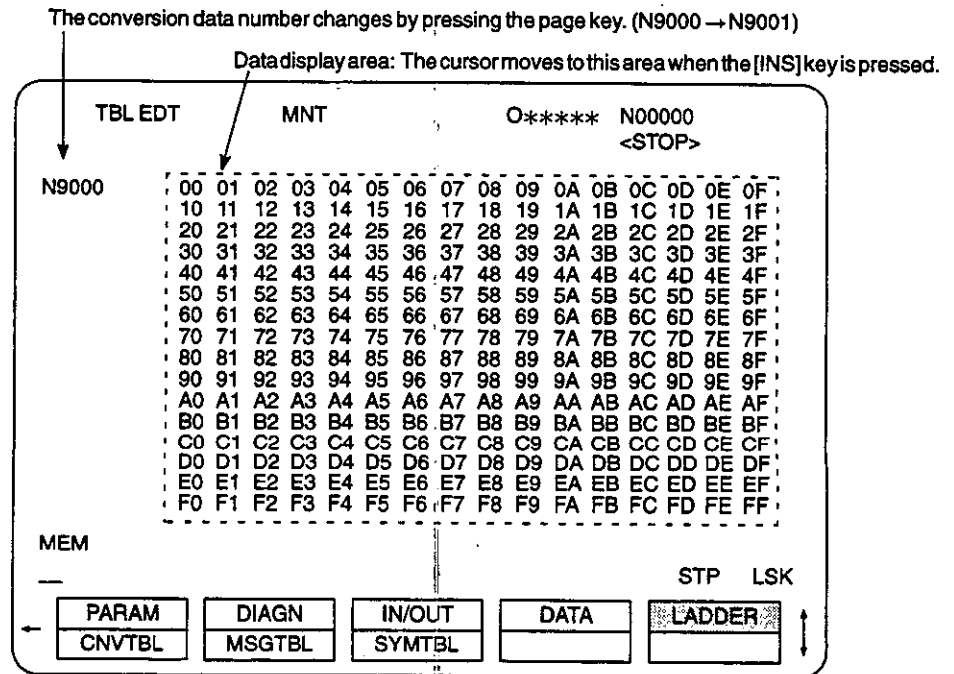


Fig. 8.20 Conversion Data

The conversion data editing procedure is explained below.

- ① After displaying the conversion data page of the required conversion data number, press the [INS] key.

The cursor appears on the first byte position in the data display area.

- ② Move the cursor to the data to be changed. Key-in the required numeric value and press the [WR] key.

Example: [4][2][WR]

- ③ Press the [INS] key.

## 8.5.2 Editing the Data in the Message Table

The message data that have been set by using the pseudo instruction "MESSAGE" can be edited.

Up to 40 characters can be input as message data.

Fig. 8.21 shows the message data edit screen.

The message data number is indicated below:

#9024 to #9323

The screenshot shows a terminal window titled "TBL EDT" and "MNT". At the top right, it displays "O***** N00000" and "<STOP>". The main area contains a list of message data entries:

- #9024 SPINDLE ALARM
- #9025 TOOL SET OK
- #9026 0123456789123456789012345678901234567890
- #9027
- #9028

At the bottom left, the word "MEM" is displayed. At the bottom right, "STP" and "LSK" are shown. Below these are several menu buttons arranged in two rows:

PARAM	DIAGN	IN/OUT	DATA	LADDER
CNVTBL	MSGTBL	SYMTBL		

Vertical arrows are present on the left and right sides of the menu buttons.

Fig. 8.21 Message Data Edit Screen

The message data editing procedure is explained below.

- ① Move the cursor to the message data number of the message that should be changed.
- ② Key-in the required message and press the [WR] key.

### 8.5.3 Editing the Data in the Symbol Table

The symbol name that has been set by using the pseudo instruction "SYMBOL" can be edited.

As a symbol name, a character-string of up to five characters can be set for one contact. Registration of a symbol is not possible in the byte register.

When registering symbol names, there must be no blank in the contact number area. If there is a blank, it is regarded as the end of registration and the symbol data registered after the blank, if any, is not output in text output operation.

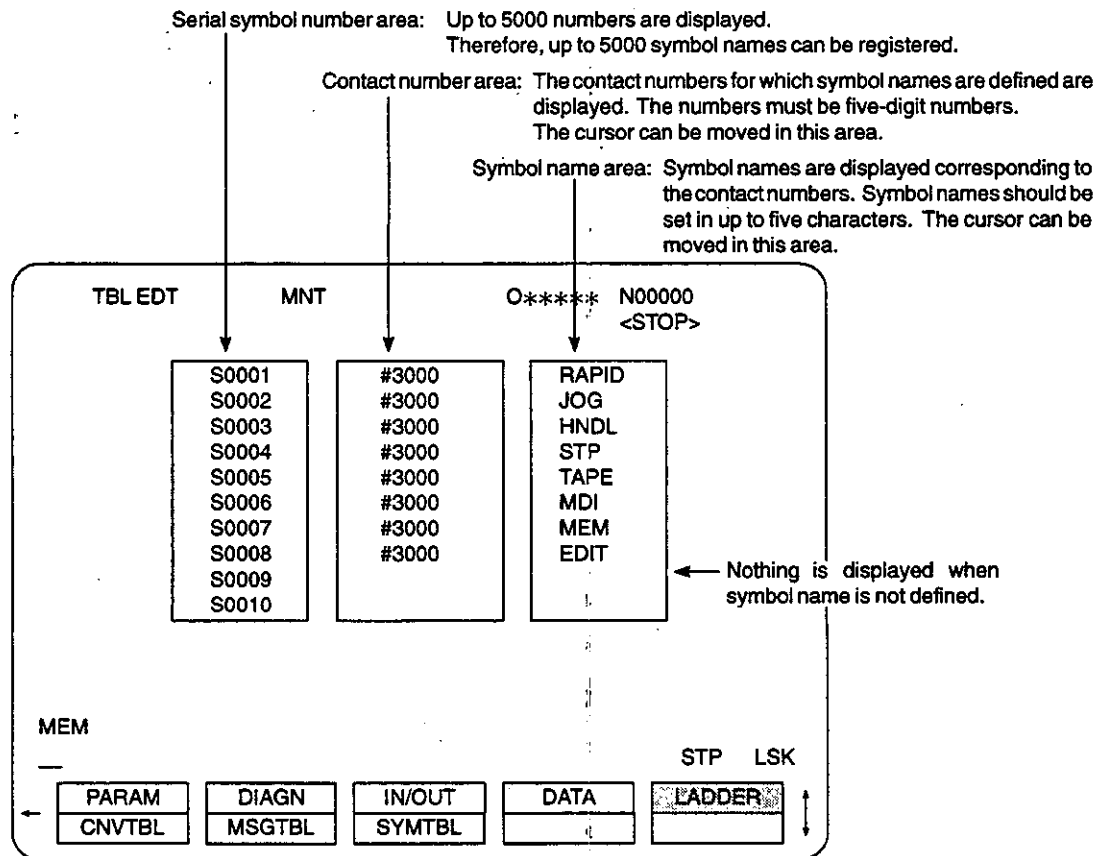


Fig. 8.22 Symbol Name Edit Screen

---

The symbol name editing procedure is explained below.

- ① Move the cursor to the contact number area and input the contact number to be set.  
The maximum serial number is 5000.
- ② Move the cursor to the symbol name area and input the symbol name. A symbol name should be input in a maximum of five characters.



- 
1. When inputting a register number, it is not necessary to input "#".
  2. For a register number, only a numeric value is allowed. If a character other than a numeric value is input "INPUT ERROR" is displayed.
  3. A symbol name is a character-string of up to five characters. If a character-string longer than five characters is input, "INPUT ERROR" is displayed.
-



## 8.6 INPUT/OUTPUT FUNCTION

The I/O function uploads the created sequence to a PC card and downloads the sequence stored in a PC card to the NC.

### 8.6.1 Downloading the Sequence Program

The procedure used for downloading the sequence program (execution module *.BIN) stored in the PC card to the NC is explained below.

- ① Save the sequence execution module to be downloaded to the PC card.
- ② Select the maintenance job and select the "SEQUENCE DATA" by the IN/OUT PARM function. See Fig. 8.23.
- ③ Select the sequence execution module to be downloaded from the list of files in the PC card. See Fig. 8.24.

If a file other than a sequence file is selected, the following warning message is displayed.

"FILE DATA ERR"

- ④ Press the [WR] key.
- ⑤ The following message starts blinking and download starts.

"INPUTTING"

- ⑥ Upon completion of downloading, the following message is displayed.

"INPUT COMPLETED"

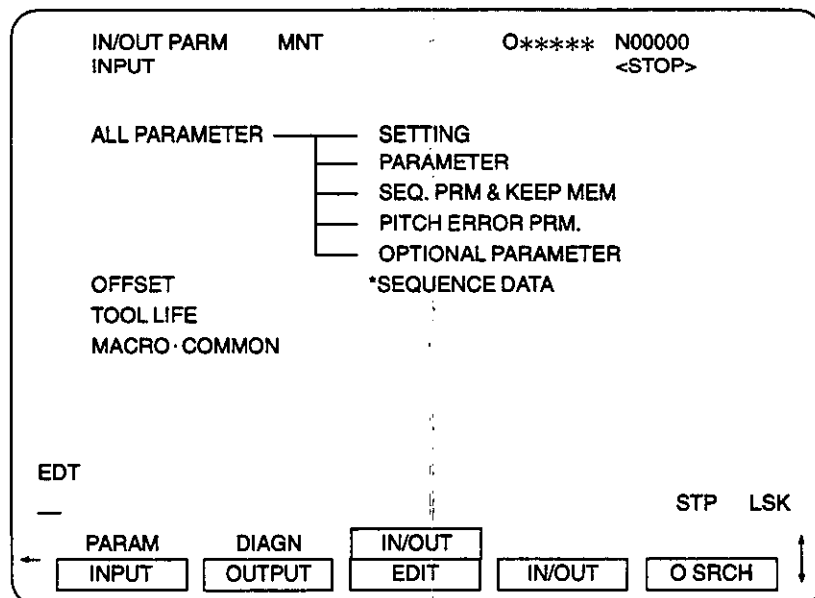


Fig. 8.23 IN/OUT PARM Function Screen

IN/OUT	PARM	MNT	
			O***** N00000
	PC CARD DIR (A: \YASNAC)		
		<DIR>	XXXXXX09 BIN
		<DIR>	XXXXXX10 BIN
	DIRECT01	<DIR>	XXXXXX11 BIN
	DIRECT02	<DIR>	XXXXXX12 BIN
	XXXXXX01 BIN		XXXXXX13 BIN
	XXXXXX02 BIN		XXXXXX14 BIN
	XXXXXX03 BIN		XXXXXX15 BIN
	XXXXXX04 BIN		XXXXXX16 BIN
	XXXXXX05 BIN		XXXXXX17 BIN
	XXXXXX06 BIN		XXXXXX18 BIN
	XXXXXX07 BIN		XXXXXX19 BIN
	XXXXXX08 BIN		XXXXXX20 BIN
		REMAINING	3,123,456 BYTES
EDT			
			STP LSK
	PARAM	DIAGN	IN/OUT
←	INPUT	OUTPUT	EDIT
			IN/OUT
			O SRCH ↑

Fig. 8.24 PC Card File Directory Screen

## 8.6.2 Uploading the Sequence Program

It is possible to upload a sequence program from the NC to a PC card. The file format of the upload file can be selected from the execution module format (the same format as in downloading) and the text format.

**Text format:** The ladder sequence program (object file) in the NC is reversely converted to the state before compilation.

The sequence program consisting of multiple modules which are linked is output in one file. If such a file is compiled, the object file capacity exceeded could occur (one object file is greater than 64K.). In this case, divide the sequence program into two or more files and create the execution module by using the JXSD offline system software package.

The procedure used for uploading the sequence program stored in the NC is explained below.

- ① Prepare a PC card having sufficient free area.
- ② Select the maintenance job and select the "SEQUENCE DATA (TEXT)" or "SEQUENCE DATA (BIN)" by the IN/OUT PARM function. See Fig. 8.25.
- ③ Key-in the sequence execution module name to be uploaded. See Fig. 8.26.
- ④ Press the [WR] key.
- ⑤ The following message starts blinking and upload starts.

"OUTPUTTING"

- ⑥ Upon completion of uploading, the following message is displayed.

"OUTPUT COMPLETED"

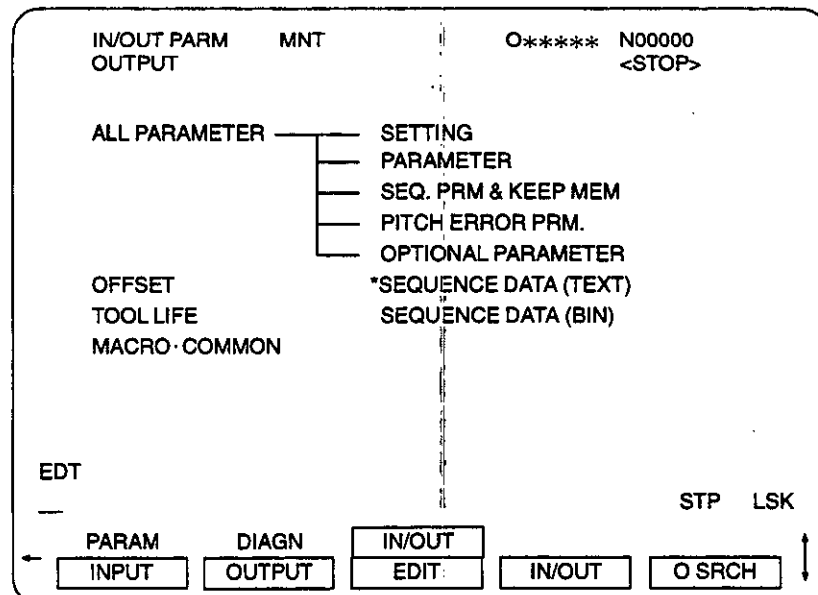


Fig. 8.25 IN/OUT PARM Function Screen

IN/OUT PARM    MNT            O***** N00000  
SEQ OUT (TEXT)

PC NAME: *****

EDT

STP    LSK

---

PARAM    DIAGN    IN/OUT  
INPUT    OUTPUT    EDIT    IN/OUT    O SRCH

Fig. 8.26 Sequence Execution Module Name Input Screen

- ⑦ If the file having the same name as the file to be uploaded already exists in the PC card, the following message is displayed.

“OVERWRITE? (Y/N)”

Press [Y] to overwrite. If the existing file must not be overwritten, press [N].



#### Warning Related with PC Card

There are cases that input/output using a PC card is not possible due to defective PC card, writing error, or other problems. If input is not possible, the relevant warning message is displayed.

“DEVICE NOT READY!”: A PC card is not installed.

“FORMAT ERROR!”: The PC card is not formatted.

“ACCESS ERROR!”: Read/write is attempted while a file in the PC card is input/output.

“PC CARD FULL!”: PC card free area is insufficient.

## 8.7 SEQ STS (SEQUENCE STATUS) FUNCTION

### 8.7.1 Display of Sequence Status

On the SEQ STS screen, sequence memory status and sequence scan speed can be displayed. It is also possible to change the set values.

The SEQ STS screen is shown below.

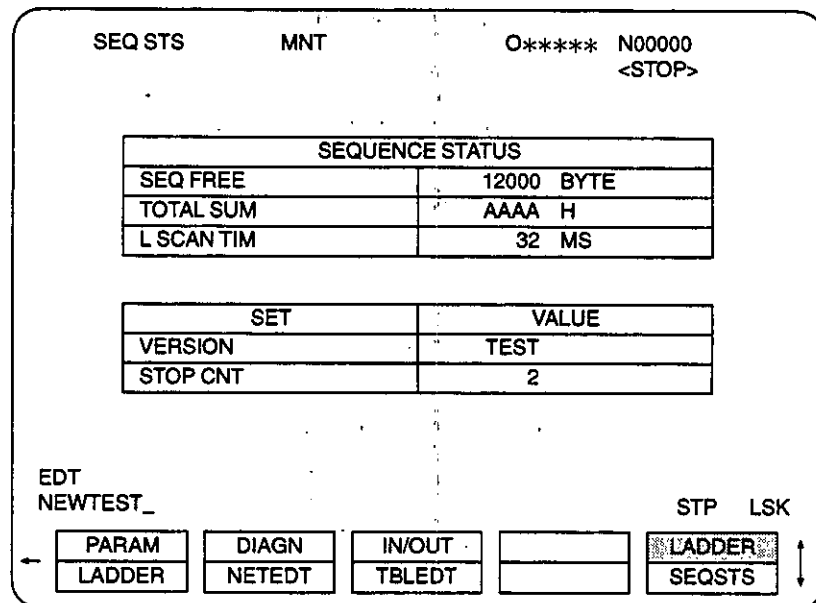


Fig. 8.27

#### (1) SEQ FREE

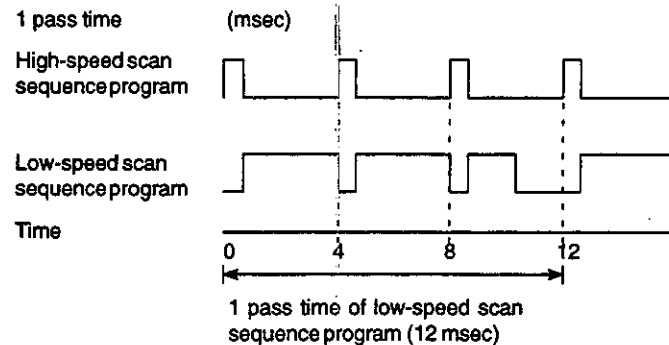
The size of the free area for storing sequence programs is displayed.

#### (2) TOTAL SUM

Total value of the data stored in the CMOS area, including sequence programs and tables, is displayed.

#### (3) L SCAN TIM

The time required to execute a low-speed scan sequence program once is displayed in a multiple of 4 msec.



#### (4) VERSION

The version name of the sequence designated by the pseudo instruction "VERSION" is displayed.

The sequence version can be changed while the sequence is stopped.

The version name can be set in up to eight characters. If more than eight characters are input, an error occurs and the following message is displayed.

"INPUT ERROR"

#### (5) STOP CNT

The stop count designated by the pseudo instruction "LOWSTOPCOUNT" is displayed.

The STOP CNT can be changed while the sequence is stopped.

Setting range is from 1 to 20; if a numeric value outside this range is set, an error occurs and the following message is displayed.

"INPUT ERROR"

### 8.7.2 INITI (Initialization) Function

The function initializes the sequence programs in the NC. The table related data is also cleared.

When the [INITI] function soft-key is pressed, the PLC manager executes the following processing.

- Sets "TEST" for VERSION.
- Sets "3" for "LOW-SPEED STOP COUNT".
- Clears the sequence program area and inserts the following programs from the beginning.
  - HIGHSEQUENCE,
  - RTH,
  - ENDP,
  - LOWSEQUENCE,
  - RET,
  - ENDP,
- Clears the message data area.
- Clears the conversion data area.
- Clears the symbol data area.

- ① Press the [INITI] function soft-key.

The following message is displayed.

“INITIALIZE SEQ.? (Y/N)”

SEQUENCE STATUS	
SEQ FREE	12000 BYTE
TOTAL SUM	AAAA H
L SCAN TIM	32 MS

SET	VALUE
VERSION	TEST
STOP CNT	2

INITIALIZE SEQ.? (Y/N)

MEM

STP LSK

PARAM	DIAGN	IN/OUT		LADDER
	INIT			

Fig. 8.28 Sequence Initialization Screen

- ② Press [Y] and [Enter] keys.

The sequence programs in the NC are initialized and the following screen is displayed.

SEC STS      MNT      O***** N00000  
<STOP>

SEQUENCE STATUS	
SEQ FREE	12798 BYTE
TOTAL SUM	???? H
L SCANTIM	12 MS

SET	VALUE
VERSION	TEST
STOP CNT	3

MEM

PARAM	DIAGN INIT	IN/OUT		STP LSK LADDER
-------	---------------	--------	--	-------------------

Fig. 8.29 Sequence Initialization Screen (After Initialization)



1. Initialization operation initializes all data being edited.
2. Before executing initialization, make sure that the data can be initialized.



## 8.8 LIST OF MESSAGES

Messages displayed during online editing are given in Tables 8.8, 8.9, and 8.10.

### 8.8.1 List of Messages

Table 8.8 List of Messages

Message	Description
COLLECTING	The nets are being selected.
DELETING	The net is being deleted.
DELETION COMPLETED	Net deletion has been completed.
NET CHECKING	Entire sequence program is being checked during the execution of a sequence program.
INPUTTING	Sequence program data are being input from a PC card.
INPUT COMPLETED	Inputting of sequence program data has been completed correctly.
OUTPUTTING	Sequence program data are being output to a PC card.
OUTPUT COMPLETED	Outputting of sequence program data has been completed correctly.
SEARCHING	Search processing is being executed on the ladder screen.
INPUT CONTACT NUMBER	The message requesting the input of a contact number when the contact function is selected.
OVERWRITE? (Y/N)	When outputting a file to a PC card, the same file name as the one already existing in the PC card is designated.
INITIALIZE SEQ.? (Y/N)	This message is displayed when the [INITI] function soft-key is pressed.



## 8.8.2 List of Warning Messages

Table 8.9 List of Warning Messages

Message	Description
SELECTION OVER	In net selection, more than 10 nets are selected.
INPUT ERROR	Error in data input
CONTACT OVER	More than 100 contacts are set in one net.
DEVICE NOT READY	PC card is not set.
FORMAT ERROR!	PC card is not formatted.
ACCESS ERROR!	PC card read/write error
PC CARD FULL!	PC card free area is insufficient for writing the data.
NO JMP-ADR	ADR instruction is not designated corresponding to JMP.
NO START INST	An instruction that must be designated at the beginning of a ladder is not designated.
NOT CONNECTED	Line connection is incomplete.
SUBP FORMAT ERROR	Format error in the SUBP instruction
EXECUTING	An invalid function is selected during sequence execution.
DOUBLE INST ERR	An instruction which must not be designated with another instruction is designated in a net.
SINGLE INST ERR	An instruction which must be designated with another instruction is designated without other instruction in a net.
SIZE OVER	The size of sequence exceeds the allowable limit.
FILE DATA ERROR	The data input to the PC card are not the sequence file for JX.
DOUBLE LABEL	There is more than one label for JMP instruction.
INVALID NET	A net which cannot be analyzed is edited.

## 8.8.3 List of Alarm Messages

Table 8.10 List of Alarm Messages

Message	Description
BACKUP THE SEQUENCE PROGRAM	After editing the sequence program, the system has been started with the system number switch set at "0" or "A" without backing up the sequence program.
PLC CMOS ERROR	The CMOS (hardware) in the JCP02 has been destroyed.
RESETTING HAS BEEN MADE!	Since the sequence data in the JCP02 have been destroyed, the contents in flash ROM have been transferred to the CMOS.

# 9

---

## DOWNLOADING AND UPLOADING LADDER PROGRAM

Chapter 9 describes the procedure for downloading and uploading the ladder program using flash ROM.

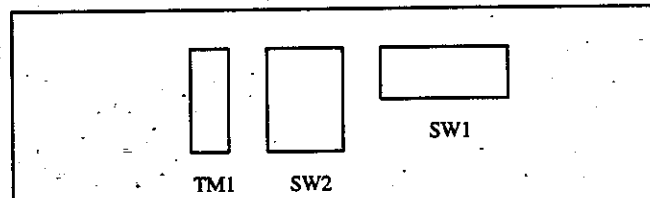
9.1	DOWNLOADING LADDER PROGRAM (PC CARD → FLASH ROM) .....	9 - 2
9.2	UPLOADING LADDER PROGRAM (FLASH ROM → PC CARD) .....	9 - 4



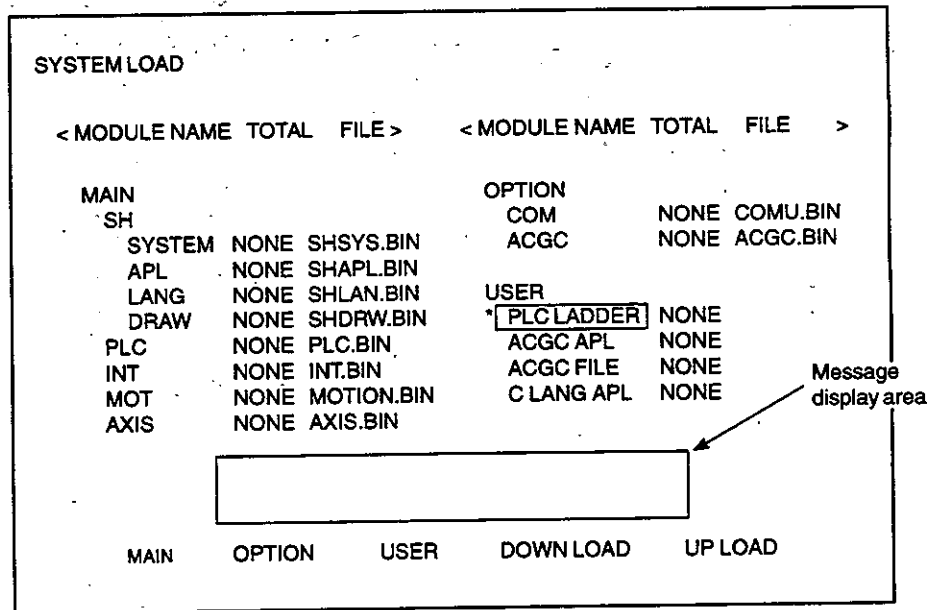
## 9.1 DOWNLOADING LADDER PROGRAM (PC CARD → FLASH ROM)

The procedure for downloading the sequence program from PC card to flash ROM is indicated below.

- ① Turn ON the SW1 on the JCP01 board.



- ② Turn ON the power.
- ③ Insert a PC card to CN03 (CARD) in the JCP01 board.
- ④ The following screen is displayed.



- ⑤ Move the cursor to "PLC LADDER" of USER by using the cursor up/down keys and press the [WR] key.
- ⑥ The "*" symbol appears to the left of "PLC LADDER" and the cursor moves to the file name input area.
- ⑦ The message "INPUT FILE NAME" appears.
- ⑧ Input the file name of the ladder program.

- ⑨ Press the [DOWN LOAD] function soft-key.

The following message is displayed.

“TRANSFERRING”

- ⑩ After the completion of transmission, the following message is displayed.

“COMPLETED”

- ⑪ Return SW1 to OFF and turn OFF the power.



---

## 9.2 UPLOADING LADDER PROGRAM (FLASH ROM → PC CARD)

The procedure for uploading the sequence program from flash ROM to PC card is indicated below.

① Carry out steps 1 to 8 described in section 9.1.

② Press the [UP LOAD] function soft-key.

The following message is displayed.

“TRANSFERRING”

③ After the completion of transmission, the following message is displayed.

“COMPLETED”

④ Return SW1 to OFF and turn OFF the power.



# YASNAC J300

# PLC PROGRAMMING MANUAL

---

**TOKYO OFFICE** New Pier Takesiba South Tower, 1-16-1, Kaigan, Minatoku, Tokyo 105 Japan

Phone 81-3-5402-4511 Fax 81-3-5402-4580

**YASKAWA ELECTRIC AMERICA, INC.**

**Chicago-Corporate Headquarters** 2942 MacArthur Blvd Northbrook, IL 60062-2028, U.S.A

Phone 1-847-291-2340 Fax 1-847-498-2430

**Chicago-Technical Center** 3160 MacArthur Blvd, Northbrook, IL 60062-1917, U.S.A.

Phone 1-847-291-0411 Fax 1-847-291-1018

**MOTOMAN INC.**

805 Liberty Lane West Carrollton, OH 45449, U.S.A

Phone 1-513-847-6200 Fax 1-513-847-6277

**YASKAWA ELÉTRICO DO BRASIL COMÉRCIO LTDA.**

Avenida Brigadeiro Faria Lima 1664-5°C.J 504/511, São Paulo, Brazil

Phone 55-11-815-7723 Fax 55-11-870-3849

**YASKAWA ELECTRIC EUROPE GmbH**

Am Kronberger Hang 2, 65824 Schwalbach, Germany

Phone 49-6196-569-300 Fax 49-6196-888-301

**Motoman Robotics AB**

Box 504 S38525 Torsås, Sweden

Phone 46-486-10575 Fax 46-486-41410

**Motoman Robotec GmbH**

Kammerfeldstraße 1, 85391 Allershausen, Germany

Phone 49-8166-900 Fax 49-8166-9039

**YASKAWA ELECTRIC UK LTD.**

3 Drum Mains Park Orchardton Woods Cumbernauld, Scotland, G68 9LD U.K

Phone 44-1236-735000 Fax 44-1236-458182

**YASKAWA ELECTRIC KOREA CORPORATION**

Paik Nam Bldg 901 188-3, 1-Ga Euljiro, Joong-Gu Seoul, Korea

Phone 82-2-776-7844 Fax 82-2-753-2639

**YASKAWA ELECTRIC (SINGAPORE) PTE. LTD.**

151 Lorong Chuan, #04-01, New Tech Park Singapore 556741, Singapore

Phone 65-282-3003 Fax 65-289-3003

**YATEC ENGINEERING CORPORATION**

Shen Hsiang Tang Sung Chiang Building 10F 146 Sung Chiang Road, Taipei, Taiwan

Phone 886-2-563-0010 Fax 886-2-567-4677

**BEIJING OFFICE** Room No. 301 Office Building of Beijing International Club, 21 Jianguomenwai Avenue, Beijing 100020, China

Phone 86-10-532-1850 Fax 86-10-532-1851

**SHANGHAI OFFICE** Room No. 8B Wan Zhong Building 1303 Yan An Road (West), Shanghai 200050, China

Phone 86-21-6212-1015 Fax 86-21-6212-1326

**YASKAWA JASON (HK) COMPANY LIMITED**

Rm 2916, Hong Kong Plaza, 186-191 Connaught Road West, Hong Kong

Phone 852-2858-3220 Fax 852-2547-5773

**TAIPEI OFFICE** Shen Hsiang Tang Sung Chiang Building 10F 146 Sung Chiang Road, Taipei, Taiwan

Phone 886-2-563-0010 Fax 886-2-567-4677



**YASKAWA**

**YASKAWA ELECTRIC CORPORATION**

**NOTICE:** To enable ongoing product modifications and improvements,  
specifications are subject to change without notice.

SIE-C843-13.1

© Printed in Japan August 1996 96-3 1.5 ◀▶