YASKAWA

A World of Automation Solutions™

YASNAC PC NC
PLC Programming Manual

Version: Beta 1.0

SAFETY INFORMATION

PRECAUTIONS

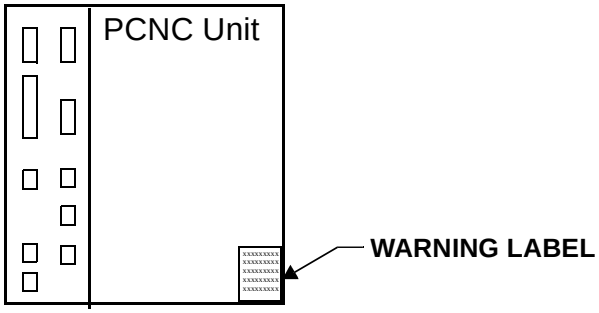
1. Read this instruction manual in its entirety before using the programming functions available in the YASNAC PCNC/PLC.

2. The following warning symbols are used to indicate precautions that the user must be aware of to safely use this equipment. Failure to follow these precautions can result in serious or possibly even fatal injury and damage to products or related equipment or systems.

This symbol indicates the presence of a potentially *hazardous condition* which, if not avoided, could result in serious personal injury or death.

This precautionary symbol appears in labels attached to YASNAC products to alert the user to conditions requiring concern for safety.

SPECIAL SAFETY NOTE: This symbol indicates that **ELECTRICAL SHOCK HAZARD** condition exists. DO NOT TOUCH any electrical connection terminals when the power is on, and for at least 5 minutes **after** switching off the power supply. Warning label is located on the CNC



NOTICE

USING THIS MANUAL

This manual decribes the procedures for programming the *YASNAC PC NC*/*PLC*. The programmable control system (hereafter called PLC) is installed in the *YASNAC PC NC* electrical cabinet.

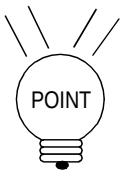
RELATED INFORMATION SOURCES

For additonal information, refer to the following manuals:

TITLE OF DOCUMENT	CONTENTS
YASNAC PCNC Operating Manual (YEA-SIE-C844-2.1)	Basic configuration and operating procedures
YASNAC PCNC Programming Manual (YEA-SIE-C844-2.2)	PCNC Program creation instructions
YASNAC PCNC PLC Programming Manual (YEA-SIE-C844-0.1)	PLC Program creation instructions
YASNAC PCNC I/O Signal Function (YEA-SIE-C844-2.3)	Describes functions between PCNC and PLC
YASNAC PCNC Connecting Manual (YEA-SIE-C844-0.2)	Instructions for connecting PCNC with machines, machine interface and peripheral equipment
	Describes man-machine-interface (MMI) programming, specifications and definitions.
MEMOCON GL120,G130 120 Series I/O Module User's Manual (Document No. SIEZ-C825-20.22)	Describes I/O power supply specifications
MEMOCON GL120,G130 Hardware User's Manual (Document No. SIEZ-C825-20.1)	Describes the AC input power supply specifications for I/O.
YASNAC PCNC Maintenance Manual (YEA-SIE-C844-2.9)	Describes service and maintenance procedures.

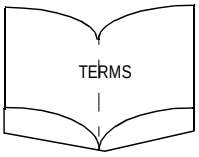
INFORMATION INDICATORS

The following symbols are used in this programming manual to emphasize particular information to the user:



Indicates important information to be remembered, i.e., precautionary alarm displays to prevent damaging devices.

Indicates supplementary material.



Indicates definitions of terminology that has not been explained before.

NOTES REGARDING SAFE OPERATION

It is important that the user should read this manual before installing, operating, performing any maintenance or inspecting the *YASNAC PC NC/PLC*. Also, the functions and performance of a NC machine tool are not determined by the CNC unit itself, therefore thoroughly read and familiarize yourself with the machine builder's documentation concerning the safe and most efficient ways to use the machine tool.

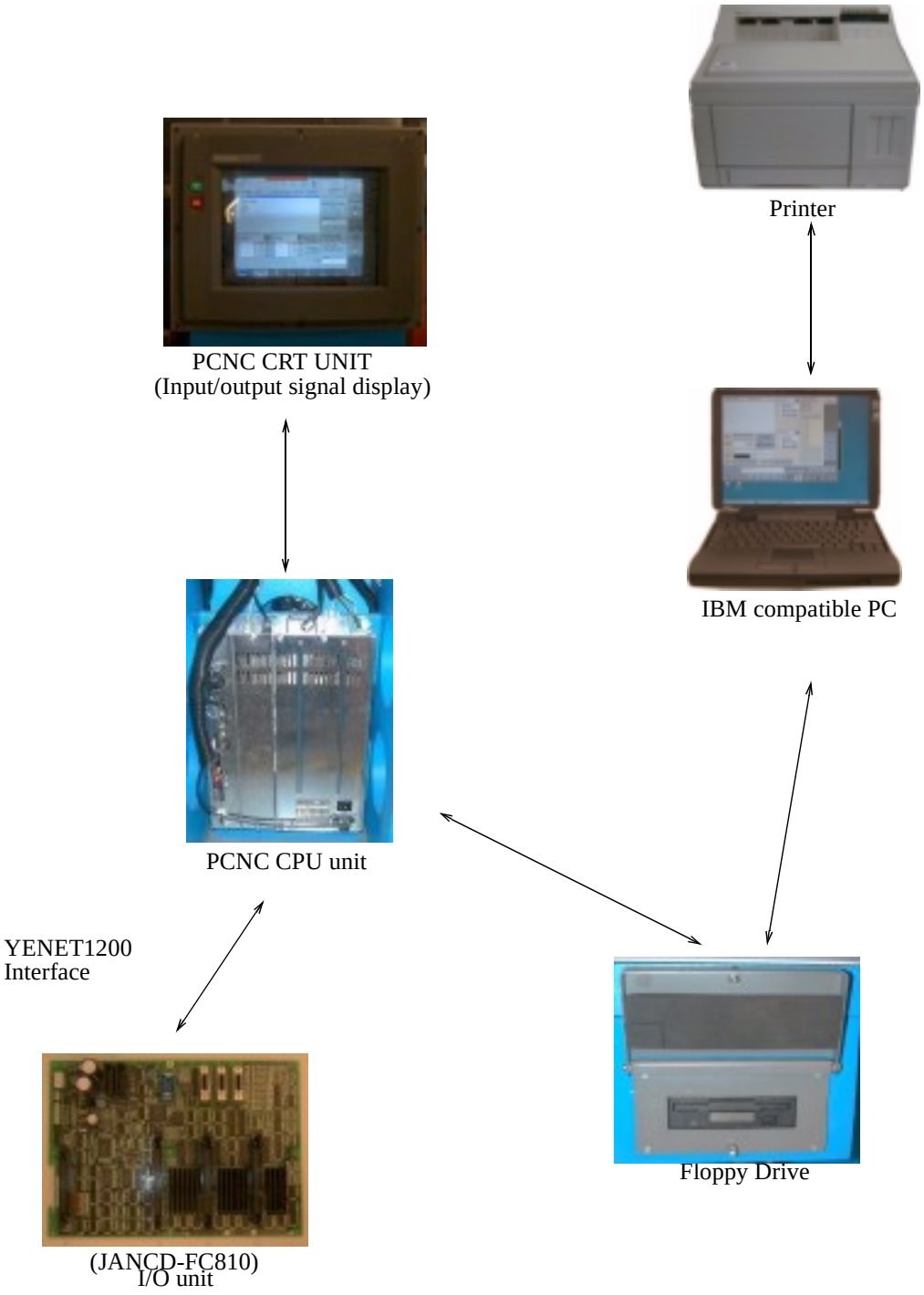
TABLE OF CONTENTS

1

System Configuration

Chapter 1 describes the YASNAC PLC PCNC system configuration

1.1 YASNAC PCNC/PLC System Configuration



2

Sequence Program Development Procedure

Chapter 2 describes the sequence program development off-line and on-line mode operations

2.1 SEQUENCE PROGRAM DEVELOPMENT

PCNC

The development procedure for PCNC is shown below.

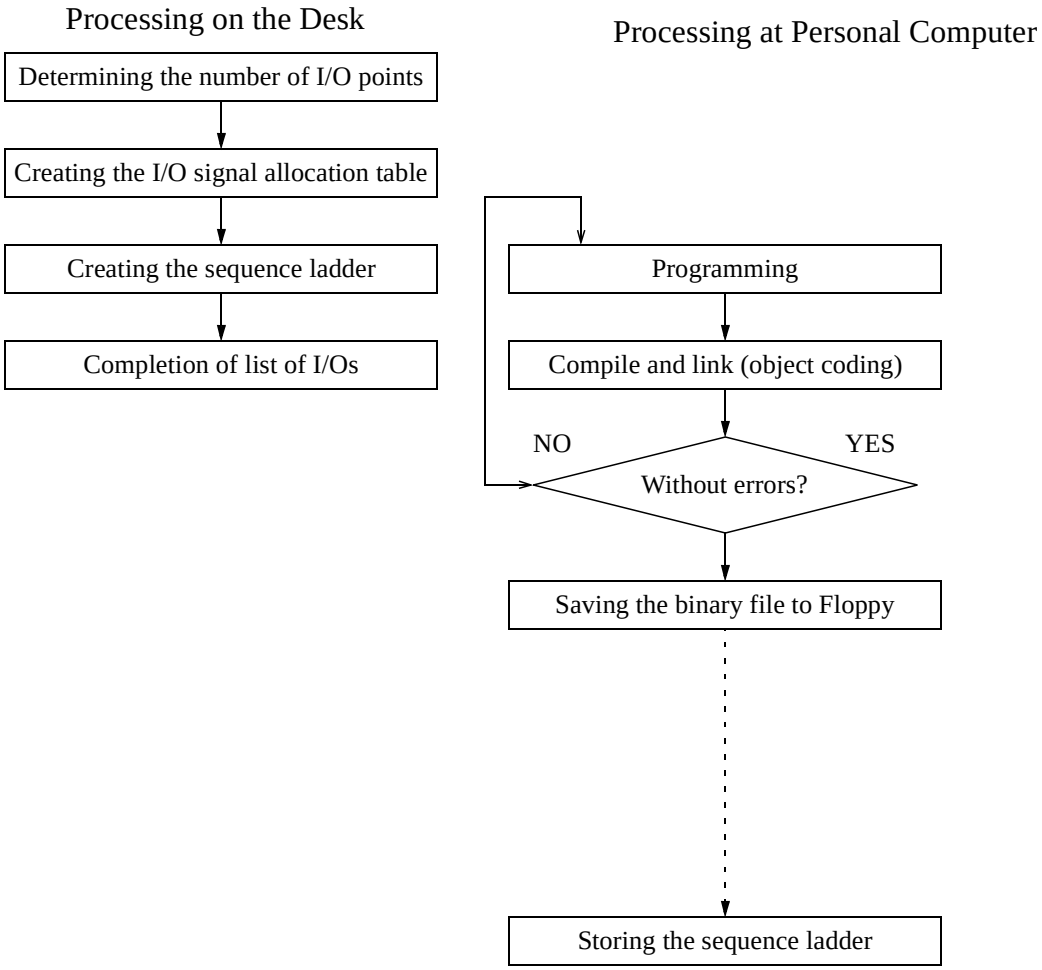


Fig. 2.2 YASNAC PCNC Sequence Program Development Procedure

3

PLC Program Specifications

Chapter 3 describes the specifications of the PLC programs

3.1 Basic Specifications

The basic specifications of YASNAC PCNC PLC are indicated below.

Table 3.2 PCNC PLC Basic Specifications

Item		Specifications
Control method		Scanning method
Processing time		0.5μs/step (approximate value)
Process Type	High-speed processing scan time	8ms
	Low-speed processing scan time	8ms × n
Program Memory Capacity	Basic	512K bytes
	Program	384K bytes
	Table symbol	128K bytes
Instructions	Basic instruction	61 kinds
	Macro instruction	22 kinds

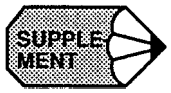
Note 1: Value “n” is determined by the high speed processing capacity and total program capacity.

2: 384K bytes are equivalent to approximately 24K steps of basic instructions.

3.2 Program Function Specifications

Table 3.3 Program Function Specifications

Item		Specifications
Internal relays		11960 points
Registers		1495 registers (8 bits/register)
Timers		188 timers (5 types)
Timer Type	8 msec to 2.4 sec	40 timers
	50 msec to 12.75 sec	60 timers
	100 msec to 25.5 sec	
	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 addressses, and the addresses used for internal relays cannot be used for registers. Similarily, 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. Similarily, the addresses used for battery back-ip memory cannot be used for keep relays.

3. Note that keep registers (#8000 to #9999) cannot be used for the keep relays.

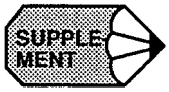
3.3 Input/Output Specifications

General-purpose input/output ports are installed on the I/O unit (Model: JANCD-FC810/FC860/FC861) and the JSP board (JSP02/04).

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

Table 3.4 Input/Output Specifications

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



An input/output port is incorporated 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 input: 560 points) is possible.

4

Sequence Control Method

Chapter 4 describes the sequence control method

4.1 Differences In Operation

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

4.1.1 Relay Sequence

All devices are processed simultaneously.

4.1.2 PLC Sequence

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.

Devices are processed sequentially and the ladder is executed repeatedly in a fixed period.

This period is called the scan time.

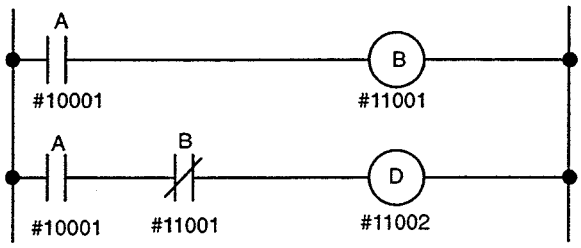


Fig. 4.1.2.1 PLC Sequence Program

The PLC sequence ladder 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 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 Operation of Sequence Program

4.2.1 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: 8 msec
- Low-speed processing scan time: 8 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.

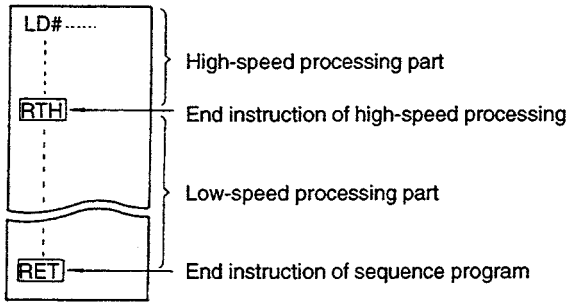


Fig. 4.2 Sequence Program Format

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.

4.2.2 Operation

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

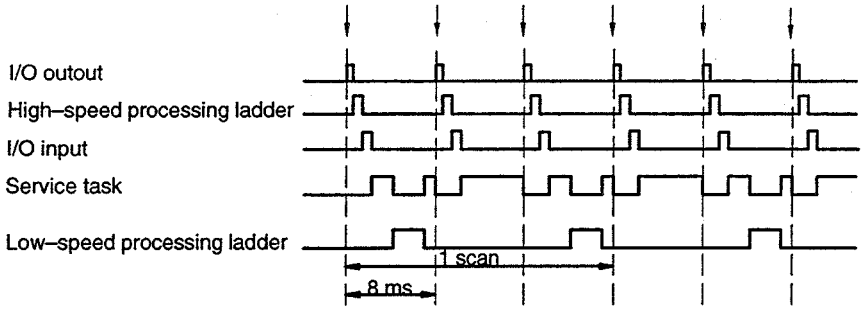


Fig. 4.3 PCNC Operation Time Chart

(1) 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 setting scan time or less as shown in the time charts above.

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

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.

(2) 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 setting interval. That is, the low-speed processing sequence program is executed once in “setting scan time × n”.

Scan time of the low-speed processing sequence program is influenced by the total capacity of the sequence program. As seen above, value “n” will be smaller as the total program capacity and the high-speed processing program capacity are smaller.

For PCNC, NC service task takes priority. If the divided low-speed program is perform-ed in the setting scan section, the remaining program will not be performed in the next section. The execution of low-speed sequence program is determined by system soft-ware, which monitors the load of entire system.

Sequence program size that can be processed in a 8-msec interval is approximately 6000 steps in contact instructions. This size is the total of high-speed processing sequence program and low-speed processing sequence program.

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.

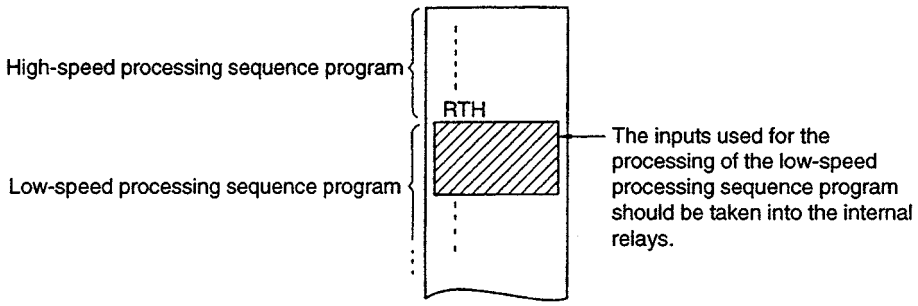


Fig. 4.4 Low-speed Processing Sequence Program

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.

SUPPLEMENT

1. If the results of the high-speed processing sequence program are output to the low-speed processing program, the same consideration must be given to the creation of the program.
2. 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.

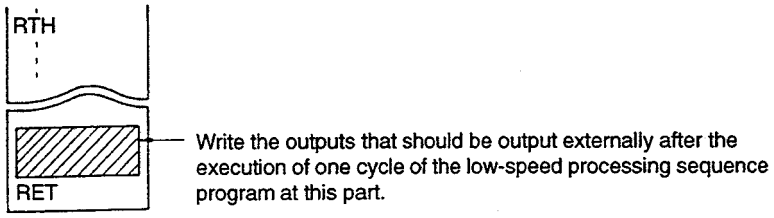


Fig. 4.5 External Output Addresses

(3) Reading and Outputting the State

(a) Reading the input state

After execution of the 8-msec high-speed sequence program, the status of all inputs is read into the PLC collectively.

At the beginning of the 8-msec intervals, the previous output status is output collectively.

4.3 Sequence Program Memory Capacity and Memory Configuration

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

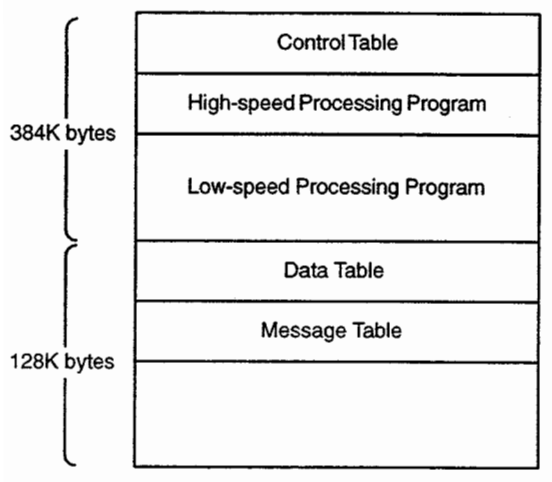


Fig. 4.6 PCNC Program Memory

Generally, relay instructions occupy 4 to 16 bytes and other instruction 1 to 39 bytes. Assuming that one instruction occupies an average of 16 bytes, 384K byte memory area is equivalent to 24K steps (384K/16 = 24K).

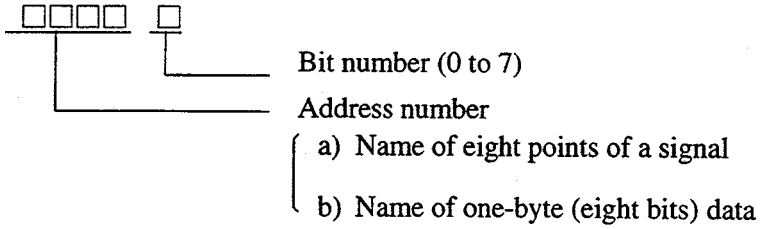
5

Address Numbers And Map

Chapter 5 describes the address numbers and address map.

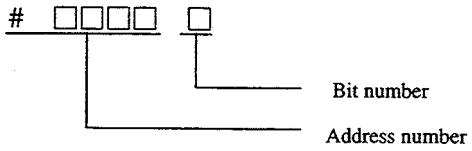
5.1 PLC Address

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



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

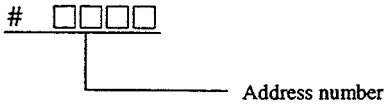
The devices which have one-bit information are designated by a five digit number (address number + bit number) following "#" as indicated below.

Device	Designation
I/O signals Internal relays 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 following "#". The address number is the name assigned to one-byte data.

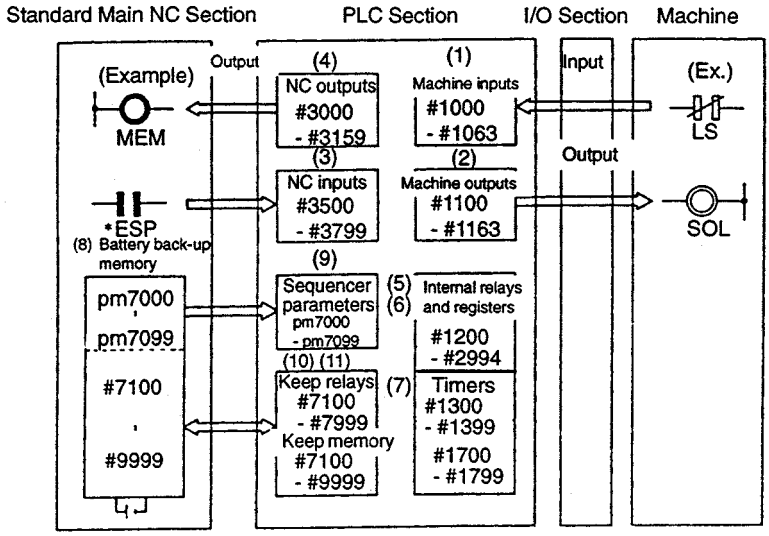
Device	Designation
Registers Timers Sequencer parameters Keep memory	

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 is shown below.



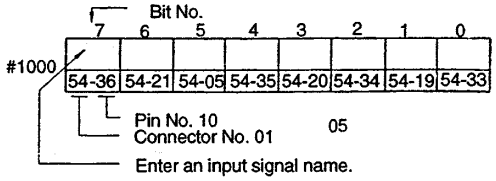
(1) Addresses of Input Signals from the Machine (#1000 to #1063)

For the signals input from the machine operation panel and electric control panel, such as those of push-button 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.

- One bit of address (#1000 to #1063) corresponds to one point of input signals.

  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:



- The input signals of #1000 to #1063 are expressed by the following symbols.

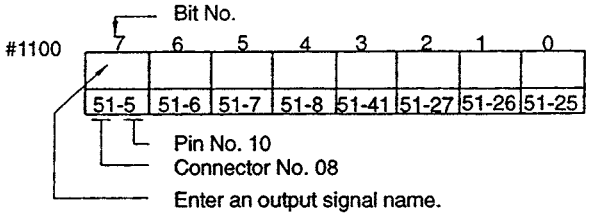
(2) Addresses of Output Signals to the Machine (#1100 to #1163)

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.

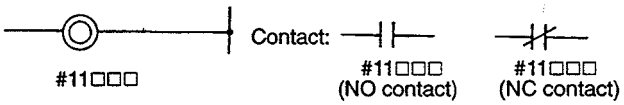
- One bit of address (#1100 to #1163) corresponds to one point of output signals.

  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:



- The output signals of #1100 to #1163 are expressed by the following symbols.



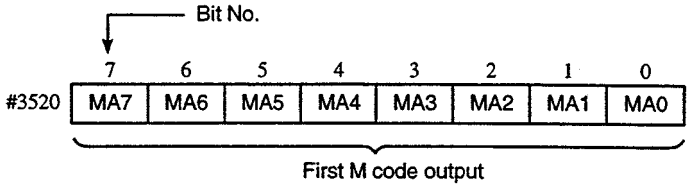
(3) Addresses of Input Signals from the NC's Main Section (#3500 to #3799)

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.

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

  Example:



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

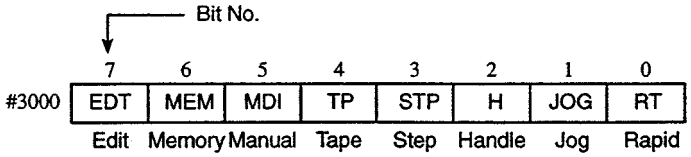
(4) Addresses of Output Signals to the NC Main Section (#3000 to #3159)

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

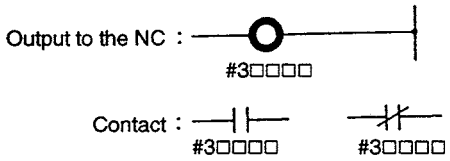
The correspondence between the signal name and the address is determined by the NC and cannot be changed.

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

  Example:



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

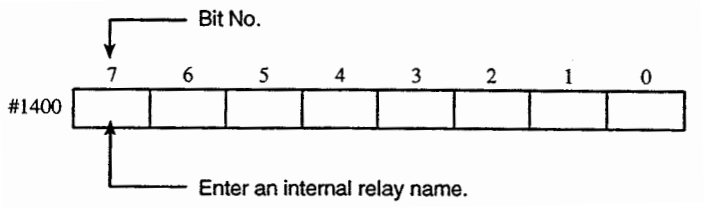


(5) Addresses of Internal Relays (#1200 to #2994; excluding #1300 to #1399 and #1700 to #1799)

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.

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

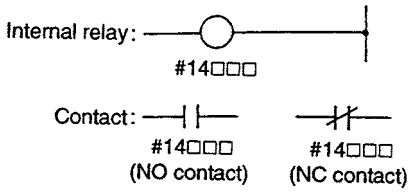
  Example:



- The number of usable internal relays is indicated below.

  500 bytes × 8 bits = 4000 relays

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.

- The address used for registers cannot be used for internal relays.

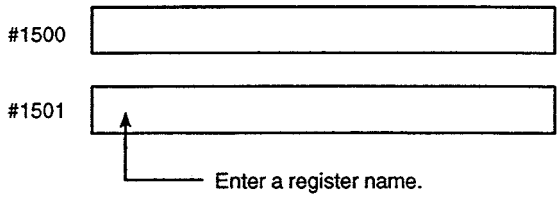
(6) Addresses of Internal Registers (#1200 to #2994; excluding #1300 to #1399 and #1700 to #1799)

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

These registers are used for register instruction and workpiece address for macro instructions,

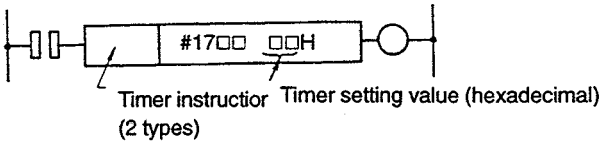
- One address number corresponds to a one-byte register.

  Example:



- For the registers, the address number itself is used as the symbol in a ladder.

  Example:



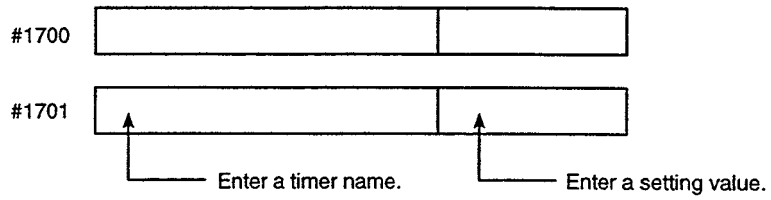
- The address used for internal relays cannot be used for registers.

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

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

- One address number corresponds to a timer.

  Example:



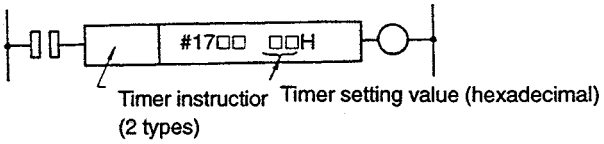
The number of available timers and timer setting units are indicated below.

Table 5.1 Available Timer and Setting Units

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

- An example of a timer symbol is indicated below.

  Example:



(8) Address of 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.

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

- Sequence parameters:pm7000 to pm7099
- Keep relays: #7100 to #7999
- Keep memory: #8000 to #9999

(a) Sequencer Parameter Data

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.

Even if 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 sequence parameter data. Do not attempt to change the data.

(b) Data in the Keep Relay and Keep Memory Data

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.

In the PCNC PLC sequence ladder program, the battery back-up memory can be read and written directly.

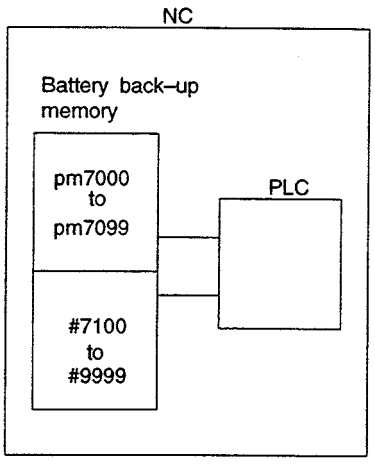


Fig. 5.1 Handling of PCNC Sequence Program

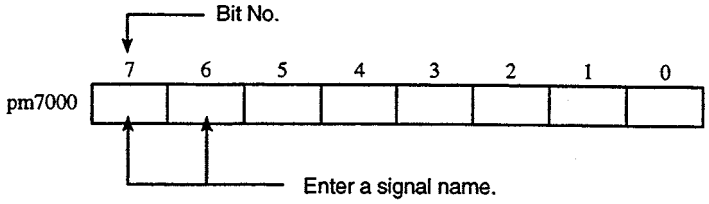
(9) Addresses of Sequencer Parameters (pm7000 to pm7099)

For the sequencer parameters, address pm #7000 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.

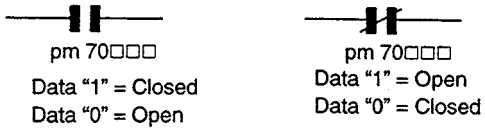
- To use as one-bit data
- To use as one-byte data

(a) Using as one-bit data

Example:

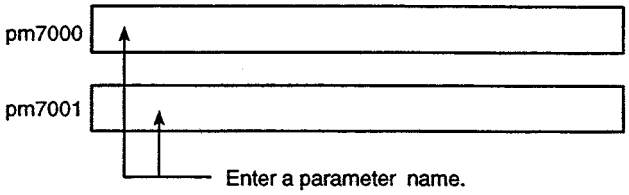


The symbol used in the ladder is indicated below.



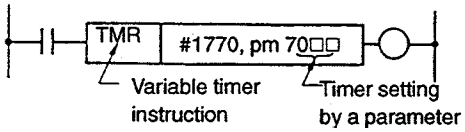
(b) Using as one-byte data

Example:



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.

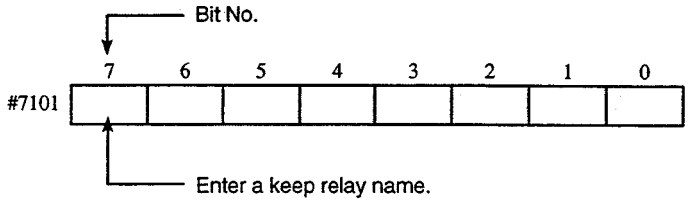


(10) Addresses of Keep Relays (#7100 to #7999)

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

- One bit corresponds to one piece of keep relay.

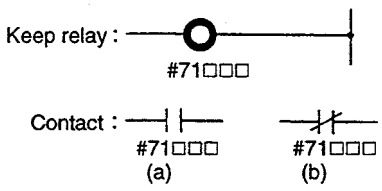
  Example:



- The number of usable keep relays is indicated below.

  900 bytes × 8 bits = 7200 relays

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



(11) Address of Keep Memory (#8000 to #9999)

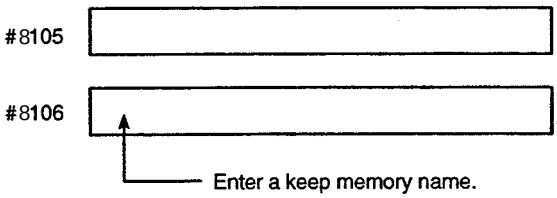
For the one-byte keep memory where the data can be retained after power OFF, addresses #8000 to #9999 are assigned. With the exception that the keep memory can retain the saved data, it can be used in the same manner as with the registers.

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.

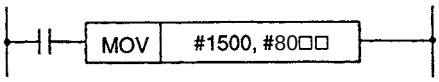
- For one-byte (8 bits) keep memory, address number #8000 or above is assigned.

  Example:



  2900 in the range from #8000 to #9999

- For the keep memory, the address number itself is used as the symbol.

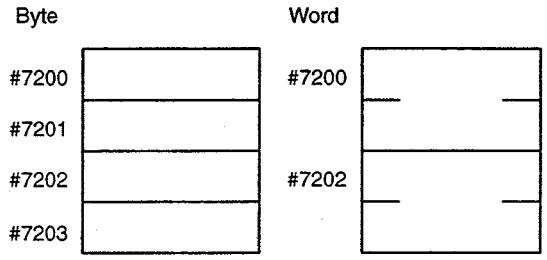


MOV: Contents of register #1500 are transferred to keep memory #80□□.

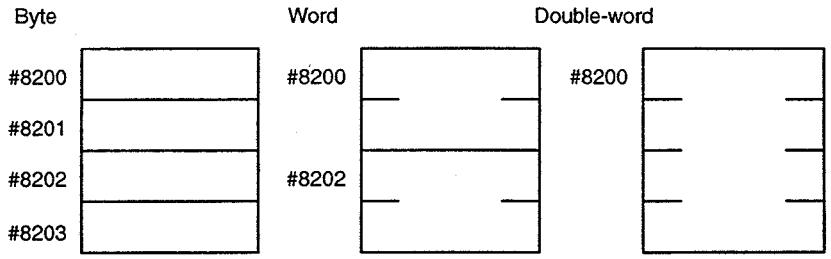
- For the devices that have one-byte or larger information, a four-digit address number is assigned.

  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 indicated below.

① In MMI environment, select UTILITIES.

② From Utilities screen, select LOGIN.

③ To initialize values for Keep Relays and Keep Memory you must log in at Machinist Level or Higher. Also, at the bottom of the Login screen "Enable Parameter Change" option must be selected.

④ Now, select the "Maintenance" screen and hit "PLC Diagnostics"

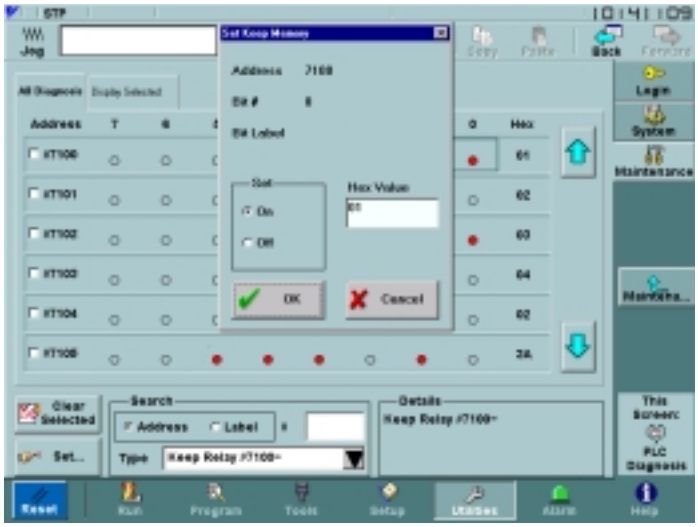
⑤ In this screen, simply search for the keep relay or keep memory that you wish to initiate. Once you have found your address, simply click on it to select it.

⑥ Once you have selected your address bit, click on "SET".

PLC Diagnostics Screen

Example: Writing a decimal number for bit data.

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



Click on “SET” to change keep relay or keep memory address.

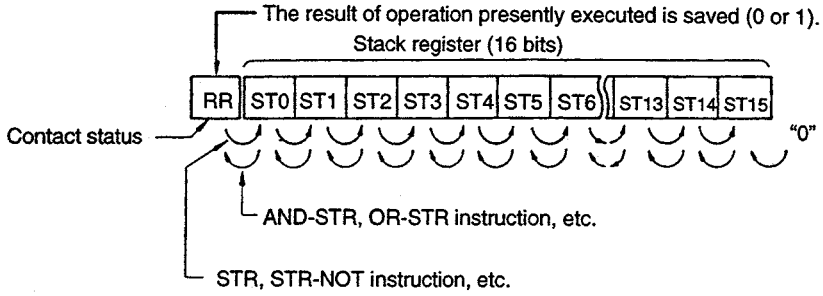
6

PLC Instructions

Chapter 6 describes the PLC instructions.The PLC can use basic instructionsand macro instrutions. Explanation is given for the function and display symbols.The list of coded instructions is also given.

6.1 Registers

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



6.1.1 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.

6.1.2 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 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

6.2.1 Types of Instructions

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

(1) Basic Instructions

Relay instruction: 13 types

Register instruction: 37 types

Timer instruction: 2 types

Control instruction: 9 types

Total 61 types

(2) Macro Instructions

Macro instruction: 22 types

Auxiliary instruction: 5 types

6.2.2 List of Instructions

(1) List of Relay Instructions

The list of relay instructions is indicated below.

Instruction	Description	RR after Operation	Reference Page
LD	Reads the signal status (0, 1) and sets it to RR.	⇳	
LD-NOT	Reads the inversion of signal status (0, 1) and sets it to RR.	⇳	
AND	Executes AND between the contact and RR, sets the result to RR. (Logical product)	⇳	
AND-NOT	Executes AND between the inversion of signal status and RR, and sets the result to RR. (Reverse logical product.)	⇳	
OR	Executes OR between the contact and RR, sets the result to RR. (Logical sum)	⇳	
OR-NOT	Executes OR between the inversion of signal status and RR, and, sets the result to RR. (Reverse logical sum)	⇳	
XOR	Sets “not-coincide” between the signal and RR to RR.	⇳	
XNR	Sets “coincide” between the signal and RR to RR.	⇳	
STR	Enters the content of RR to the stack register and executes the LD instruction.	⇳	
STR-NOT	Enters the content of RR to the stack register and executes the LD-NOT instruction.	⇳	

AND-STR	Executes AND between RR and the stack resister and sets the result to RR.	⇳	
OR-STR	Executes OR between RR and the stack resister and sets the result to RR.	⇳	
OUT	Writes the result of operation (RR) to the rely (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.

(2) List of Timer Instructions

The list of timer instruction is indicated below.

Instruction	Description	RR after Operation	Reference Page
TIM	Timer processing (fixed timer)	time up = 1	
TMR	Timer processing (variable timer)	time up = 1	

(3) List of Register Instructions

The list of register instruction is indicated below.

Instruction	Description	RR after Operation	Reference Page
INR	Adds “+1” to the register content.	—	
DCR	Adds “-1” to the register content.	—	
CLR	Clears the content of the register.	—	
CMR	Inverts the content of the register.	—	
ADI	Adds a numeric value to the content of the register.	—	
SBI	Subtracts a numeric value from the content of the register.	—	
ANI	AND operation between a numeric value and the content of the register.	—	
ORI	OR operation between a numeric value and the content of the register.	—	
XRI	XOR operation between a numeric value and the content of the register.	—	
DEC	Coincidence between a numeric value and the content of the register.	—	
COI	Coincidence between a numeric value and the content of the register.	—	
CMP	Compares a numeric value to the content of the register.	—	

CPI	Compares a numeric value to the content of the register.	—	
MVI	Loads a numeric value to the register.	—	
ADD	Executes addition between register R1 and register R2 and stores the result to R2.	—	
SUB	Executes subtraction between register R1 and register R2 and stores the result to R2.	—	
ANR	Executes AND between register R1 and register R2 and stores the result to R2.	—	
ORR	Executes OR between register R1 and register R2 and stores the result to R2.	—	
XRR	Executes XOR between registers R1 and R2 and stores the result to R2.	—	
CPR	Executes comparison between registers R1 and R2 and stores the result to RR.	⇳	
COR	Executes comparison between registers R1 and R2 and stores the “coincide” result to RR.	⇳	
MOV	Transfers the content of the register R1 to Register R2.	—	
DST	Executes AND between a numeric value and the content of the register R1 and transfers the result to register R2.	—	
DIN	Extracts data.	—	
ADC	Executes double-length addition.	⇳	
ADDW	Executes addition between double-length register (WR2) and double-length register (WR1) and stores the result to WR2.	—	
SUBW	Subtracts the content of double-length register (WR1) from the content of double-length register (WR2) and stores the result to WR2.	—	
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.	
DIVW	Divides content of double-length register (WR1) by the content of double-length register (WR2) and stores the result to WR2.	—	
INRW	Adds “+1” to the content of the double-length register.	—	
DCRW	Adds”-1” to the content of the double-length register.	—	
CLRW	Clears the content of the double-length register to “0’.	—	
CMRW	Inverts the content of the double-length register.	—	
CORW	Executes comparison between double-length register R1 and double-length register R2 and stores the “coincide” result to RR.	⇳	

CPRW	Executes comparison between double-length register R1 and double-length register R2 and stores the result to RR.	⇳	
MVIW	Loads a numeric value to the double-length register.	—	
DSTW	Executes AND between the content of double-length register (WR1) and a numeric value and transfers the result to double-length register (WR2).	—	

(4) List of Control Instructions

The list of control instructions is indicated below.

Instruction	Description	RR after Operation	Reference Page
NOP	No operation	—	
MCR	Start of master control relay	—	
END	End of master control relay	—	
RET	End of sequence program	—	
RTI	Executes the RET instruction if “RR = 1”.	—	
SET	Sets “1” to RR	1	
RTH	End of high-speed processing sequence program	—	
JMP	Executes jump to the location indicated by ADR.	—	
ADR	Indicates the location of destination of jump indicated by JMP.	—	

(5) List of Macro Instructions

The list of macro instructions is indicated below.

Instruction	Description	RR after Operation	Reference Page
SUBP003	Detects the rising edge of the signal.	⇳	
SUBP004	Detects the falling edge of the signal.	⇳	
SUBP005	Counter	⇳	
SUBP006	Rotation (for the control of rotating object)	⇳	
SUBP007	Code conversion	⇳	
SUBP009	Pattern clear	⇳	
SUBP011	Parity check	⇳	
SUBP014	Data conversion (binary ⇆ BCD)	⇳	
SUBP017	Data search	⇳	
SUBP018	Index data transfer	⇳	

SUBP023	Message display (option)	⇳	
SUBP025	Binary decode processing	⇳	
SUBP027	Binary code conversion	⇳	
SUBP031	Expansion data transfer	⇳	
SUBP032	Binary conversion	⇳	
SUBP034	Binary data search	⇳	
SUBP035	Binary index modifier data transfer	⇳	
SUBP036	Binary addition	⇳	
SUBP037	Binary subtraction	⇳	
SUBP038	Binary multiplication	⇳	
SUBP039	Binary division	⇳	
SUBP040	Binary constant definition	⇳	

(6) List of Auxiliary Instructions

The list of auxiliary instructions is indicated below.

Instruction	Description	RR after Operation	Reference Page
IPSH	Designation of a numeric value used by SUBP instruction	—	
APSH	Designation of the address of a register used by SUBP instruction.	—	
PUSH	Designation of the address of a register used by SUBP instruction.	—	
TPSH	Designation of the table number of a PC table used by SUBP instruction.	—	
IPSHD	Designation of the data used by SUBP instruction	—	

6.3 Details of Instructions

6.3.1 Relay Instructions

The relay instructions are described below.

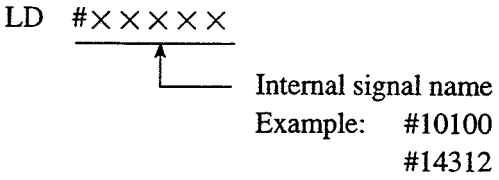
(1) LD (Load) RR after operation: RR⇳

(a) Function

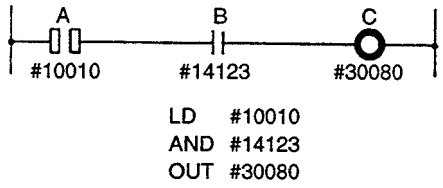
Reads the signal status (1 or 0) and sets it to RR.

Normally, the instruction is used for NO contact.

(b) Format



(c) Example



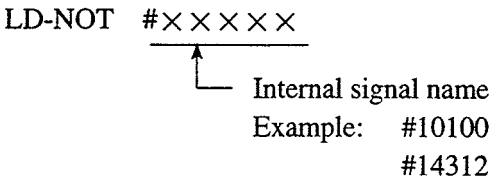
(2) LD-NOT (Load Not) RR after operation: RR⇳

(a) Function

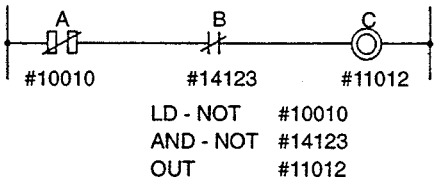
Reads the signal status (1 or 0) and sets it to RR.

Normally, the instruction is used for NC contact.

(b) Format



(c) Example

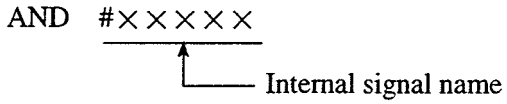


(3) AND RR after operation: RR⇳

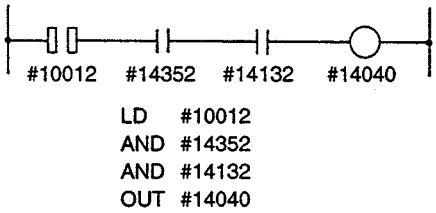
(a) Function

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

(b) Format



(c) Example

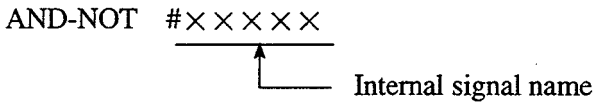


(4) AND NOT RR after operation: RR⇳

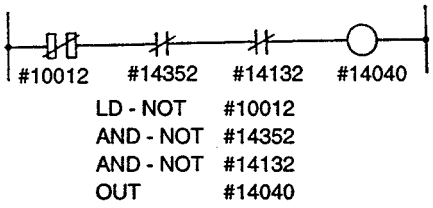
(a) Function

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

(b) Format



(c) Example

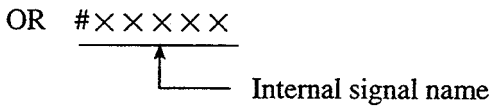


(5) OR RR after operation: RR⇳

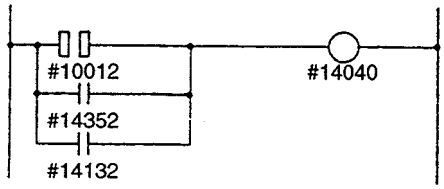
(a) Function

Executes OR between the contact and RR and sets the result to RR (logical sum).

(b) Format



(c) Example



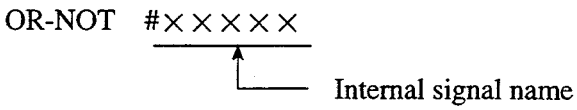

```
LD   #10012
OR   #14352
OR   #14132
OUT  #14040
```

(6) OR-NOT RR after operation: RR⇳

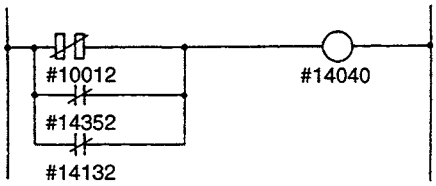
(a) Function

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

(b) Format



(c) Example




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

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

(a) Function

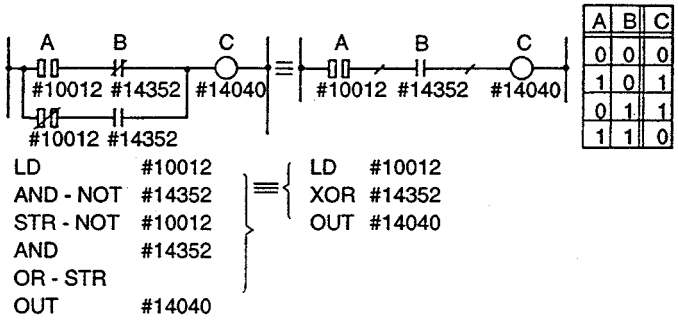
Sets “not agree” between contact and RR to RR.

(b) Format

XOR #×××××

Internal signal name

(c) Example



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

(a) Function

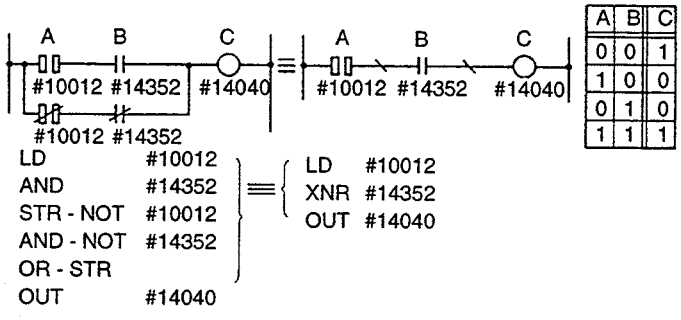
Sets “agree” between contact and RR to BR.

(b) Format

XNR #×××××

Internal signal name

(c) Example

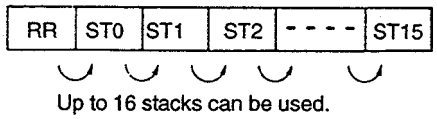


(9) STR (Store) RR after operation: RR⇳

(a) Function

Sets the contents of RR to stack and executes the LD instruction.

Normally, the instruction is used for NO contact.

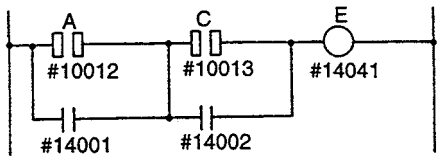


(b) Format

STR #×××××

Internal signal name

(c) Example




```
LD          #10012
OR          #14001
STR         #10013
OR          #14002
AND - STR
OUT         #14041
```

(10) STR-NOT (Store NOT) RR after operation: RR⇳

(a) Function

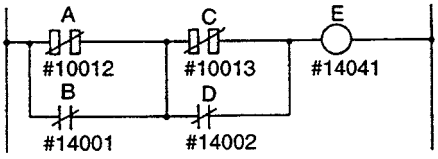
Sets the contents of RR to stack and executes the LD-NOT instruction.

(b) Format

STR-NOT #×××××

Internal signal name

(c) Example




```
LD - NOT    #10012
OR - NOT    #14001
STR - NOT   #10013
OR - NOT    #14002
AND -STR
OUT         #14041
```

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

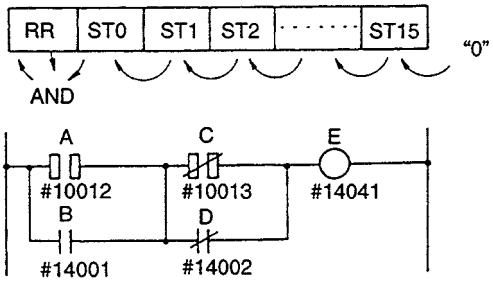
(a) Function

Executes AND between RR and stack (ST0) and sets the result to RR.
Stacks shift to the left by one.

(b) Format

AND-STR

(c) Example




```
LD          #10012
OR          #14001
STR - NOT   #10013
OR - NOT    #14002
AND -STR
OUT         #14041
```

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

(a) Function

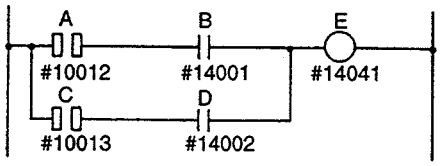
Executes OR between RR and stack (ST0) and sets the result to RR.

Stacks shift to the left by one.

(b) Format

OR-STR

(c) Example



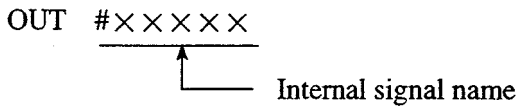

```
LD        #10012
AND       #14001
STR       #10013
AND       #14002
OR - STR
OUT       #14041
```

(13) OUT RR after operation: RR⇳

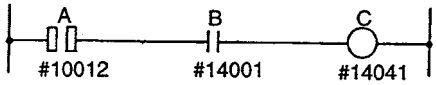
(a) Function

Writes the result of operation (RR) to the relay.

(b) Format



(c) Example




```
LD    #10012
AND   #14001
OUT   #14041
```

6.3.2 Timer Instructions

The timer instructions are described below.

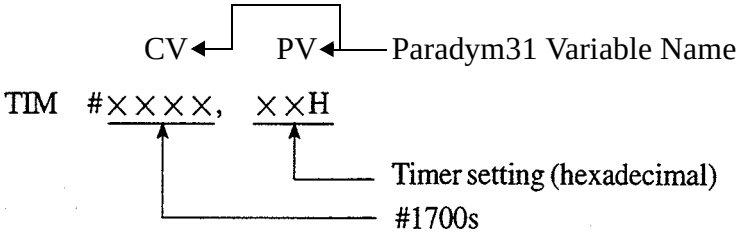
(1) TIM (Fixed Timer) RR = 1 at time-up

(a) Function

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.

(b) Format



(c) Explanation

- 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.

- The following five types of timers can be used.

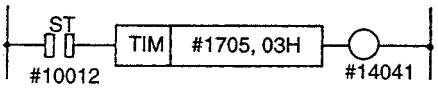
Address No.	Timer Type	Number of Timers
#1700 to #1709, #1760 to #1769 #1300 to #1309, #1360 to #1369	1 = 8 msec	40
#1710 to #1729, #1790 to #1799 #1310 to #1329, #1390 to #1399	1 = 0.1 sec	60
#1730 to #1749, #1780 to #1789 #1330 to #1349, #1380 to #1389	1 = 50 msec	60
#1750 to #1759, #1350 to #1359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

- To use the timers of #1300 to #1399, the compile of Ver. 3.5 or later necessary.

(d) Example




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

SUPPLEMENT

Do not use the same address for both a fixed timer and a variable timer.

If used, correct operation cannot be guaranteed.

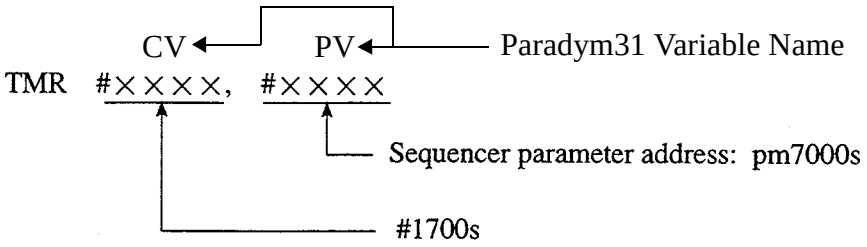
(2) TMR (Variable Timer) RR = 1 at time-up

(a) Function

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.

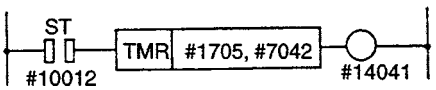
(b) Format



(c) Explanation

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

(d) Example:




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

SUPPLEMENT

Do not use the same address for both a fixed timer and a variable timer.

If used, correct operation cannot be guaranteed.

6.3.3 Register Instructions

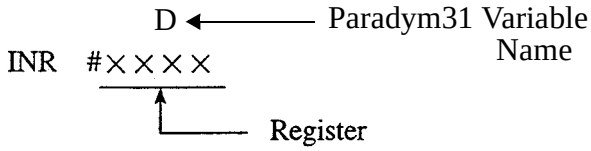
The register instructions are described below.

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

(a) Function

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.

(b) Format

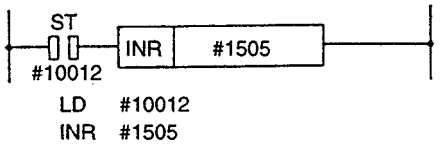


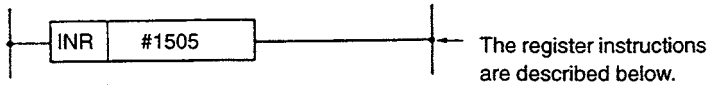
(c) Explanation

- An ST contact must be entered before the INR instruction.
- The INR instruction is executed in intervals of "4 × n" msec while the ST contact is ON.
- There is no function to detect overflow. If a timer counts to FFH, it returns to 0H.

(d) Example

The following examples show when the register of #1500s is used.



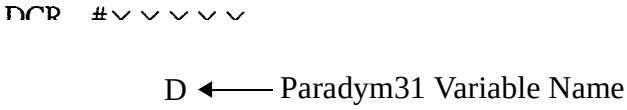


(2) DCR (Decrement Register) RR after operaiton: RR-

(a) Function

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.

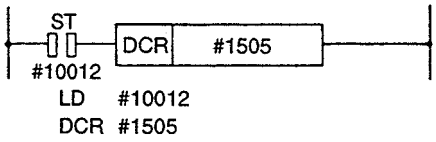
(b) Format



(c) Explanation

- An ST contact must be entered before the DCR instruction.
- The DCR instruction is executed in intervals of "4 × n" msec while the ST contact is ON.
- There is no function to detect overflow. If a timer counts to 0H, it returns to FFH.

(d) Example



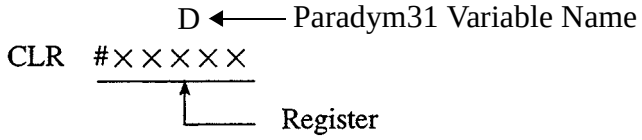
(3) CLR (Clear) RR after operation: RR-

(a) Function

The instruction clears the content of the register "0" when the ST contact is ON (RR = 1). If the ST contact is OFF (RR = 0), the content of the register is not cleared.

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

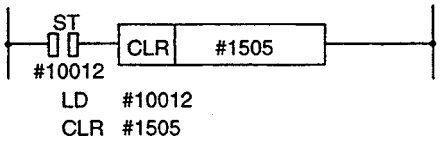
(b) Format



(c) Explanation

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

(d) Example



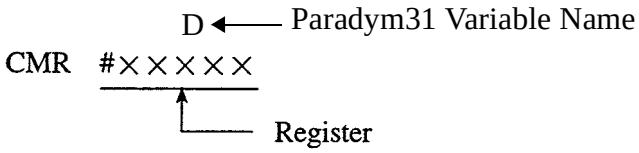
(4) CMR (Complement Register) RR after operation: RR-

(a) Function

The instruction reverses the content of the register when the ST contact is ON (RR = 1). If the ST contact is OFF (RR = 0), the instruction does not reverse the content of the register.

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

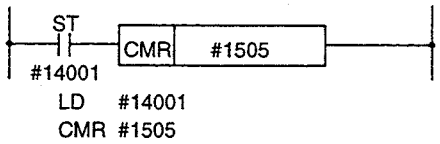
(b) Format



(c) Explanation

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

(d) Example



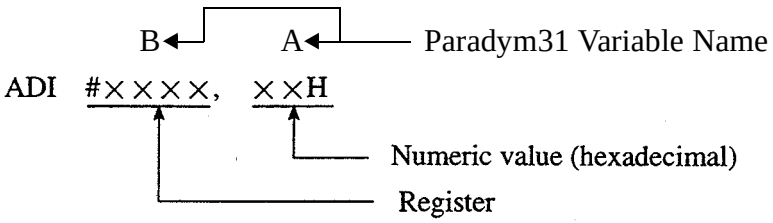
(5) ADI (Add Immediate) RR after operation: RR-

(a) Function

The instruction adds the specified numeric value to 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 remains unchanged before and after the execution of the ADI instruction.

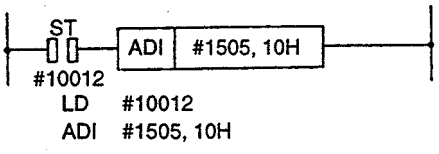
(b) Format



(c) Explanation

- An ST contact must be entered before the ADI instruction.
- The ADI instruction is executed in intervals of “4 × n” msec while the ST contact is ON.
- There is no function to detect overflow. Make sure that the result will not exceed FFH.

d) Example



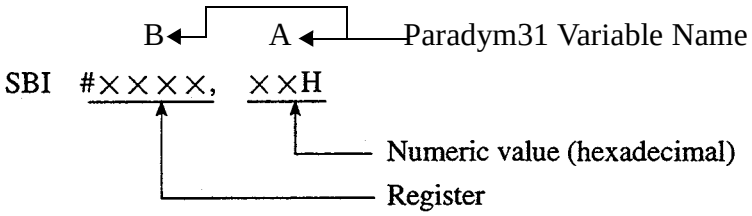
(6) SBI (Subtract Immediate) RR after operation: RR-

(a) Function

The instruction subtracts the specified numeric value from 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 SBI instruction.

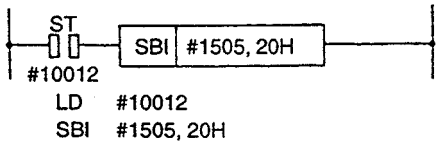
(b) Format



(c) Explanation

- An ST contact must be entered before the SBI instruction.
- The SBI instruction is executed in intervals of "4 × n" msec while the ST contact is ON.
- There is no function to detect underflow. Make sure that "numeric value contents of register".

(d) Example



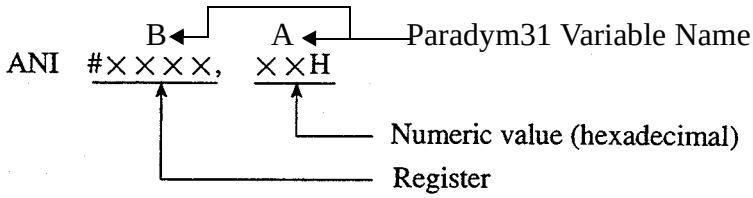
(7) ANI (AND Immediate) RR after operation: RR-

(a) Function

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.

(b) Format

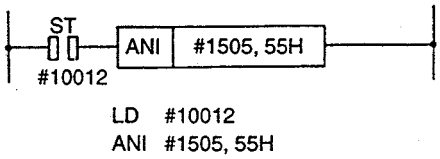


(c) Explanation

- An ST contact must be entered before the ANI instruction.
- The ANI instruction is executed in intervals of "4 × n" msec while the ST contact is ON.
- The relationship between register and numeric value is shown below".

	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

(d) Example



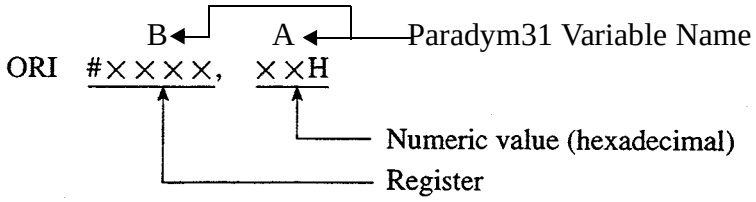
(8) ORI (OR Immediate) RR after operation: RR-

(a) Function

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.

(b) Format

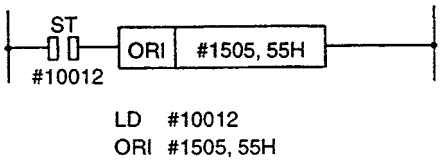


(c) Explanation

- An ST contact must be entered before the ORI instruction.
- The ORI instruction is executed in intervals of “4 × n” msec while the ST contact is ON.
- The relationship between register and numeric value is shown below”.

	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

(d) Example



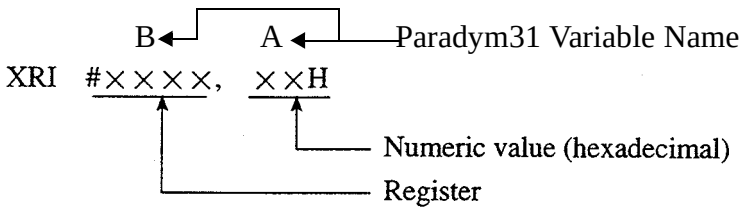
(9) XRI (XOR Immediate)

(a) Function

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.

(b) Format



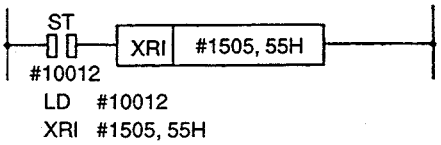
(c) Explanation

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

- The relationship between register and numeric value is shown below".

	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

(d) Example

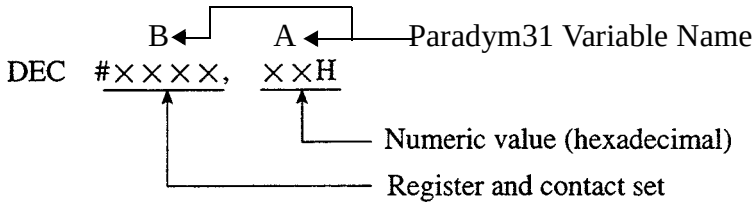


(10)DEC (Decode) RR after operation: RR⇳

(a) Function

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.

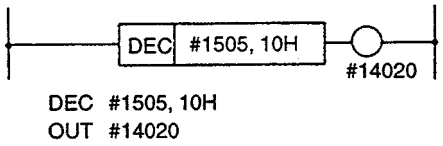
(b) Format



(c) Explanation

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

(d) Example



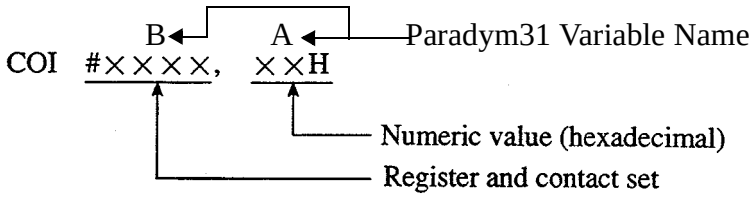
(11)COI (Coincide Immediate) RR after operation: RR⇳

(a) Function

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 comparisons is “coincide”.

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

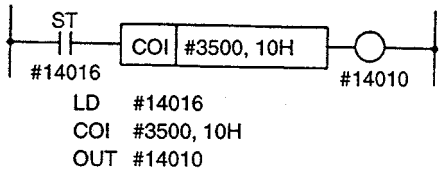
(b) Format



(c) Explanation

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

(d) Example



(12)CMP (Compare) RR after operation: RR⇳

(a) Function

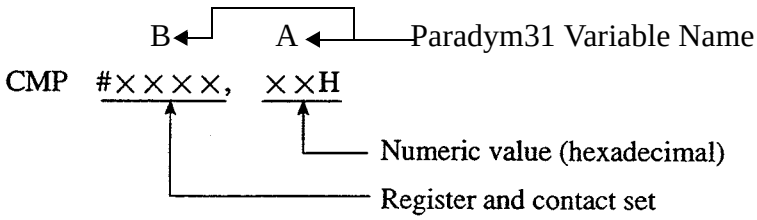
The instruction compares the numeric value to the eight-bit data of the register or contact set and sets “1” or “0” depending on the result of comparison.

Register (contact) ≥ Numeric value: RR = 1

Register (contact) < Numeric value: RR = 0

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

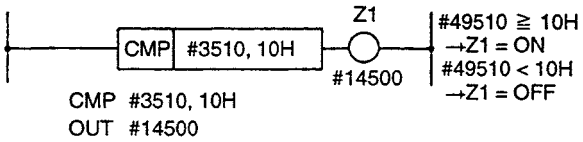
(b) Format



(c) Explanation

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

(d) Example



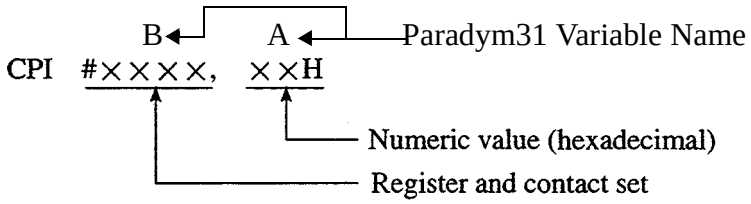
(13)CPI (Compare) RR after operation: RR⇳

(a) Function

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 "Register (contact) ≥ Numeric value".

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

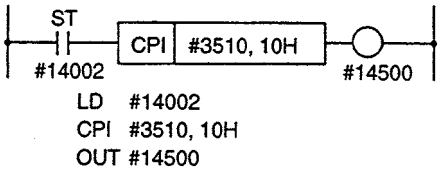
(b) Format



(c) Explanation

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

(d) Example

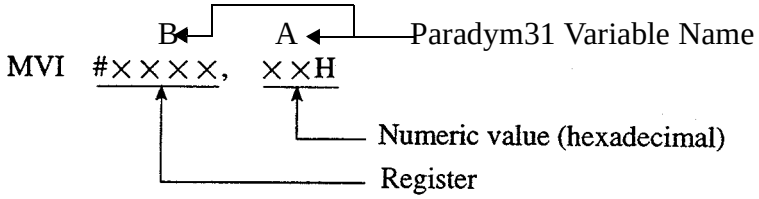


(14)MVI (Move Immediate) RR after operation: RR⇳

(a) Function

The function transfers the numeric value to the register when the ST contact is ON (RR = 1). If the ST contact is OFF (RR = 0), the instruction is not executed.

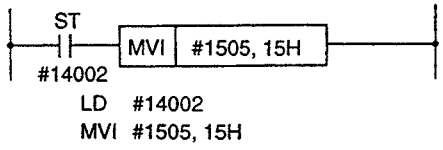
(b) Format



(c) Explanation

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

(d) Example



(15)ADD (Add Register) RR after operation: RR⇳

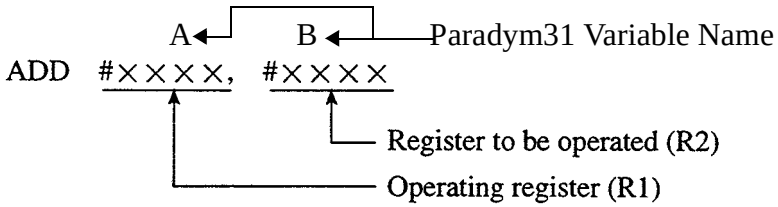
(a) Function

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

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

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

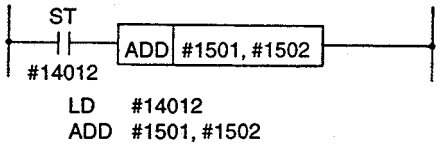
(b) Format



(c) Explanation

- A ST contact must be entered before the ADD instruction.
- The ADD instruction is executed in intervals of "4 × n" msec while the ST contact is ON.
- There is no function to detect overflow. Make sure that the result will not exceed 255 (FFH).

(d) Example



(16)SUB (SUB Register) RR after operation: RR-

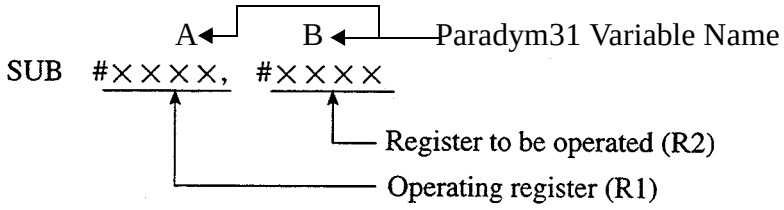
(a) Function

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

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

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

(b) Format



(c) Explanation

There is no function to detect underflow. Make sure that the following is always satisfied: R1≤ R2.

(17)ANR (AND Register) RR after operation: RR-

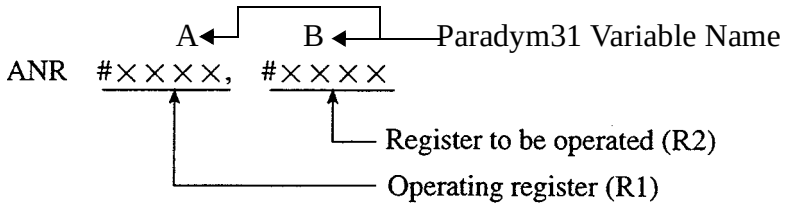
(a) Function

The function executes AND operation between the content in the resister R2 and the content in register R1 when the ST contact is ON (RR = 1) and stores the result to register R2.

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

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

(b) Format



(18)ORR (OR Register) RR after operation: RR-

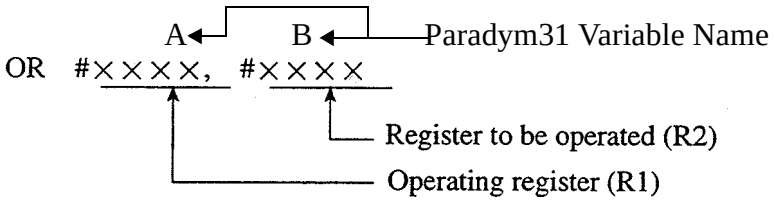
(a) Function

The function executes OR operation between the content in the resister R2 and the content in register R1 when the ST contact is ON (RR = 1) and stores the result to register R2.

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

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

(b) Format



(19)XRR (XOR Register) RR after operation: RR-

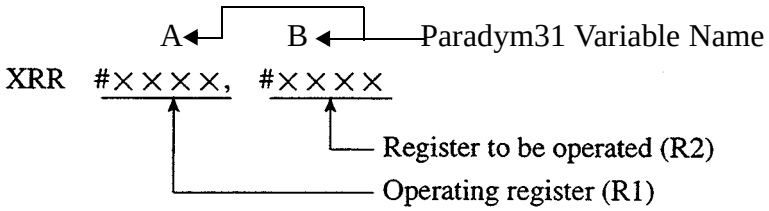
(a) Function

The function executes XOR operation between the content in the resister R2 and the content in register R1 when the ST contact is ON (RR = 1) and stores the result to register R2.

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

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

(b) Format



(20)CPR (Compare Register) RR after operation: RR-

a) Function

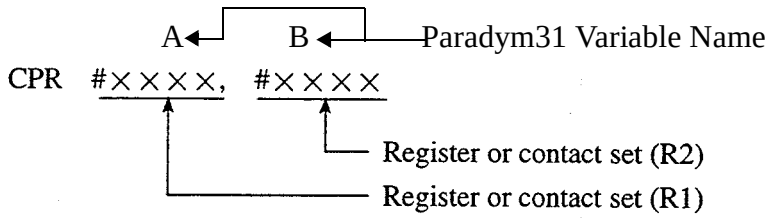
The instruction executes comparison between R1 and R2 when the ST contact is ON (RR = 1), and sets “0” or “1” to Z1 according to the result of comparison.

R1 < R2 Z1 = 0

R1 ≥ R2 Z1 = 1

If the ST contact is OFF (RR = 0), the CPR instruction is not executed.The content of RR remains at “0”.

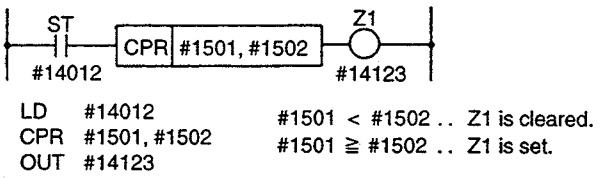
(b) Format



(c) Explanation

- After the execution of the CPR instruction, the contents of R1 and R2 remain unchanged.
- An ST contact must be entered before the CPR instruction.
- The CPR instruction is executed in intervals of “4 × n” msec while the ST contact is ON.

(d) Example



(21)COR (Coincide Register) RR after operation: RR-

(a) Function

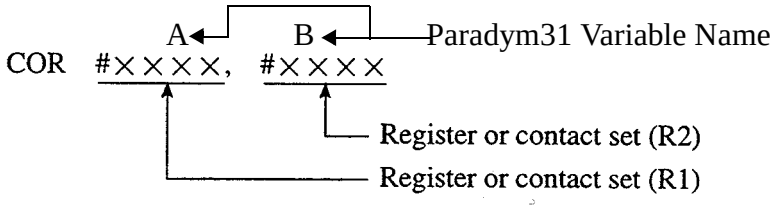
The instruction executes comparison between R1 and R2 when the ST contact is ON (RR = 1), and sets “0” or “1” to Z1 according to the result of comparison.

R1 = R2 Z1 = 0

R1 ≠ R2 Z1 = 1

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

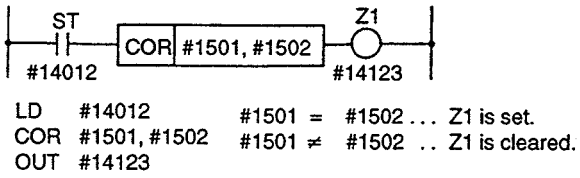
(b) Format



(c) Explanation

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

(d) Example



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

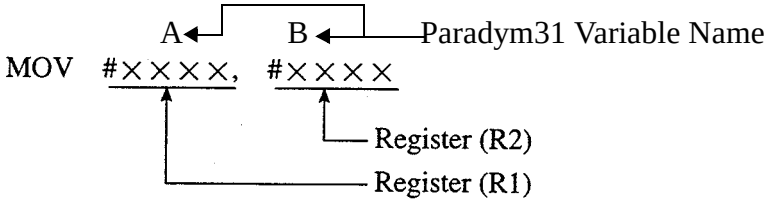
(a) Function

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.

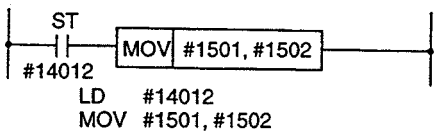
(b) Format



(c) Explanation

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

(d) Example



(23)DST (Data Store) RR after operation: RR-

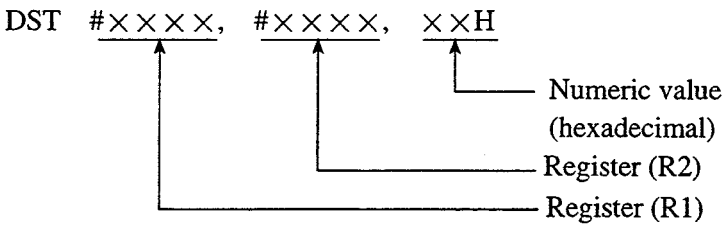
(a) Function

The instruciton 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.

(b) Format



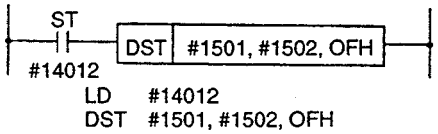
(c) Explanation

- An ST contact must be entered before the DST instruction.
- The DST instruction is executed in intervals of “4 × n” msec while the ST contact is ON.
- The relationship between register and numeric value is shown below.

	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
Result R2	0	0	0	0	B	B	B	B

B: “0” or “1”.

(d) Example



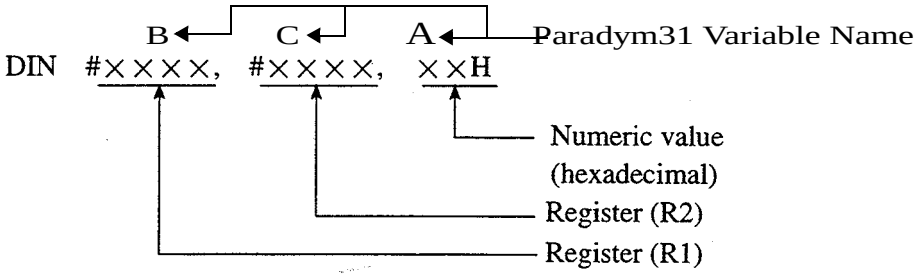
(24) DIN (Data Insert) RR after operation: RR-

(a) Function

The function 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.

(b) Format



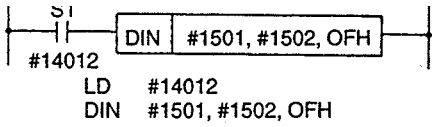
(c) Explanation

- A ST contact must be entered before the DIN instruction.
- The DIN instruction is executed in intervals of "4 × n" msec while the ST contact is ON.
- The relationship between register and numeric value is shown below.

	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"

(d) Example

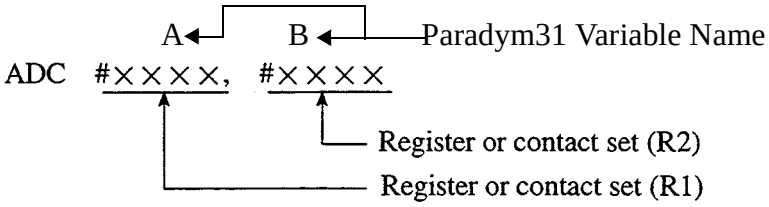


(25)ADC (Add with Carry) RR after operation: RR⇳

(a) Function

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.

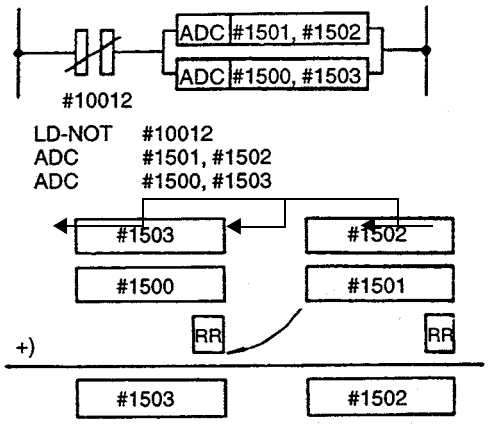
(b) Format



(c) Explanation

- The ADC instruction is executed in intervals of "4 × n" msec while the ST contact is ON.
- To execute the ADC instruction, the content of RR must be "0".

(d) Example



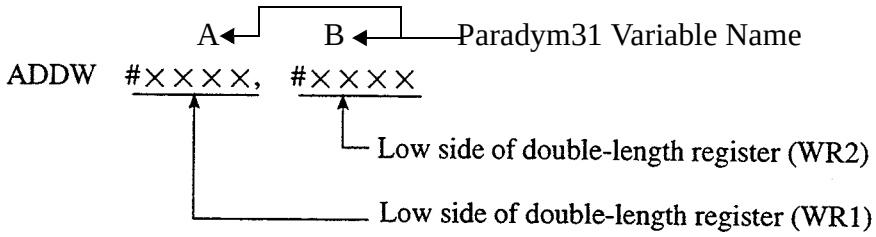
(26)ADDW (ADD Word Register) RR after operation: RR-

(a) Function

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 register (WR2).

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

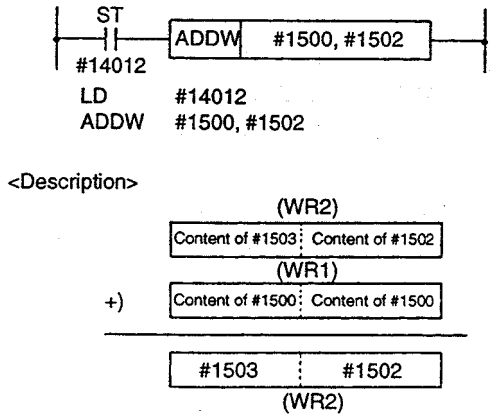
(b) Format



(c) Explanation

- A ST contact must be entered before the ADDW instruction.
- The ADDW instruction is executed in intervals of “4 × n” msec while the ST contact is ON.
- There is no function to detect overflow. Make sure that the result will not exceed FFFFH.

(d) Example



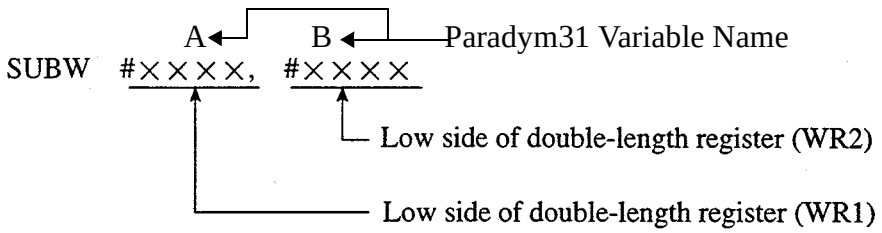
(27)SUBW (SUB Word Register) RR after operation: RR-

(a) Function

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 register (WR2).

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

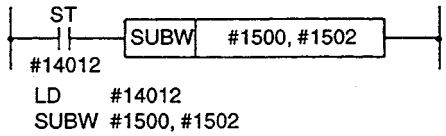
(b) Format



(c) Explanation

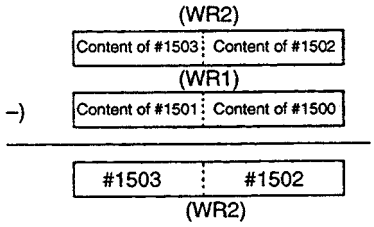
- A ST contact must be entered before the SUBW instruction.
- The SUBW instruction is executed in intervals of “4 × n” msec while the ST contact is ON.
- There is no function to detect underflow. Make sure that the following is always satisfied WR1 ≤ WR2.

(d) Example




```
LD    #14012
SUBW  #1500, #1502
```

<Description>



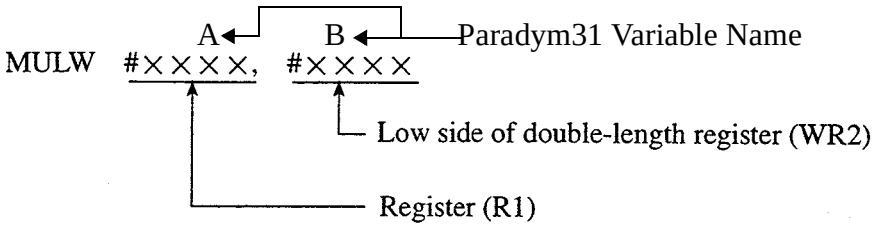
(28)MULW(MUL Word Register) RR after operation: RR⇳

(a) Function

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

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

(b) Format

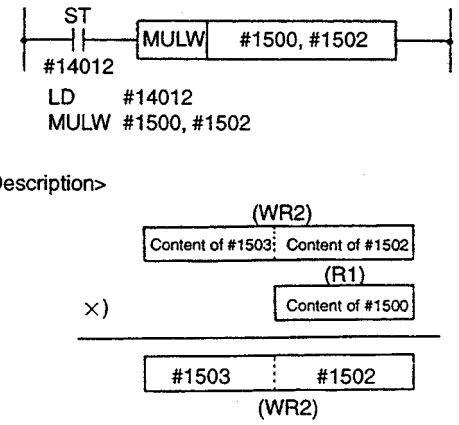


(c) Explanation

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

(d) Example



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

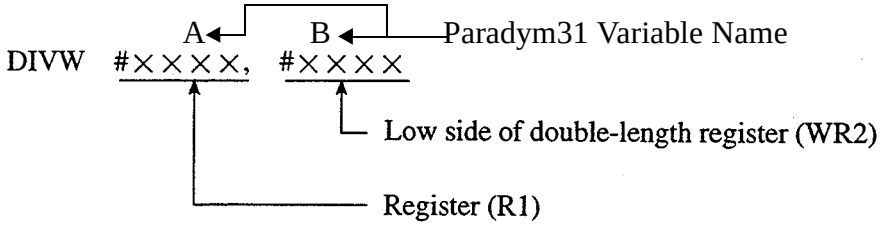
(a) Function

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.

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

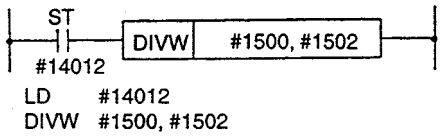
(b) Format



(c) Explanation

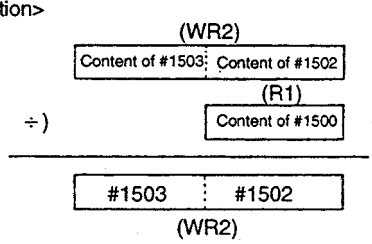
- A ST contact must be entered before the DIVW instruction.
- The DIVW instruction is executed in intervals of “4 × n” msec while the ST contact is ON.
- The instruction is not executed if the content of R1 is “0”.

(d) Example




```
LD    #14012
DIVW  #1500, #1502
```

<Description>

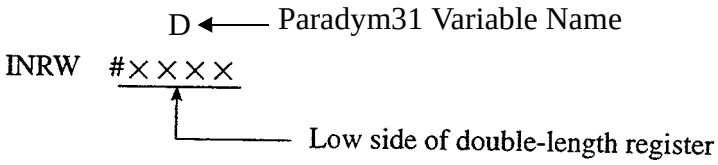


(30)INRW (Increment Word Register) RR after operation: RR-

(a) Function

The instruction adds "+1" to the content of the double-length register when the ST content is ON (RR = 1)

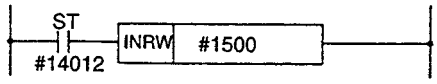
(b) Format



(c) Explanation

- A ST contact must be entered before the INRW instruction.
- The INRW instruction is executed in intervals of "4 × n" msec while the ST contact is ON.
- There is no function to detect overflow. If the result of operation exceeds FFH, it returns to 0H.

(d) Example



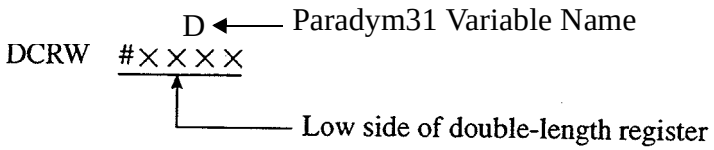

```
LD    #14012
INRW  #1500
```

(31)DCRW (Decrement Word Register) RR after operation: RR-

(a) Function

The instruction adds "-1" to the content of the double-length register when the ST content is ON (RR = 1).

(b) Format



(c) Explanation

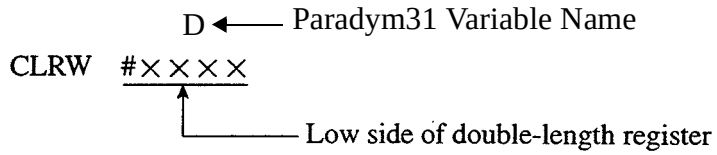
- A ST contact must be entered before the INRW instruction.
- The DCRW instruction is executed in intervals of "4 × n" msec while the ST contact is ON.
- There is no function to detect overflow. If the result of operation exceeds 0H, it returns to FFFH.

(32)CLRW (Clear Word Register) RR after operation: RR-

(a) Function

The instruction clears the content of double-length register when the ST content is ON (RR = 1)

(b) Format



(c) Explanation

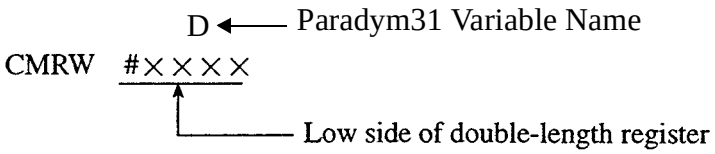
- A ST contact must be entered before the CLRW instruction.
- The CLRW instruction is executed in intervals of "4 × n" msec while the ST contact is ON.
- There is no function to detect underflow. If the result of operation exceeds 0H, it returns to FFFH.

(33)CMRW (Complement Word Register) RR after operation: RR-

(a) Function

The instruction reverses the content of double-length register when the ST content is ON (RR = 1)

(b) Format



(c) Explanation

- A ST contact must be entered before the CMRW instruction.
- The CMRW instruction is executed in intervals of "4 × n" msec while the ST contact is ON.
- There is no function to detect underflow. If the result of operation exceeds 0H, it returns to FFFH.

(34)CORW (Coincide Word Register) RR after operation: RR⇳

(a) Function

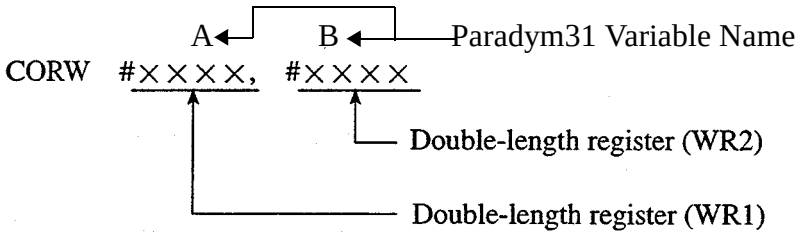
The instruction executes comparison between WR1 and WR2 when the ST content is ON (RR = 1), and sets "0" or "1" to Z1 according to the result of comparison.

WR1 = WR2 Z1= 1

WR ≠ WR2 Z1 = 0

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

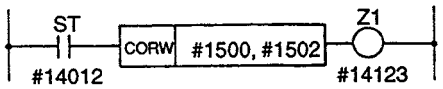
(b) Format



(c) Explanation

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

(d) Example




```
LD      #14012
CORW    #1500, #1502
OUT     #14123
```

#1500 = #1502 .... Z1 is set.
#1500 ≠ #1502 .... Z1 is cleared.

(35)CPRW (Compare Word Register) RR after operation: RR⇳

(a) Function

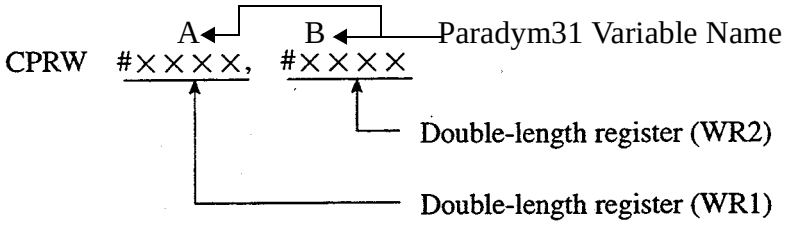
The instruction executes comparison between WR1 and WR2 when the ST content is ON (RR = 1), and sets “0” or “1” to Z1 according to the result of comparison.

WR1 < WR2 Z1 = 0

WR ≥ WR2 Z1 = 1

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

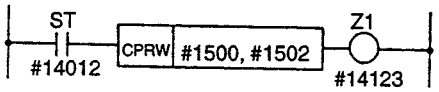
(b) Format



(c) Explanation

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

(d) Example




```
LD      #14012
CPRW    #1500, #1502
OUT     #14123
```

#1500 < #1502 .... Z1 is set.
#1500 ≧ #1502 .... Z1 is cleared.

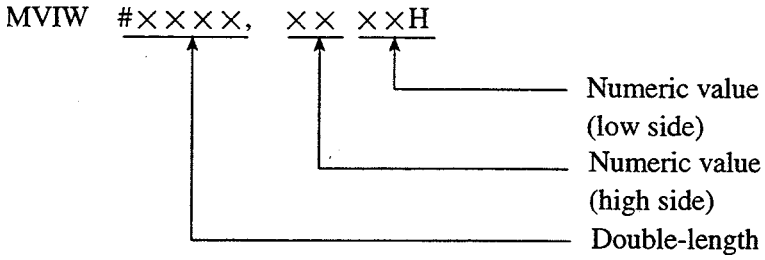
(36)MVIW (Move Immediate Word Register) RR after operation: RR-

(a) Function

The instruction transfers the numeric value to the register when the ST content is ON (RR = 1).

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

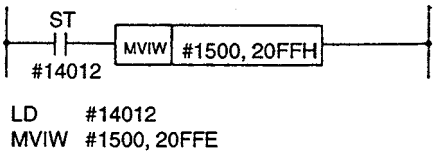
(b) Format



(c) Explanation

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

(d) Example



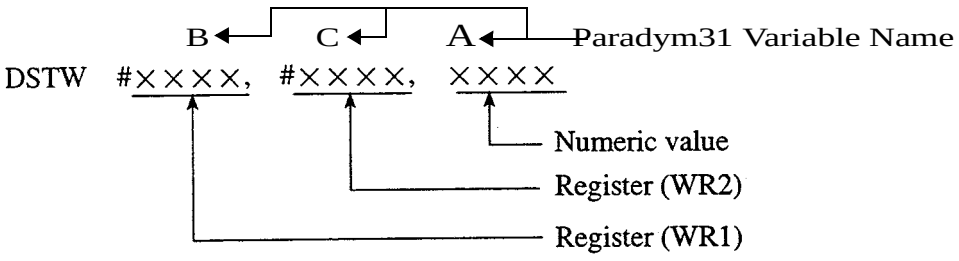
(37)DSTW (Data Store Word Register) RR after operation: RR-

(a) Function

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.

(b) Format



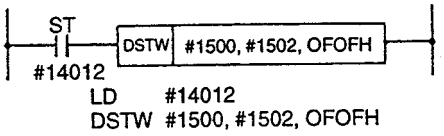
(c) Explanation

- A ST contact must be entered before the DSTW instruction.
- The DSTW instruction is executed in intervals of “4 × n” msec while the ST contact is ON.
- The relationship between register and numeric value is shown below.

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

a: “0” or “1”

(d) Example



6.3.4 Control Instructions

The control instructions are described below.

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

(a) Function

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.

(b) Format

NOR

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

(a) Function

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

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

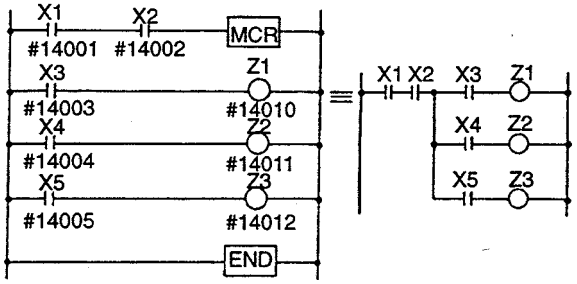
(b) Format

MCR

(c) Explanation

- It is possible to enter another MCR instruction between the MCR and END instructions (max. 7 levels).
- When a timer instruction is included in the MCR instruction, the timer is cleared when the MCR instruction is OFF.
- 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.
- With the MCR instruction, the output coil state is not retained.

(d) Example




```
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) END (Master Control End) RR after operation: RR-

(a) Function

The instruction indicates the end of the MCR instruction.

(b) Format

END

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

(a) Function

The RET instruction indicates the end of a sequence program.

(b) Format

RET

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

(a) Function

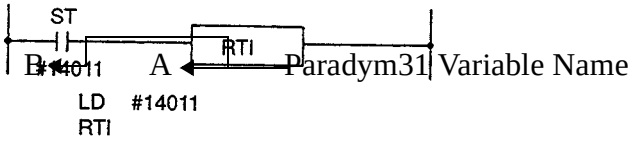
The instruction executes the RET instruction when the ST contact is ON.

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

(b) Format

RTI

(c) Example



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

(a) Function

The instruction forcibly sets “1” for “RR”.

(b) Format

SET

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

(a) Function

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

(b) Format

RTH

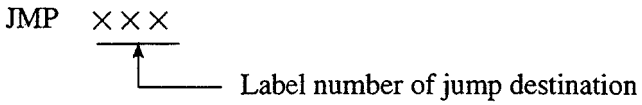
(8) JMP (Jump) RR after operation: RR-

(a) Function

The instruction executes jump to the ADR012 when the ST contact is ON (RR = 1).

If the ST contact is OFF (RR = 0), the ladder of the next step is executed.

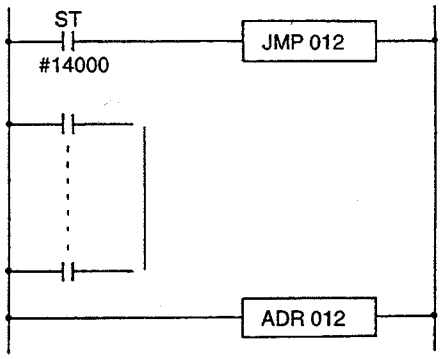
(b) Format



(c) Explanation

With the JMP instruction, the states of output coils up to ADR are retained when RR = 1.

(d) Example



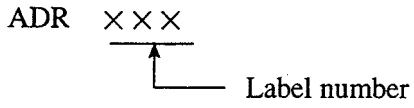

```
LD   #14000
JMP  012
```

(9) ADR (Address) RR after operation: RR-

(a) Function

The instruction indicates the destination of jump called up by the JMP instruction.

(b) Format



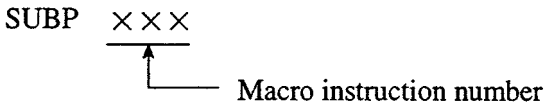
(c) Explanation

The JMP and ADR instructions are always used in pairs. The label numbers specified by the JMP and ADR instructions given in a pair must be the same.

6.3.5 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.

Marco instructions are written in the following format.

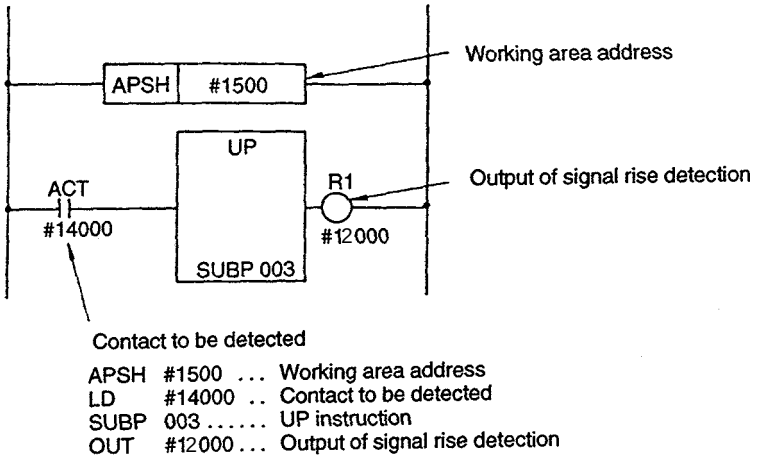


(1) SUBP 003 (UP: Detecting rising edge of a signal) RR after operation: ⇳

(a) Function

The instruction detects the rising edge of a signal.

(b) Format



(c) Control conditions

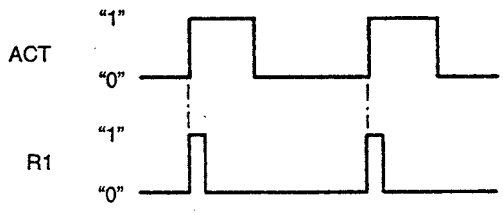
- Working area address (APSH#XXXX)

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

- 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..



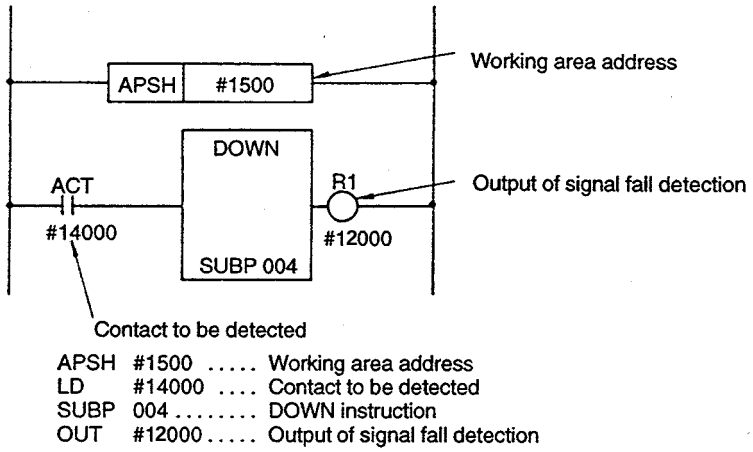
- If “ACT = 1” when the power is turned ON, it is regarded as the rise.

(2) SUBP 004 (DOWN: Detecting falling edge of a signal) RR after operation: ⇳

(a) Function

The instruction detects the falling edge of a signal.

(b) Example



(c) Control conditions

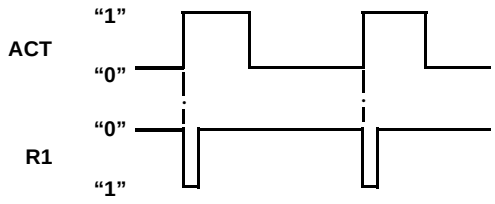
- Working area address (APSH#XXXX)

  Designate an address that is not used by other instructions. One byte is necessary for one SUBP 004.

- Contact to be detected (ACT) and output of signal fall detection (R1)

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

  ACT = 1: R1 value changes “0” → “1” → “0” at the detection of the falling edge.



- Even if “ACT = 0” when the power is turned ON, it is not regarded as the falling edge.

(3) SUBP 005 (COUNTER) RR after operation: ⇳

(a) Function

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

- 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 to the preset value.

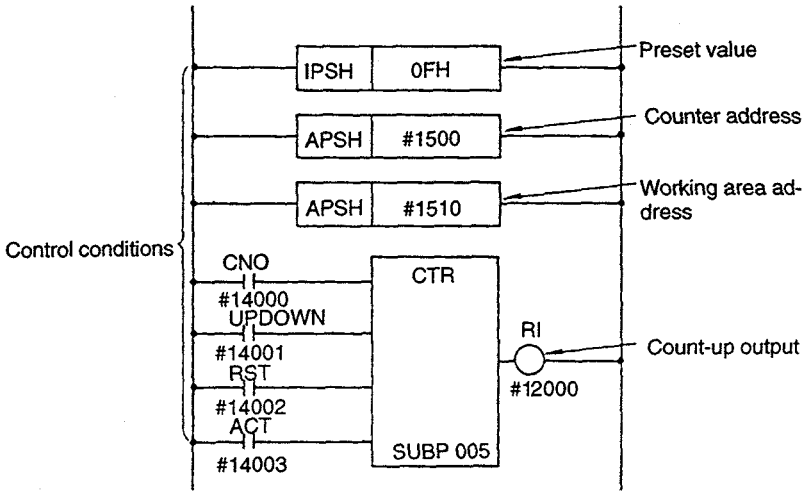
- Preset counter

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

- Up/down counter

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

(b) Example




```
IPSH   0FH ..... Preset value
APSH   #1500 ... Counter address
APSH   #1510 ... Working area address
LD     #14000 .. Designation of initial value
STR    #14001 .. Designation of up/down counter
STR    #14002 .. Reset
STR    #14003 .. Count signal
SUBP   005 ..... Counter instruction
OUT    #12000 ... Count-up output
```

(c) Control conditions

- Designation of preset value (IPSHXX)

  Designate the preset value directly.

  To designate the 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.

- Designation of counter address (APSH#XXXX)

  Designate the counter address

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

- Designation of working area address (APSH#XXXX)

  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.

- Designation of initial value (CNO)

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

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

- 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.

- Reset (RST)

  RST = 0: Reset released

  RST = 1: Reset

  R1 is cleared to “0”.

  Counted value is reset to the initial value.

- Count signal (ACT)

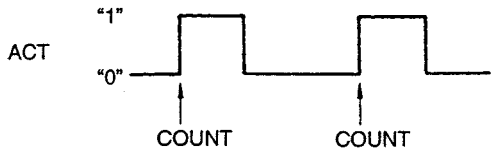
  ACT = 0: The counter does not operate. Contents of R1 remain unchanged.

  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 t the initial value at the first ACT signal.

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



- 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”.

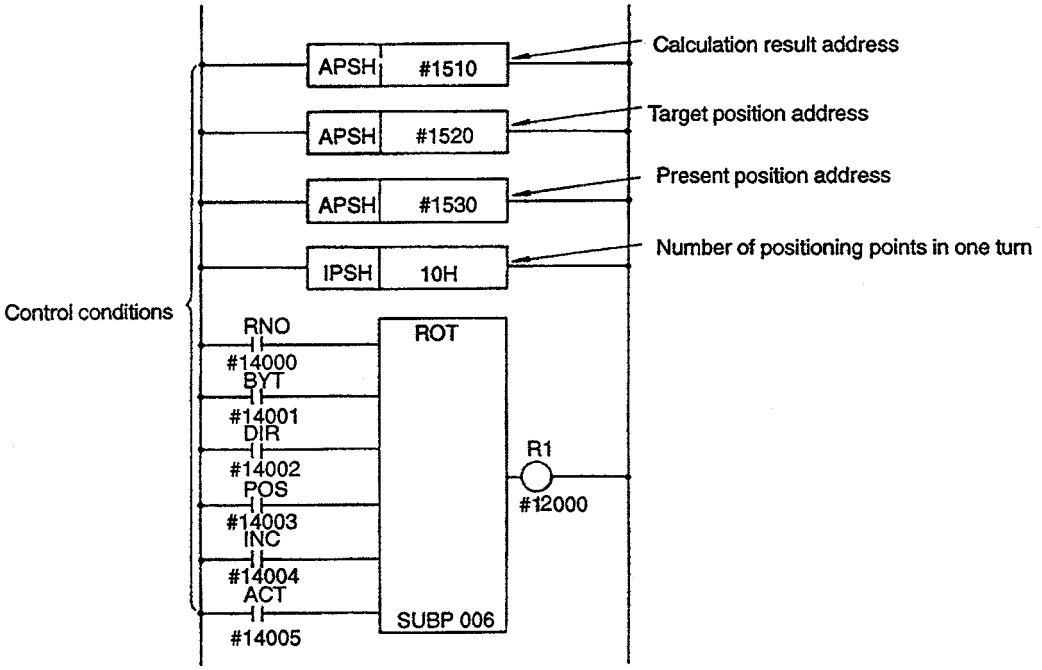
(4) SUBP 006 (ROTATION) RR after operation: ⇳

(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) Example




```
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   #12000 .. Output of direction of rotation
```

(c) Control conditions

- Designation of calculation result storing address (APSH#XXXX)

  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.

- Designation of target position address (APSH#XXXX)
- Designate the address where the target position is stored: for example, the address where the T command is output from the NC.
- Designation of preset position address (APSHXXXX)
- 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.

- 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”.

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

  BYT = 0: Rotation body position number is 2-digit BCD data (1 byte).

  BYT = 1: Rotation body position number is 4-digit BCD data (2 byte).

  DIR = 0: The short cut rotation direction is not made. (only forward rotation is performed.)

  DIR = 1: The short cut rotation direction is made. (one shortest path direction is output to R1.)

  POS = 0: The number of steps to the target position is calculated.

  POS = 1: The number of steps that is required to reach the step immediately before the target position is calculated.

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

  INC = 0: The target position number is calculated.

  INC = 1: The number of steps required to reach a target position is calculated.

- Execution command (ACT)

  ACT = 0 ROT instruction is not executed.

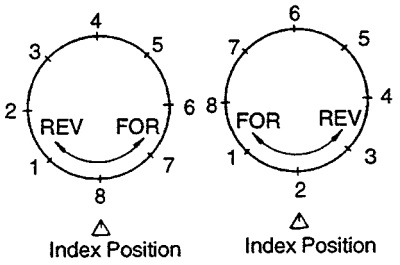
  ACT = 1: ROT instruction is executed.

- 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.

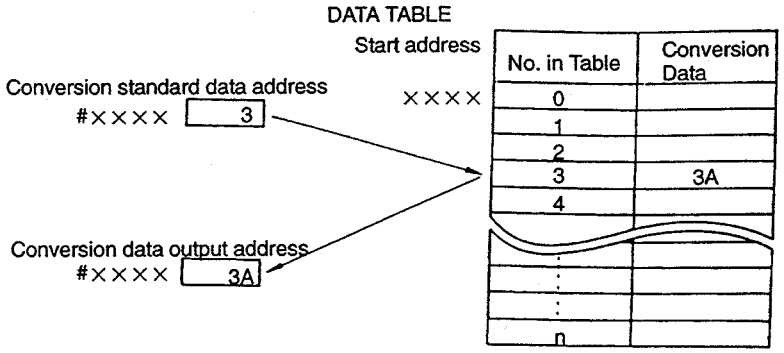


If the number of steps to the position one step before the target position is calculated while the present position is equal to the target position (POS = 1, INC = 1), the result of calculation is “0”.

(5) SUBP 007 (CODE CONVERT) RR after operation: ⇳

(a) Function

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



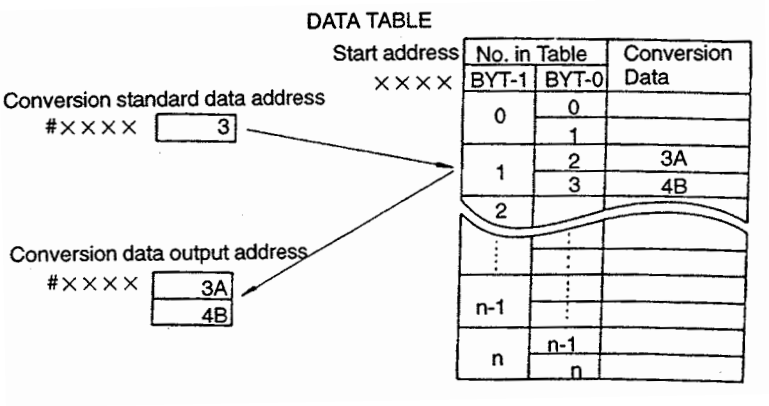
- BYT = 0

  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.

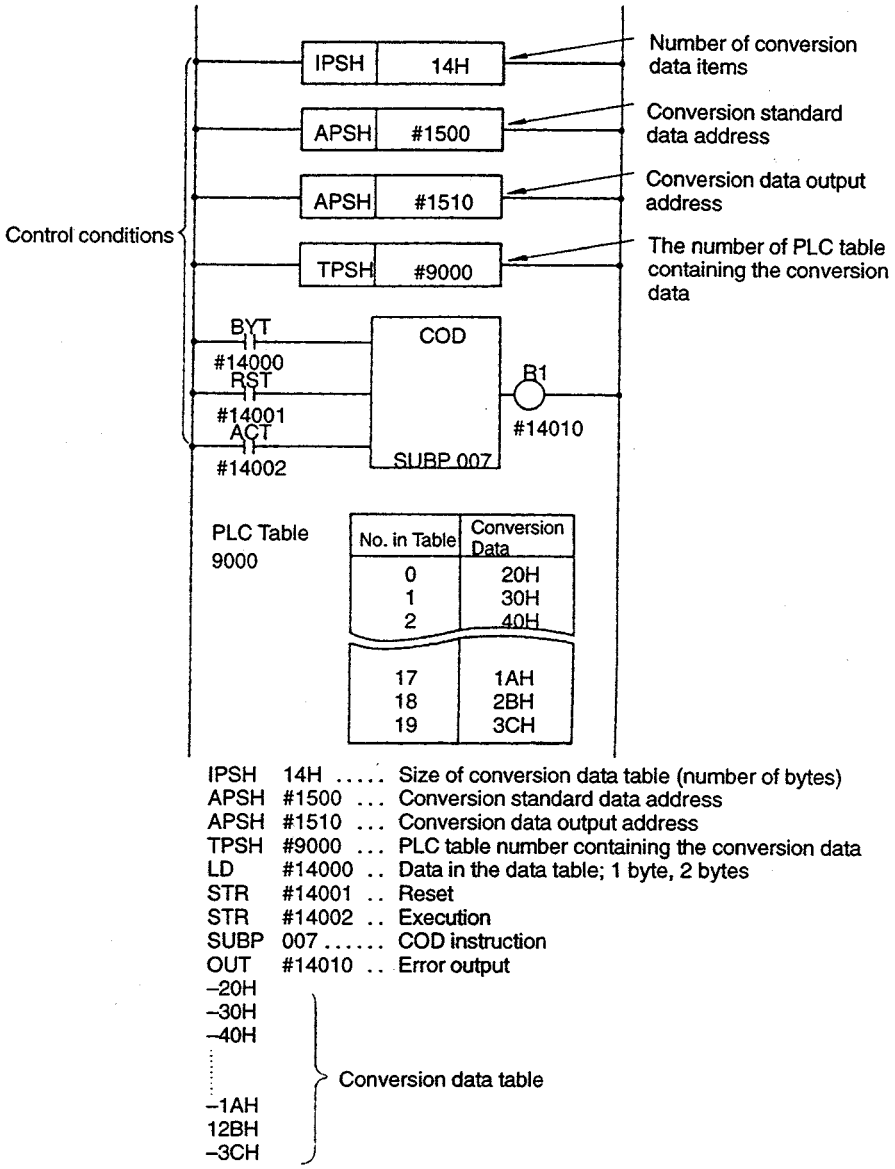
  The start of the table is 0th.

- BYT = 1

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



(b) Example



(c) Control conditions

- Designation of the number of conversion data items (IPSHXX)
- Designate the size of the conversion data table by the number of bytes. The maximum size is 256 bytes.
- Designation of the conversion standard data address (APSH#XXXX). The data in the conversion data table can be read out by the designating the number in the table.
- Designate the number in the table.
- Designation of the conversion data output address (APSH#XXXX)

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.

- Designation of the conversion data table (TPSHXXXX)

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

  9000 to 9007: Max 256 bytes

  9008 to 9023: Max. 128 bytes

- 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

- Reset (RST)

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

  RST = 0: Not reset

  RST = 1: Reset

- Execution command (ACT)

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

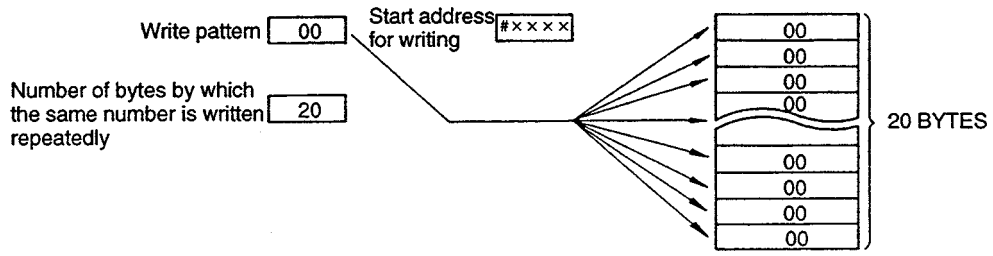
  ACT = 1: The COD instruction is executed.

- Error output (R1)

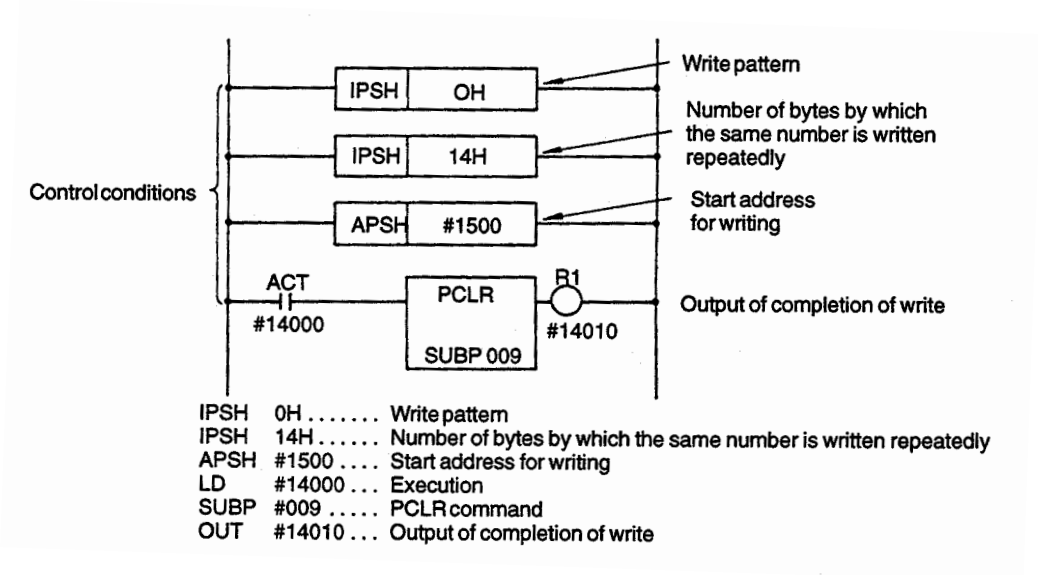
  If an error occurs during the execution of the COD instruction (a numeric value greater that the size of the table is set), “1” is set for “R1” (R1” (R1 = 1) indicating the occurrence of an error.

(6) SUBP 009 (PATTERN CLEAR) RR after operation: ⇳

(a) Function



(b) Example



(c) Control conditions

- Designation of the write pattern (IPSHXX)

  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.

- Designation of the number of bytes by which the same number is written repeatedly (IPSHXX)

  Designate the number of bytes to clear the pattern.

- Designation of the start address of writing (APSH#XXXX)

  Designate the start address of writing.

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

- Execution command (ACT)

  ACT = 0: The PCLR instruction is executed.

  ACT = 1: The PCLR instruction is not executed.

- Out of completion of write (R1)

  R1 = 0: Writing not completed.

  R1 = 1: Writing completed.

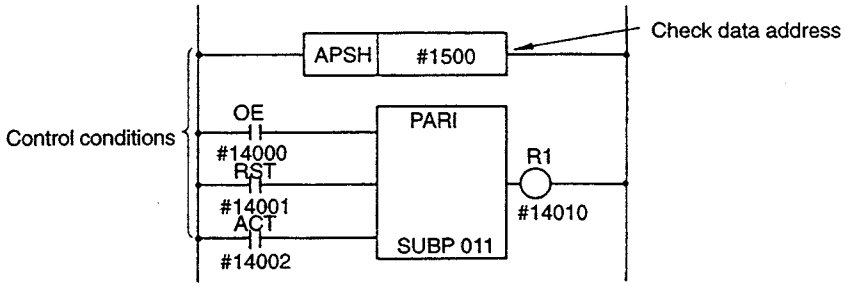
(7) SUBP 011 (PARITY CHECK) RR after operation: ⇳

(a) Function

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

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

(b) Example




```
APSH  #1500 ... Check data address
LD    #14000 .. Designation of parity scheme (even/odd)
STR   #14001 .. Reset
STR   #14002 .. Execution command
SUBP  011 ..... PARI instruction
OUT   #14010 .. Error output
```

(c) Control conditions

- Designation of check data address (APSHXXXX)

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

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

- Designation of parity scheme (OE)

  OE = 0: Even parity check

  OE = 1: Odd parity check

- Reset (RST)

  RST = 0: Error output R1 is not reset.

  RST = 1: Error output R1 is reset.

- Execution command (ACT)

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

  ACT = 1: The PARI instruction is executed.

- Error output (R1)

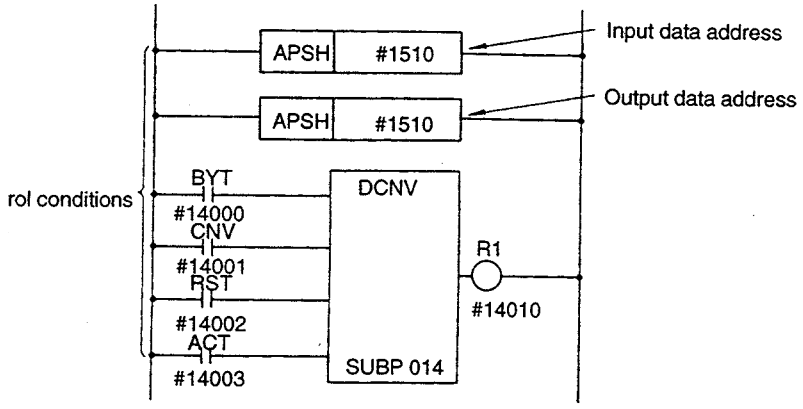
  If the result of parity check does not meet the designated parity scheme, "1" is set for "R1" (R1 = 1).

(8) SUBP 014 (DATA CONVERT) RR after operation: ⇳

(a) Function

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

(b) Example




```
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

- Designation of the conversion data address (APSHXXXX)

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

  If “BYT = 1”, continuous two bytes are used.

- Designation of the converted data address

  Designate the address where the result of conversion is stored.

  If “BYT = 1”, continuous two bytes are used.

- 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.

- Designation of the conversion type (CNV)

  CNV = 0: Conversion of binary data to BCD data.

  CNV = 1: Conversion of BCD data to binary data.

- Reset (RST)

  RST = 0: Error output R1 is not reset.

  RST = 1: Error output R1 is reset.

- Execution command (ACT)

  ACT = 0: The DCNV instruction is not executed.

  ACT = 1: The DCNV instruction is executed.

- Error output (R1)

  R1 = 0: Normal

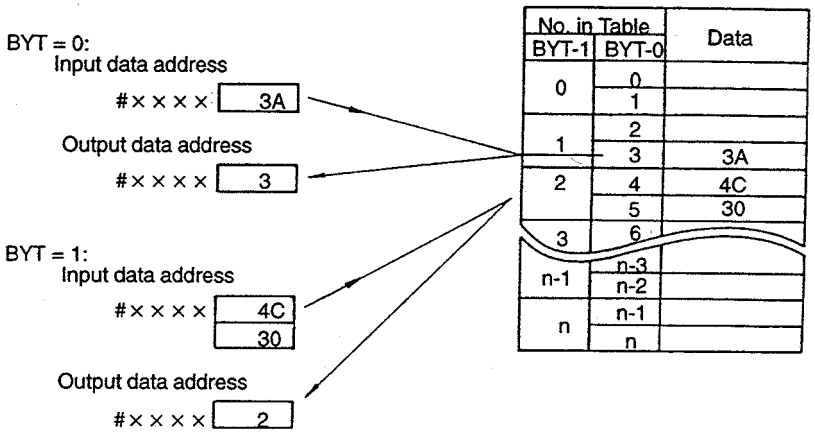
  R1 = 1: Error

  (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) RR after operation: ⇳

(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.

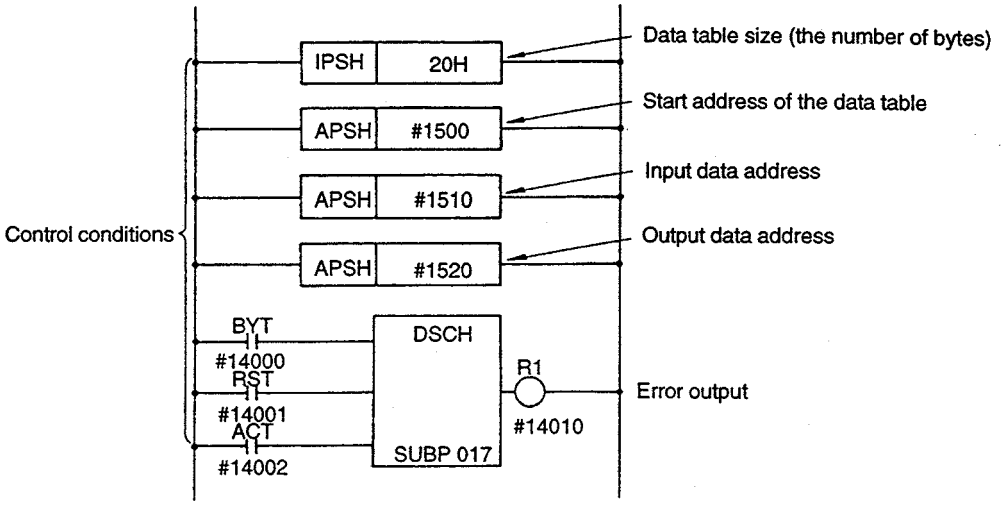


Note 1: When "BYT = 1", the size of the table must be an even number of bytes.

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) Example




```
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 ..... DSCH instruction
OUT    #14010 .. Error output
```

(c) Control conditions

- Designation of the data table size (the number of bytes) (IPSHXXXX)

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

- Designation of the start address of the data table (APSH#XXXX)

  Designate the start address of the data table.

  The data table can be created at any place.

- Designation of the input data address (APSH#XXXX)

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

- Designation of the output data address (APSH#XXXX)

  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.

- 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.

- Execution command (ACT)

  ACT = 0: The DSCH instruction is executed.

  ACT = 1: The DSCH instruction is not executed.

- Reset (RST)

  RST = 0: Error output R1 is not reset.

  RST = 1: Error output R1 is reset.

- Error output (R1)

  R1 = 0: The search data are found.

  R1 = 1: The search data are not found.

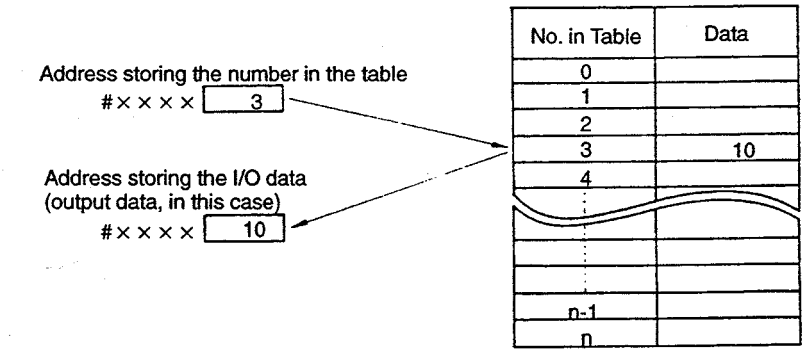
(10)SUBP 018 (INDEX DATA MOVE) RR after operation: ⇳

(a) Function

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

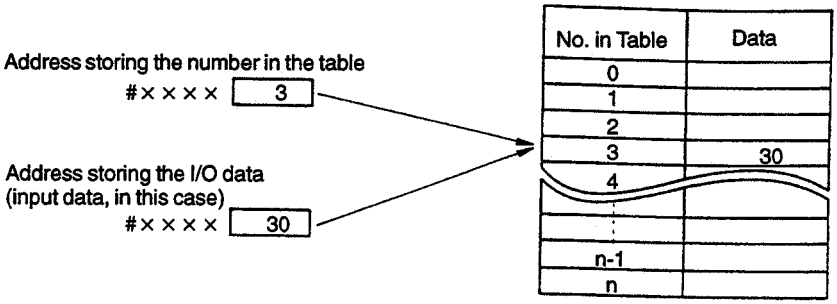
- Reading

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

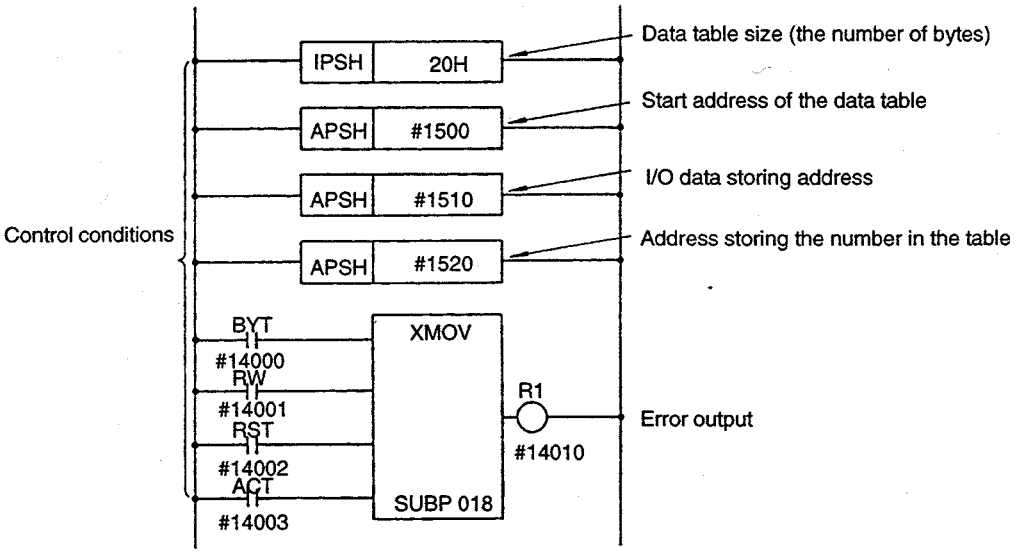


- Rewriting

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



(b) Example




```
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 instruction
OUT   #14010 .. Error output
```

(c) Control conditions

- Designation of the data table size (number of bytes) (IPSHXX)

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

- Designation of the start address of the data table (APSH#XXXX)

  Designate the start address of the data table.

  The data table can be created any place.

- Designation of the address storing the I/O data (APSH#XXXX)

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

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

- Designation of the address storing the number in the table (APSH#XXXX)

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

- 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.

- Designation of read/write processing (RW)

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

RW = 1: Data in the data table are rewritten

- Reset (RST)

  RST = 0: Error output R1 is not reset.

  RST = 1: Error output R1 is reset.

- Execution command (ACT)

  ACT = 0: The XMOV instruction is executed.

  ACT = 1: The XMOV instruction is not executed.

- Error output (R1)

  R1 = 0: Normal

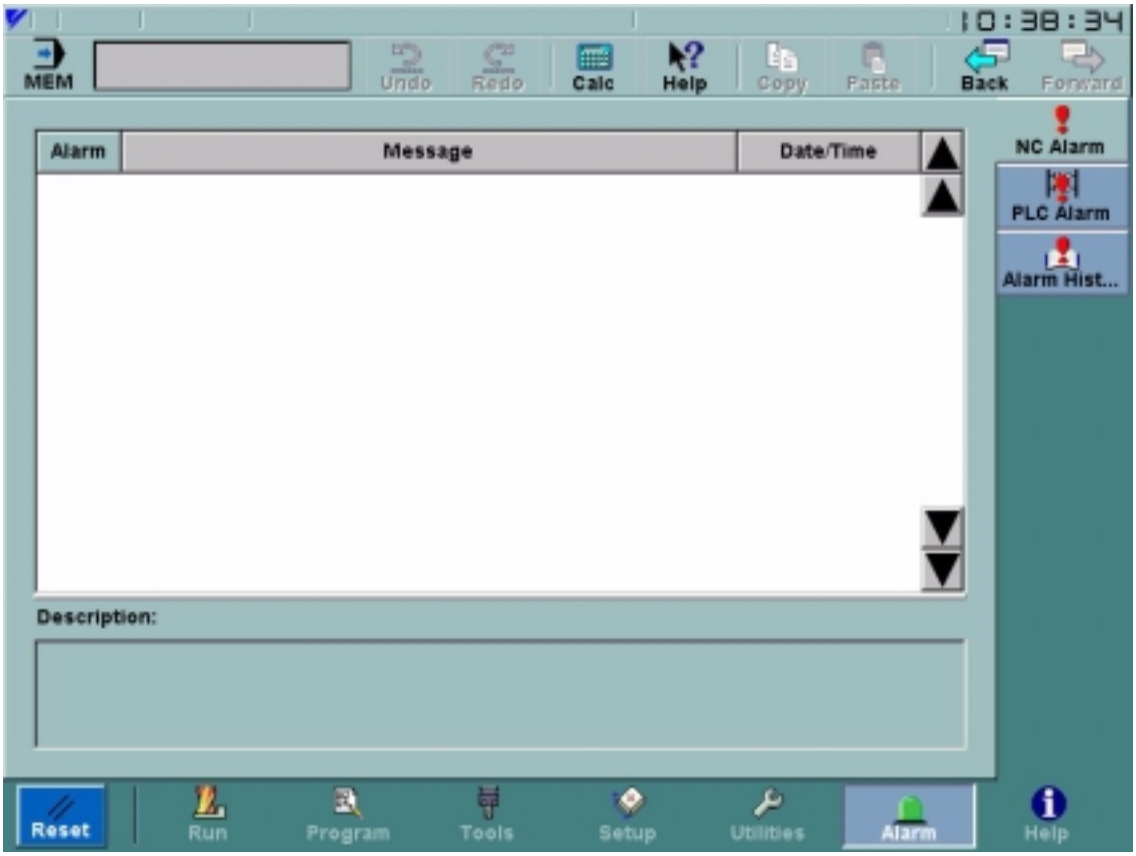
  R1 = 1: Error

  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) RR after operation: ⇳

(a) Function

The instruction displays the message on the screen.



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

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

- 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).
- 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.

- 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 designing a JMP or other appropriate instructions.

(b) Example

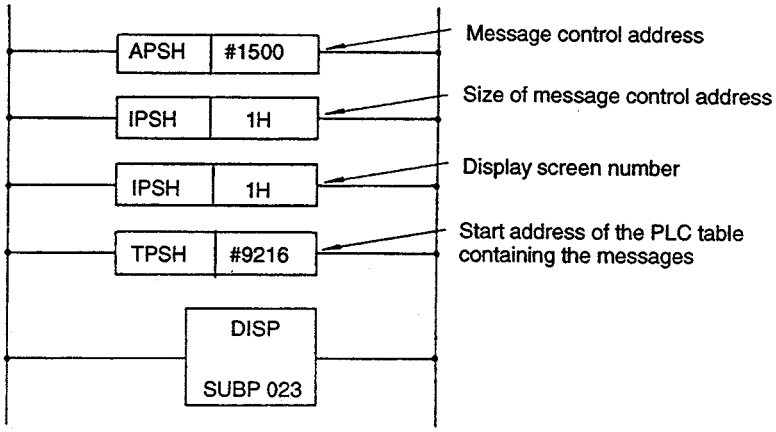


Table Address	Display Request	Message
#9216	#15000	SPINDLE ALARM
#9217	#15001	M06 ERROR
#9218	#15002	TAPPING ERROR
#9219	#15003	
#9229	#15015	UNUSABLE S-CODE
#9230	#15016	UNUSABLE M-CODE
#9231	#15017	PARAMETER ERROR

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

- Designation of the message control address (APSH#XXXX)

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

- Designation of the size of message control address (IPSHXX).

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

  Example: APSH#1500

  IPSH 1H

  With the designation indicated above, continuous 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 message is used.

- Designation of the display screen number (IPSHX)

  Designate the display page number to be used.

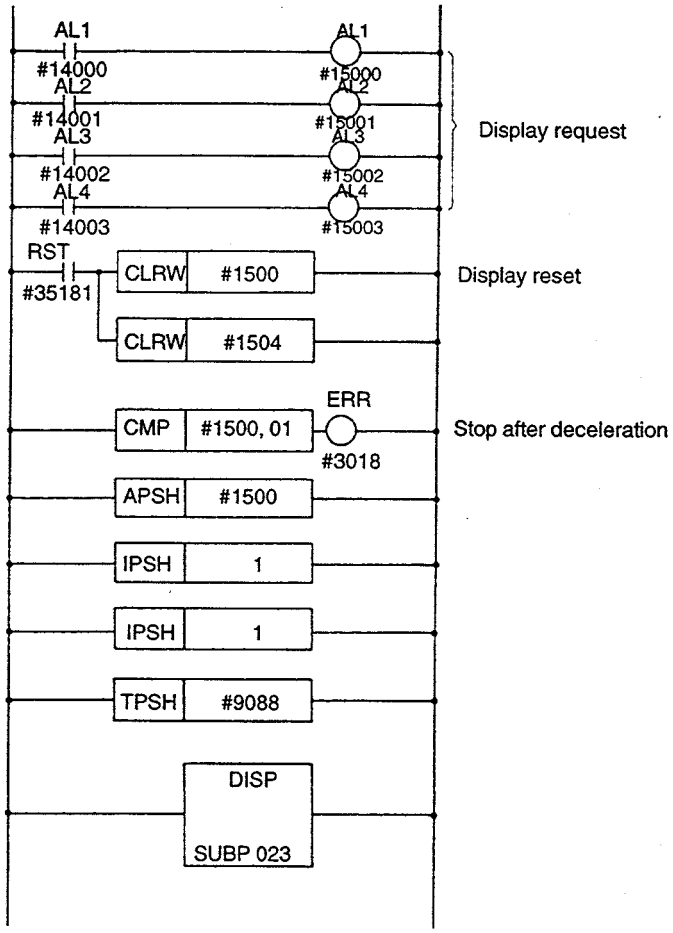
  A total of two pages (No. 1 and No. 2) can be used for both the high-speed processing and low-speed processing sequence programs.

- Designation of the start address of the PLC table containing the message (TPSHXXXX)

(d) Example 2

When the contact of AL 1 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.

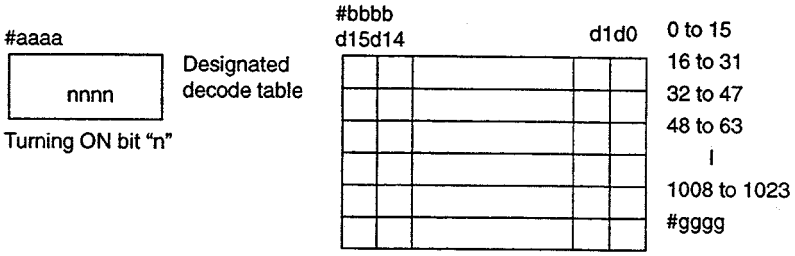
For the display of user's messages, two pages exist. To change the display page between No.1 and No. 2 pages, use the page keys.

(12)SUBP 025 (Binary Decoding) RR after operation: ⇳

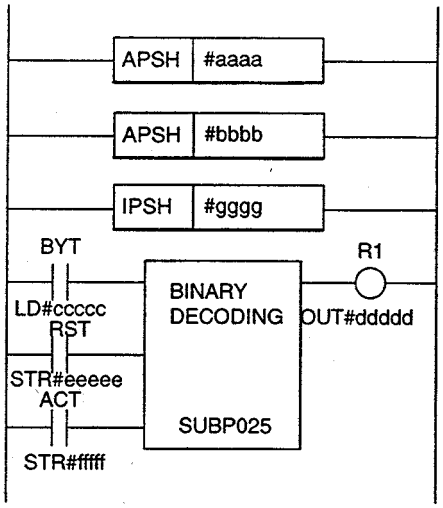
(a) Function

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) Example



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
STR #eeee.e ..... Reset
STR #ffff.f ..... Execution
SUBP 025 ..... BINARY DECODING instruction
OUT #dddd.d ..... Error output

(c) Control conditions

- Designation of the data code area (APSH#XXXX)

  Designate the address of the code to be decoded.

  Two bytes are used.

- Designation of the start address of the decode table (APSHXXXX)

  Designate the start address of the table to be decoded.

- Maximum decode number (IPSHXXXX)

  Designate the maximum number of the decode bits.

- Designation of the size of the data in the decode area (BYT).

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

  LD = 0: The size if the data in the conversion table is 2 byte.

- Reset (RST)

  RST =0: Error output R1 is not reset.

  RST = 1: Error output R1 is reset.

- Execution command (ACT)

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

  ACT = 1: The binary decode instruction is to executed.

- Error output (R1)

  R1 = 0: Normal

  R1 = 1: Error

  A numeric value greater than the table size is set.

(13)SUBP 027 (Binary Code Conversion) RR after operation: ⇳

(a) Function

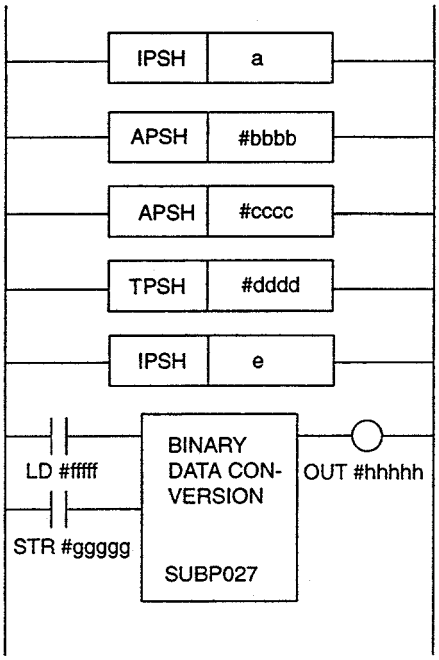
The instruction converts the data by using the conversion table created on the PLC table.

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 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 . . . n	2 . . . n	4 . . . n	

(b) Example



IPSH a ..... Size of the conversion table (number of bytes)
APSH #bbbb ..... Conversion standard data address
APSH #cccc ..... Conversion data output address
TPSH #dddd ..... Table number of the conversion data
IPSH e ..... Designation of the size of the data
LD #ffff.f ..... Reset
STR#gggg.g ..... Execution
SUBP 027 ..... BINARY CODE CONVERSION instruction
OUT #hhhh.h ... Error output

(c) Control conditions

- Designation of the size of the conversion data table (IPSHX)

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

  The maximum size is 256 bytes.

- Designation of the conversion standard data table address (APSH#XXXX)

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

  Designate this number in the table.

- Designation of the conversion data output address (APSH#XXXX)

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

- Designation of the number of conversion table (TPSH#XXXX)

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

  9000 to 9007: Max. 256 bytes

  9008 to 9023: Max. 128 bytes

- Designation of the size of the data in conversion table (IPSHX)

  IPSH e = 1: 1 byte

  IPSH e = 2: 2 byte

  IPSH e = 3: 4 byte

- Reset (RST)

  RST = 0: Error output R1 is not reset.

  RST = 1: Error output R1 is reset.

- Execution command (ACT)

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

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

- Error output (R1)

  R1 = 0: Normal

  R1 = 1: Error

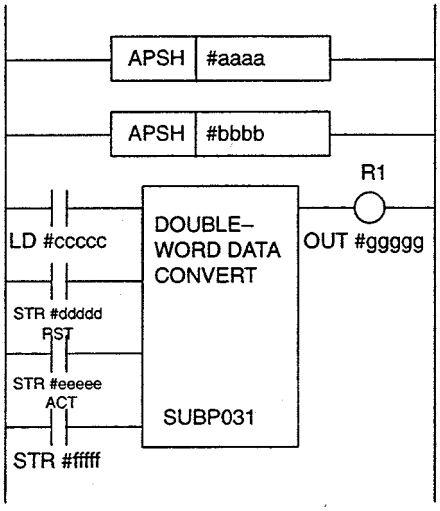
  A numeric value greater than the table size is set.

(14)SUBP 031 (Double-word Data Convert) RR after operation: ⇳

(a) Function

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

(b) Example



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

(c) Control conditions

- Designation of the conversion data address (APSH#XXXX)

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

  Both the binary and BCD data use 4 bytes.

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

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

- Designation of the converted data address (APSH#XXXX)

  Designate the address where the data result of conversion is stored.

  Both the binary and BCD data use 4 bytes.

- Designation of the number of bytes (LD#XXXX.X)

  LD = 0: No conversion

- Designation of the conversion type (STR#XXXX.X)

  STR = 0: Conversion of binary data to BCD data.

  STR = 1: Conversion of BCD data to binary data

- Reset (RST)

  RST = 0: Error output R1 is not reset.

  RST = 1: Error output R1 is reset.

- Execution command (ACT)

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

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

- Error output (R1)

  R1 = 0: Normal

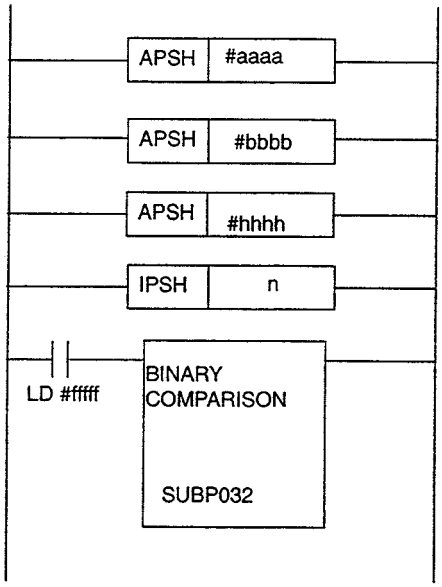
  R1 = 1: Error

(15)SUBP 032 (Binary Comparison) RR after operation: ⇳

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) Example



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

(c) Control conditions

- Designation of the input data address (APSH#XXXX)

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

- Designation of the comparison data address (APSH#XXXX)

  Designate the address where the comparison data is stored.

- Designation of the size of the data (LD#XXXX.XX)

  LD = 0: 1 byte

  LD = 1: 2 bytes

  LD = 2: 4 bytes

- Execution command (ACT)

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

  ACT = 1: The binary data comparison instruction is executed.

- Result of comparison (APSH#XXXX)

  d0 = 1 aaaa = bbbb

  d1 = 1 aaaa > bbbb

  d2 = 1 aaaa < bbbb

(16)SUBP 034 (Binary Data Search) RR after operation: ⇳

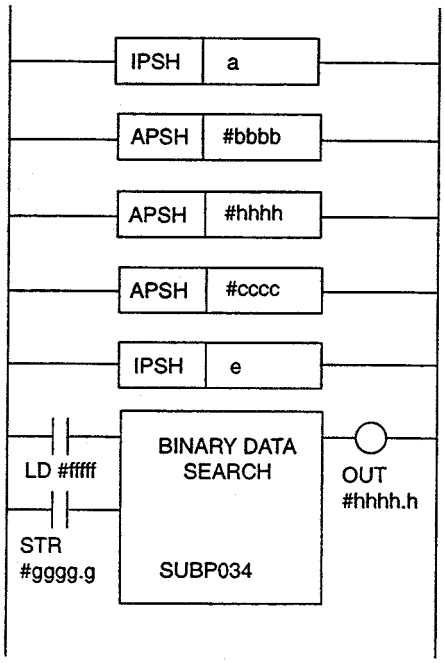
a) Function

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 . . . n	2 . . . n	4 . . . n	

(b) Example



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 ..... Designation of the data size
LD #ffff.f ..... Reset
STR#gggg.g ..... Execution
SUBP 034 ..... Binary data search instruction
OUT #hhhh.h ... Error output

(c) Control conditions

- Designation of the size of the data table (number of bytes) (IPSH X)

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

- Designation of the start address of the data table (APSH#XXXX)

  Designate the start address of the data table

  The data table can be created at any place.

- Designation of the input data address (APSH#XXXX)

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

- Designation of the output data address (APSH#XXXX)

  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.

- Designation of the data size (IPSH X)

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

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

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

- Reset (RST)

  RST = 0: Error output R1 is not reset.

  RST = 1: Error output R1 is reset.

- Execution command (ACT)

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

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

- Error output (R1)

  R1 = 0: The search data are found.

  R1 = 1: The search data are not found.

(17)SUBP 035 (Binary Index Modifier Data Transfer) RR after operation: ⇳

a) Function

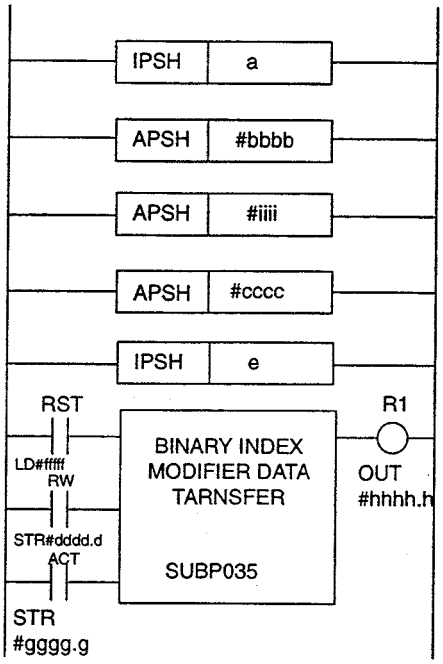
The instruction reads the data table or rewrites the data in the data table.

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 . . . n	2 . . . n	4 . . . n	

(b) Example



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 ..... Designation of the data size
LD #ffff.f ..... Reset
STR#dddd.d ... Read or rewrite
STR#gggg.g ... Execution
SUBP 035 ..... Binary index modifier data tarnsfer instruction
OUT #hhhh.h . Error output

(c) Control conditions

- Designation of the size of the data table (number of bytes) (IPSH X)

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

- Designation of the start address of the data table (APSH#XXXX)

  Designate the start address of the data table

  The data table can be created at any place.

- Designation of the address storing the I/O data (APSH#XXXX)

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

- Designation of the address storing the number in the table (APSH#XXXX)

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

- Designation of the data size (IPSH X)

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

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

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

- Designation of read/write processing (RW)

  STR = 0: Data is read from the data table.

  STR = 1: Data in the data table is rewritten.

- Reset (RST)

  RST = 0: Error output R1 is not reset.

  RST = 1: Error output R1 is reset.

- 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.

- 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 is not a multiple of the designated data size.

  Example: Data size.

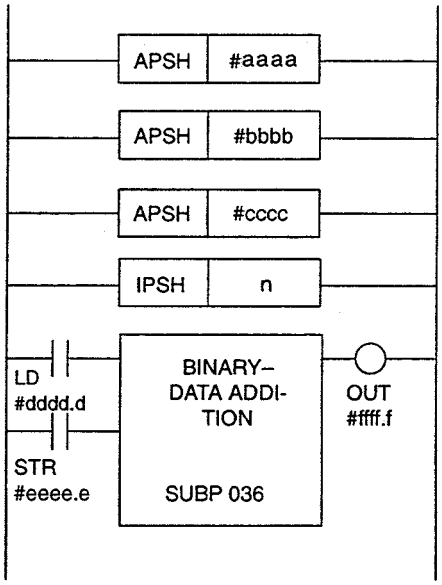
Byte	1 × Nbytes
Word	2 × Nbytes
Double-word	4 × N bytes

(18)SUBP 036 (Binary-data Addition) RR after operation: ⇳

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) Example



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 ..... Binary data addition instruction
OUT #ffff.f ..... Error output

(c) Control conditions

- Designation of the augend data address (APSH#XXXX)

  Designate the address where the augend data are stored.

- Designation of the addend data address (APSH/IPSH#XXXX)

  Designate the address where the addend data is stored.

- Designation of the operation result output address (IPSH X)

  Designate the address where the result of operation is output.

  The address is stored as 4-byte data.

- Designation of the type of operation.

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

  1st digit = 0: 1byte

= 1: 2 bytes

= 2: 4bytes

2nd digit = 0: Constant data

= 1: Address data

- 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.

- Error output (OUT)

  OUT = 0: Normal

  OUT = 1: Error

  #2999

  Operation status is written

  d0 = 1: 0

  d1 = 1: Negative

  d5 = 1: Overflow

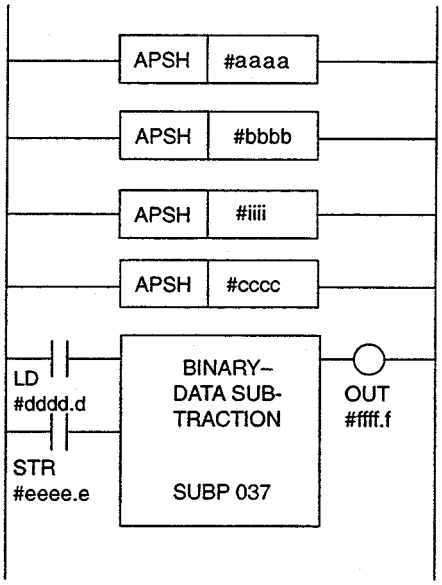
(19)SUBP 037 (Binary-data Subtraction) RR after operation: ⇳

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) Example



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 ..... Binarydata subtraction instruction
OUT #ffff.f ..... Error output

(c) Control conditions

- Designation of the minuend data address (APSH#XXXX)

  Designate the address where the minuend data is stored.
- Designation of the subtrahend data address (APSH/IPSH#XXXX)

  Designate the address where the subtrahend data is stored.

  It is also possible to subtract the designated data.
- Designation of the operation result output address (APSH#XXXX)

  Designate the address where the result of operation is output.

  The address is stored as 4-byte data.
- Designation of the type of operation (IPSH X)

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

  1st digit = 0: 1byte

  = 1: 2 bytes

  = 2: 4bytes

  2nd digit = 0: Constant data

  = 1: Address data

- 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.

- Error output (OUT)

  OUT = 0: Normal

  OUT = 1: Error

- #2999

  Operation status is written.

  d0 = 1: 0

  d1 = 1: Negative

  d5 = 1: Overflow

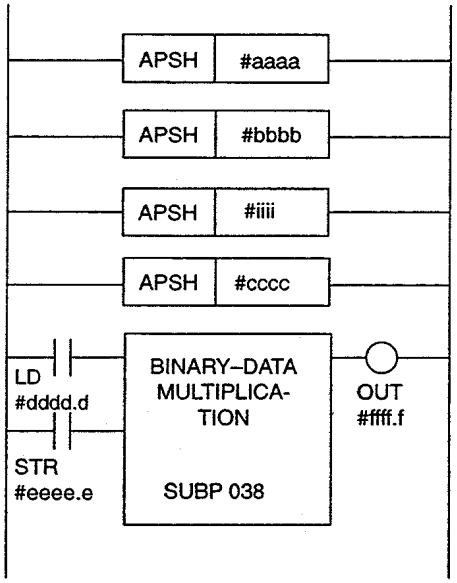
(20)SUBP 038 (Binary-data Multiplication) RR after operation: ⇳

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) Example



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 ..... Binary–data multiplication insttruction
OUT #ffff.f ..... Error output

(c) Control conditions

- Designation of the multiplicand data address (APSH#XXXX)

  Designate the address where the multiplicand data is stored.

- Designation of the multiplier data address (APSH/IPSHD#XXXX)

  Designate the address where the multiplier data is stored.

- Designation of the operation result output address (APSH#XXXX)

  Designate the address where the result of operation is output.

  The address is stored as 4-byte data.

- Designation of the type of operation (IPSH X).

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

  1st digit = 0: 1byte

  = 1: 2 bytes

  = 2: 4bytes

  2nd digit = 0: Constant data

  = 1: Address data

- 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.

- Error output (OUT)

  OUT = 0: Normal

  OUT = 1: Error

- #2999

  Operation status is written.

  d0 = 1: 0

  d1 = 1: Negative

  d5 = 1: Overflow

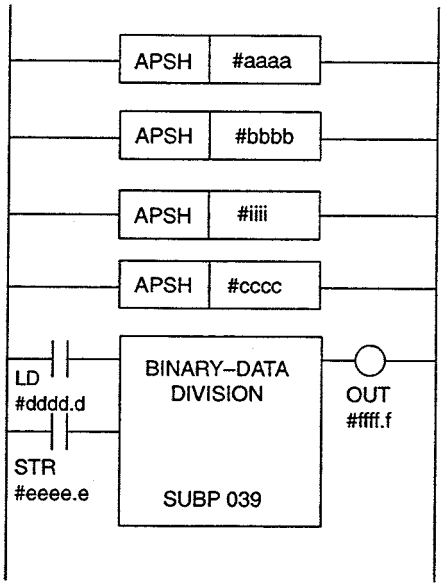
(21)SUBP 039 (Binary-data Division) RR after operation: ⇳

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) Example



APSH #aaaa ..... Dividend data address
APSH (IPSHD) #bbbb ... Divisor data address
APSH #cccc ..... Operation result output address
IPSH n ..... Operation type
LD #dddd.d ..... Reset
STR#eeee.e ..... Execution
SUBP 039 ..... Binary–data division instruction
OUT #ffff.f ..... Error output

(c) Control conditions

- Designation of the dividend data address (APSH#XXXX)

  Designate the address where the dividend data is stored.

- Designation of the divisor data address (APSH/IPSHD#XXXX)

  Designate the address where the divisor data is stored.

- Designation of the operation result output address (APSH#XXXX)

  Designate the address where the result of operation is output.

  The address is stored as 4-byte data.

- Designation of the type of operation (IPSH X).

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

  1st digit = 0: 1byte

  = 1: 2 bytes

  = 2: 4bytes

  2nd digit = 0: Constant data

  = 1: Address data

- 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.

- Error output (OUT#XXXX.X)

  OUT = 0: Normal

  OUT = 1: Error

- #2999

  Operation status is written.

  d0 = 1: 0

  d1 = 1: Negative

  d5 = 1: Overflow

  Division by “0” causes overflow.

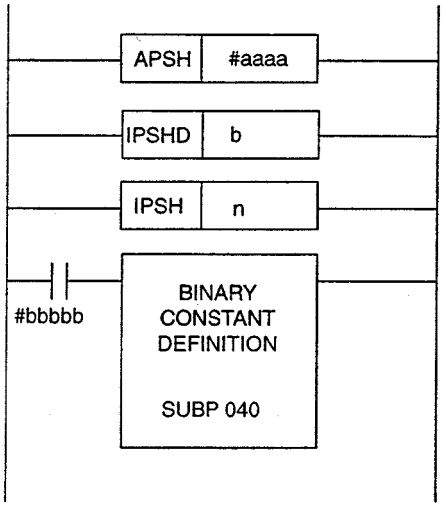
(22)SUBP 040 (Binary Constant Definition) RR after operation: ⇳

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) Example



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

(c) Control conditions

- Designation of the constant output address (APSH#XXXX)

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

- Designation of setting data (IPSH#XXXX)

  Set the constant in decimal.

- Data length must be within the specified byte length. The range of setting data is ±999999999.

- Designation of byte length (IPSH X)

  IPSH = 0: 1byte

  IPSH = 1: 2 bytes

  IPSH = 2: 4bytes

- 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.

6.3.6 Auxiliary Instruction of Macro Instructions

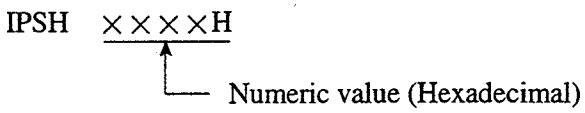
The auxiliary instructions of macro instructions are described below.

(1) IPSH (Immediate Plus) RR after operation: RR-

(a) Function

Directly designate the numeric value which is used by SUBP.

(b) Format

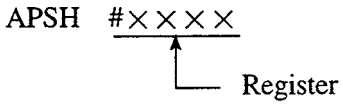


(2) APSH (Address Pus) RR after operation: RR-

(a) Function

Designate the address of the register to be used by SUBP.

(b) Format

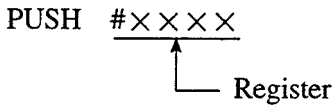


(3) PUSH (Push) RR after operation: RR-

(a) Function

Designate the address where the numeric value to be used by SUBP is stored.

(b) Format

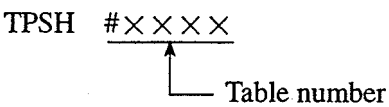


(4) TPSH (Table Push) RR after operation: RR-

(a) Function

Designate the table number of the PLC table which is used by SUBP.

(b) Format

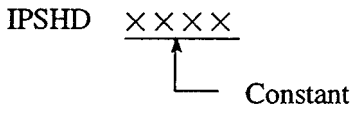


(5) IPSHD RR after operation: RR-

(a) Function

Directly designate the numeric value which is used by SUBP 036 to SUBP 040.

(b) Format



7

Offline Editing

Chapter 7 describes offline editing

7.1 Outline of Offline System

The off-line 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/AT or compatibles

OS: MS-DOS Ver.6.2 or above.

Memory: 400K bytes minimum

7.1.2 Execution Files

The YASNAC PCNC off-line systems consists of the following software packages.
- J1LCOMP.EXE Ladder language compiler
- J1LLINK.EXE Linker

7.1.3 Outline of the Execution Files

(1) Ladder Language Complier

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 number
- High-speed scan ladder program
- Low-speed scan ladder program
- Low speed ladder stop count
- Conversion data table
- Message data table
- Symbol data

PCNC treats the low-speed ladder stop count as no-meaning code.

(2) Linker

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

7.1.4 Sequence Program Development Procedure

The following chart shows the YASNAC PCNC procedure for developing the sequence 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 7.2 “SOURCE FILE”.

YELADDER.SRC

```
            ;************************
            **********;
VERSION JXSD LADDER

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

LOW SEQUENCE; 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
```

② Compile the created or modified source file.

Generate the object file by using the J1LCOMP.

For the operation procedure, refer to 7.3 “COMPILER”.

③ By consolidating the object files into a single file, generate the executable file.
Use the J1LLINK to generate the executable file.

For the operation procedure, refer to 7.4 “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.

7.2 Source File

The format of 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.

The following shows the available, pseudo instructions.

- 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 sequence1] [Low-speed scan sequence 2]

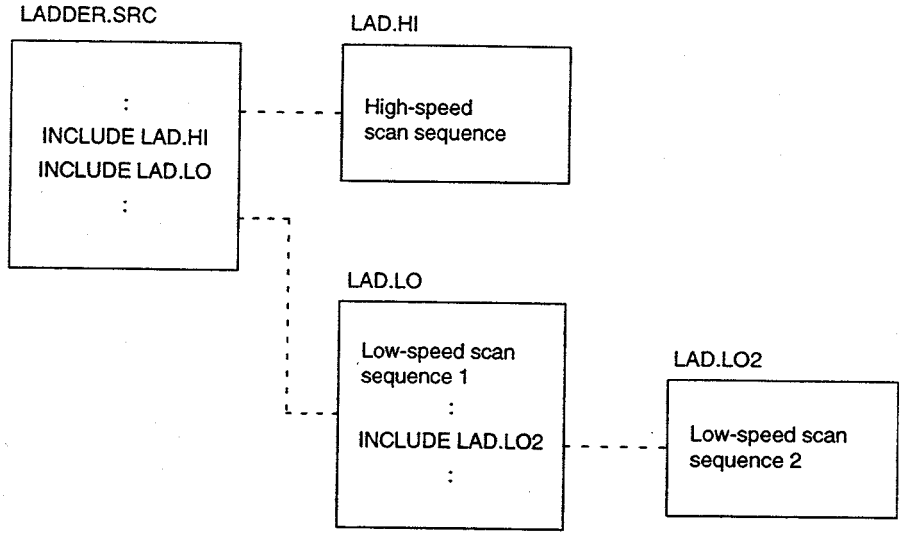


Fig. 7.1Nesting in the Source File

- 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)

```
                ;*************************
                **********;
①   VERSION JXSD LADDER
②
③   LOWSTOPCOUNT2; Low-speed scan ladder pro-
④   gram stop count
⑤   HIGHSEQUENCE; High-speed scan ladder program
⑥   INCLUDE  LAD.HI
    ENDP
    LOW SEQUENCE; Low-speed scan ladder program
    INCLUDE  LAD.LO1
    INCLUDE  LAD.LO2
⑦   INCLUDE  LAD.LO3
    ENDP

⑧   CONVERSION; Conversion data
    INCLUDE CONV.DAT
    ENDP
⑨
    MESSAGE;Message data
```

Fig. 7.2 Main File

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.

   Format: VERSION AAAAAAAAAAAAAAAAAAAA

   A version number should be set in up to 20 characters.

   If no entry is made, spaces occupy 20 columns.

   ②HIGHSEQUENCE

   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.

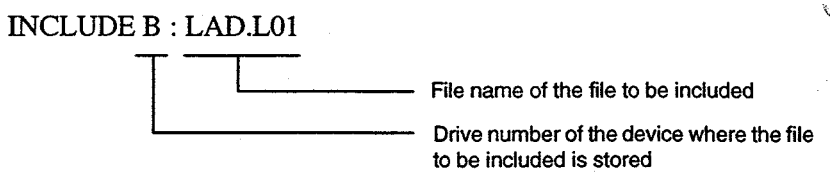
Format: HIGHSEQUENCE . . . . . ENDP

If no entry is made, a high speed scan ladder is not generated.
This pseudo instruction must always be written in a main file.

③ INCLUDE

This instruction calls up the files to be included.

Format:



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

INCLUDE B:\LPROG\LOW\LAD.L01

④ 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.

⑤ 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.

⑥ 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.

⑦ 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.

⑧ 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 makes is 5000.

Format: SYMBOL . . . . . ENDP

(b) Included files

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

LAD.HI

```
        ;*************************
        **********

LD  #10000
OUT#11000
    -
```

LAD.L01

```
        ;*************************
        **********

LOWSEQUENCE
LD  #14000
INR#1500
    -
```

LAD.L02

```
        ;*************************
        **********

LD  #14056
DST#1552, #1532, 0FFH
    -
Out #14033
```

LAD.L03

```
            ;*************************
            **********
LD  #10012
OUT#14500
     -
OUT#14010
```

CONV.DAT

```
            ;*************************
            **********
N90000H, 1H, 2H, 3H, 4H, 5H, 6H, 7H
     -
N90230FAH, 0FBH, 0FCH, 0FDH, 0FEH, 0FFH
```

MESSAGE.DAT

```
            ;*************************
            **********
N9024‘SPINDLE ALARM’
     -
N9323‘TROUBLE EXTERNAL DEVICE’
```

SYMBOL.DAT

```
            ;*************************
            **********
#10000JOG; Jog
     -
#79990  OIL; Coolant
```

(c) Source files

- 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.

- 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”.

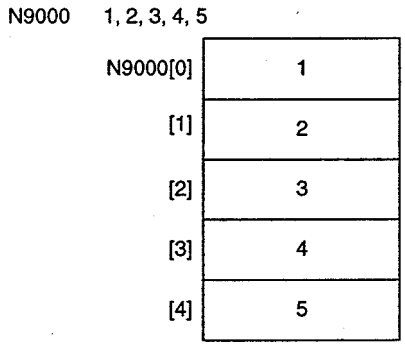
  Delimiter “,” is used between data.

- The table numbers that can be used are indicated below.

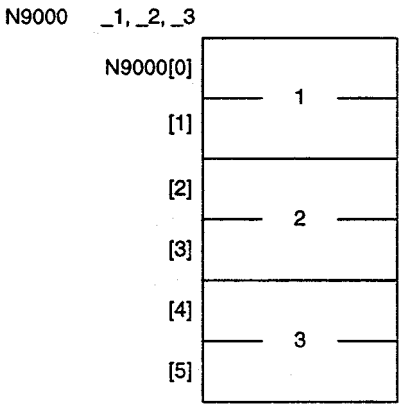
  #9000 to #9007: 256bytes

  #9008 to #9023: 128bytes

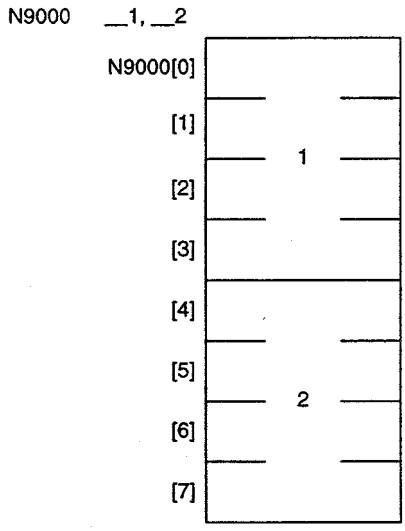
  In normal format, data are stored in the ladder table as byte data.



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



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



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 256data.

Omission of entry is treated as 0H.

- Message data source file

  Write the message data to be used by the macro instruction SUBP023.

  The message 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: 40words

  #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.

KENKYOU.SRC

```
            ;*************************
            **********

VERSION JXSD LADDER
LOWSTOPCOUNT1
```

LAKDHI.SRC

```
            ;*************************
            **********

HIGHSEQUENCE
LD  #14000
INR
     -
OUT#11010
```

LADLOW1.SRC

```
            ;*************************
            **********

LOWSEQUENCE
LD  #14000
INR
     -
OUT#11010
```

LADLOW2.SRC

```
            ;*************************
            **********

LOWSEQUENCE
LD  #14056
DST#1530, 1532, 0F FH
     -
OUT#14033
```

LADLOW3.SRC

```
            ;*************************
            **********
LOWSEQUENCE
LD  #10012
OUT#14500
     -
OUT#1410
RET
```

CONV.SRC

```
            ;*************************
            **********
CONVERSION#10012
N90000H, 1H, 2H, 3H, 4H, 5H, 6H, 7H
     -
N90230FAH, 0FBH, 0FCH, 0FDH, 0FEH, 0FFH
```

MESSAGE.SRC

```
            ;*************************
            **********
MESSAGE
N9024‘APINDLE ALARM’
     -
N9323‘TROUBLE IN EXTERNAL DEVICE’
```

SYMBOL.SRC

```
            ;*************************
            **********
SYMBOL
N10000JOG- Jog
     -
N19990OIL; Coolant
```

7.3 COMPILER

7.3.1 Compiler

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

(1) Starting the Compiler

The J1LCOMP is started by the following procedure.

J1LCOMP 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 name is set.
- If only "J1LCOMP" is input, the guide messages for inputting the parameters are displayed.
- Example: J1LCOMP 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 Complier Error List

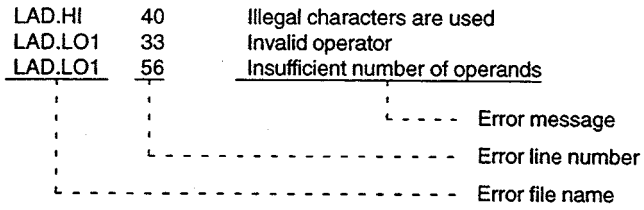
If an error occurs during compilation, the compiler outputs the error list file having the extension of ".ERR" with the same main file name as the input file.

It is also possible to designate the file name of the error list file at start of the compiler. 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 detected when the error list file is generated.

Error list file.



7.3.3 Complier 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

- Operand code check

  Permitted: LD, LD-NOT, AND . . .

  Not permitted: ABS, XOR-NOT . . .

- Operand number check

  Permitted: DEC #1001, OFFH . . .

  Not permitted: DEC #1001 . . .

- Operand address designation range check

  Permitted: LD #10001 . . .

  Not permitted: LD #10 . . .

- 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) Time 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 cheeked.

(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 file 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 laddr 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 limker 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 either RET nor RTI.
- The limker 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 desginated address.
- An error occurs if a version mumber is defined in more than object file.

7.4.2 Linker Operation

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

(1) Link Module File

It is necessary to create the link module file before starting the J1LLINK.

The object files to be linked are designated by this file.

(a) Link module file name

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

Example: FILE1.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 ladder are executed in the order they are designated in this file.

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

(2) Starting the Linker

J1LLINK 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 file 1 is assigned.

If only "J1LLINK" 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 linker instruction is generated in one output file.

Example: J1LLINK YELAD.LNK[ENTER]

Output file

YELAD.BIN Ladder execution file

7.5 List of Messages

7.5.1 Error Messages

1-line character 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 file open error.

7.5.2 Warning Messages

Output contact of OUT-instruction is defined duplicatedly.

Symbol-case-number exceed 5000.

8

Online Editing

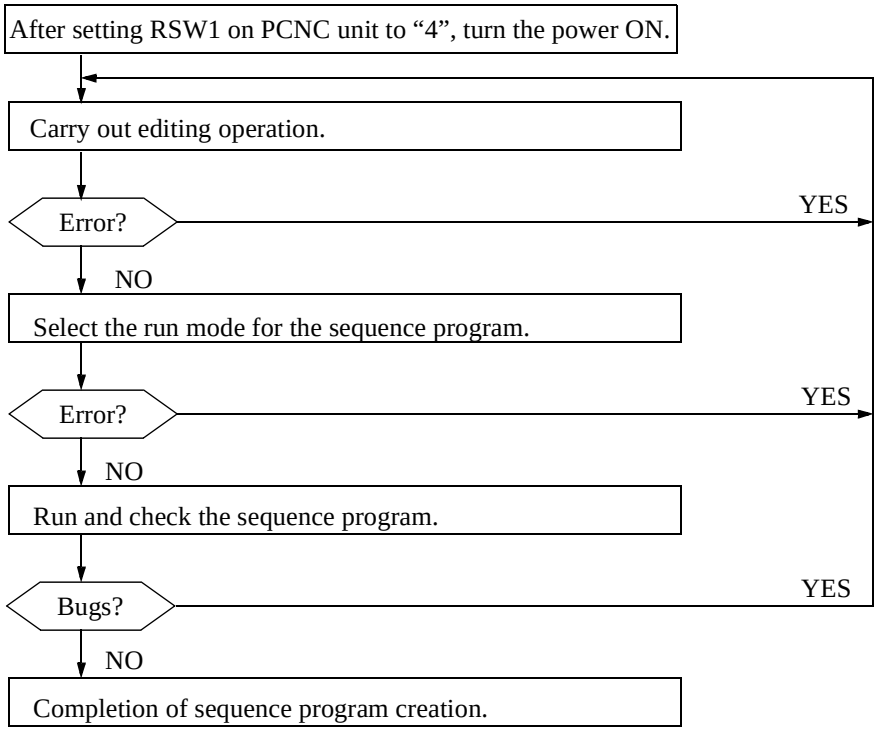
Chapter 8 describes on-line editing operations.

8.1 Out-line 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 New Sequence Program

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



① Test is set for “VERSION”.

② Sequence area is cleared and the following program is inserted automatically from the beginning.

- High sequence
- RTH
- ENDP
- LOWSEQENCE
- RET

- ENDP

③ The message data area is cleated.

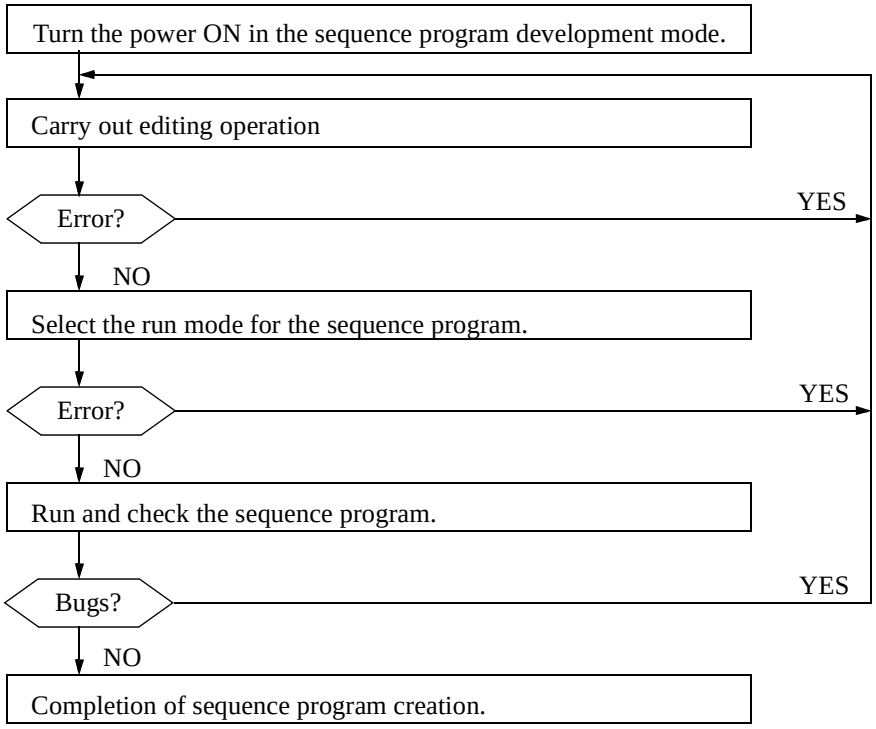
④ The conversion data area is cleated.

⑤ The symbol data area is cleated.

8.1.1 Creating a Sequence Program by Modifying the Existing Sequence Program

PCNC:Set the RSW (rotary switch) to “4” on the NC unit.

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

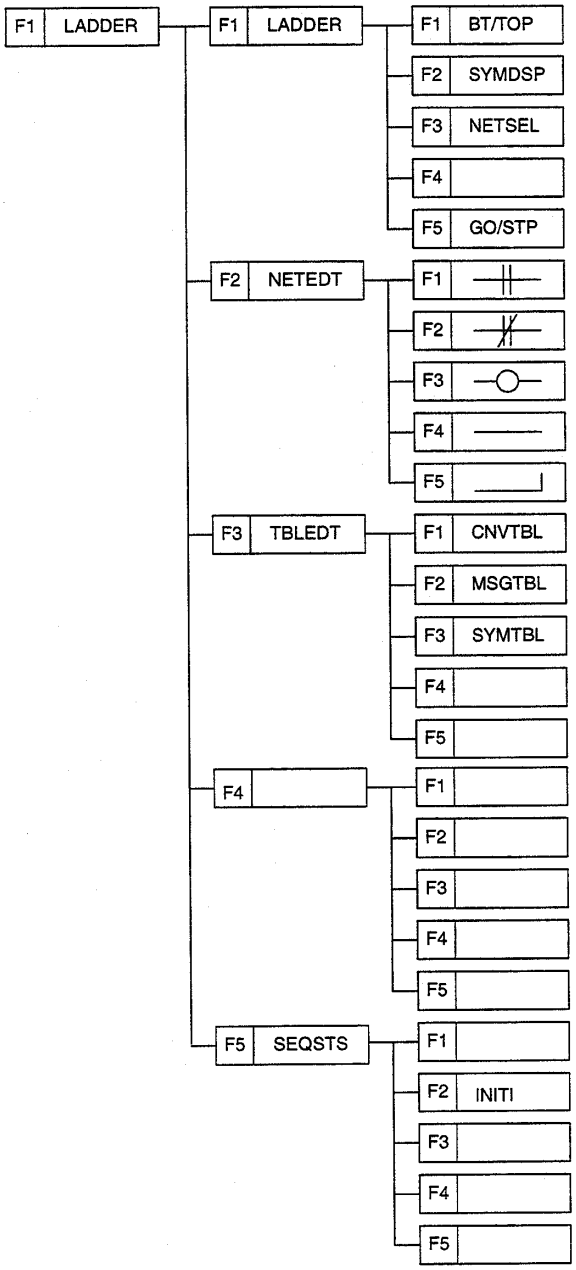


8.2 Function Structure and Display Screens

8.2.1 Function Structure

The condition for calling up the functions differ between J300 and PCNC.

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



Function Display	Conditions for Calling Up the Function
Always displayed	
In sequence program development mode	While the sequence program stopped.

8.2.2 Ladder Display Screen

Ladder display screen is shown below.

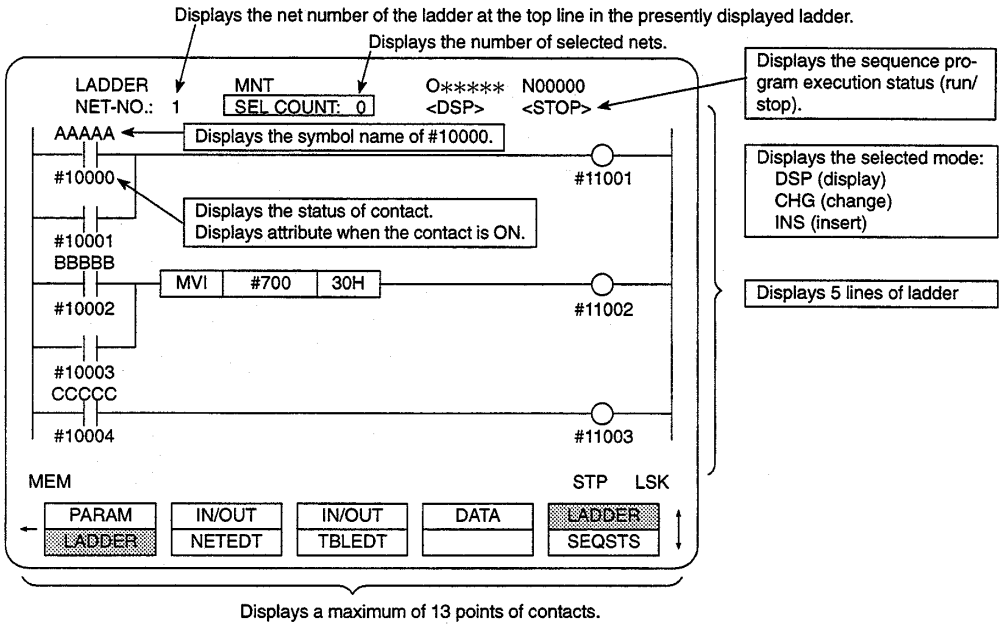


Fig. 8.1 Ladder Display Screen

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 8.3.1 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 ENTER key when the net number to be selected is displayed.

② The selection symbol 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!"

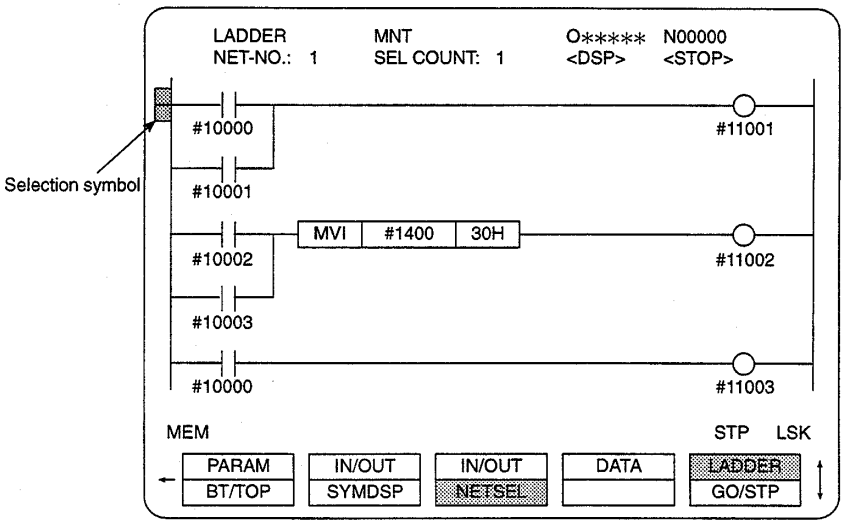


Fig. 8.2 Ladder Diagram Display Screen

③ Press the [NET SEL] function soft-key.

While the selected nets are collected, the message "COLLECTING" is displayed.

After the completion of met selection, the ladder of the collected nets is displayed and the message is cleared.

④ Press the [NET SEL] function soft-key once again, and the screen returns to the normal sequence ladder display screen.

SUPPLEMENT The selected status of the nets is cleared when the power is turned OFF.

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.

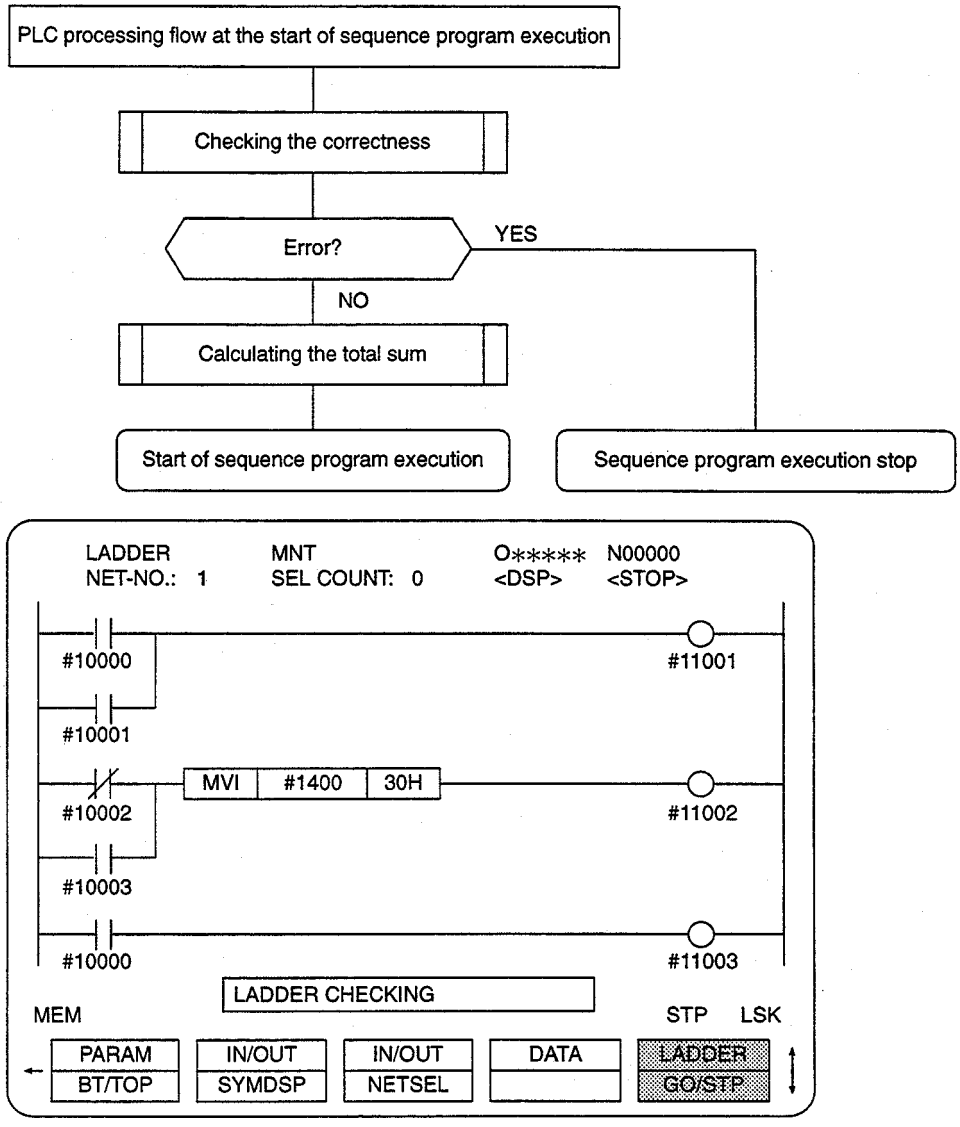


Fig. 8.3

Before the execution of the sequence program, its correctness is checked.

Table 8.3.4.1 Sequence Program Correctness Check

Check Items	Description
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).
SUBP calling up sequence check Warning message: SUBP CALL ERROR	The order and argument of the following are checked: APSH, TPSH, IPSH, 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 Edit 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. 8.4.

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.

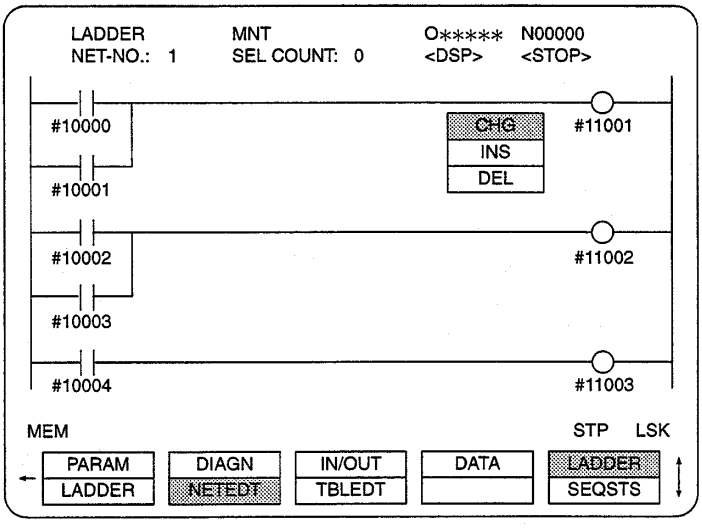


Fig. 8.4 Net Edit Function Screen

SUPPLEMENT

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 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 EDT] 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 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 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 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 indicated 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>.

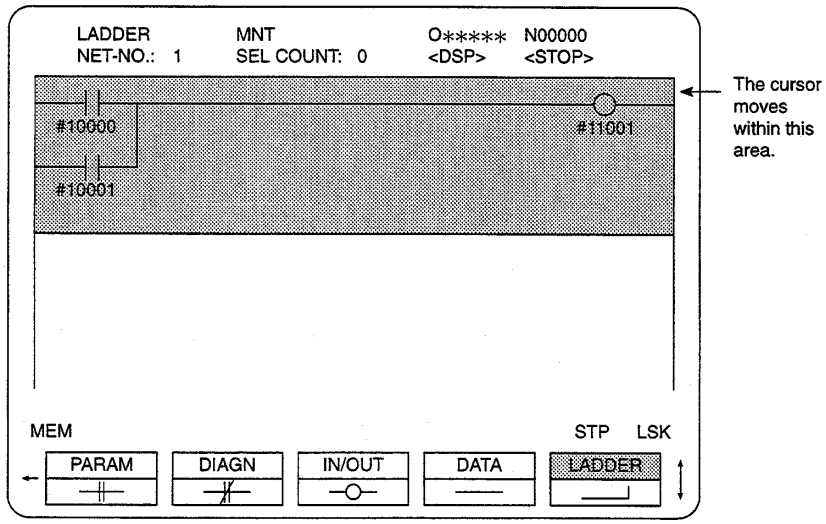
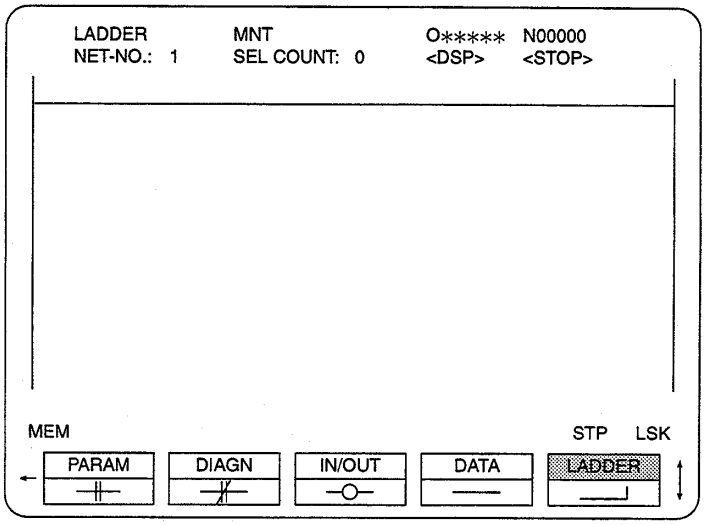


Fig. 8.5 Change Mode Screen

(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>.



(c) Delete mode

In this node, the selected net is deleted. At the mode display area, <DEL> is displayed.

The: DELETING" is displayed.

Fig. 8.7 and Fig. 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"

Select the contact to be detected by moving the cursor onto it, select DEL pop-up menu and press the ENTER key. This deletes the selected contact.

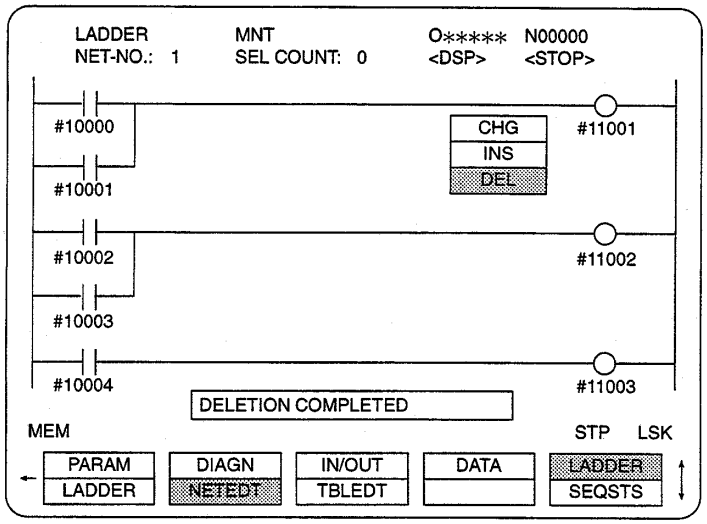


Fig. 8.7 Delete Mode Screen (Before Deletion)

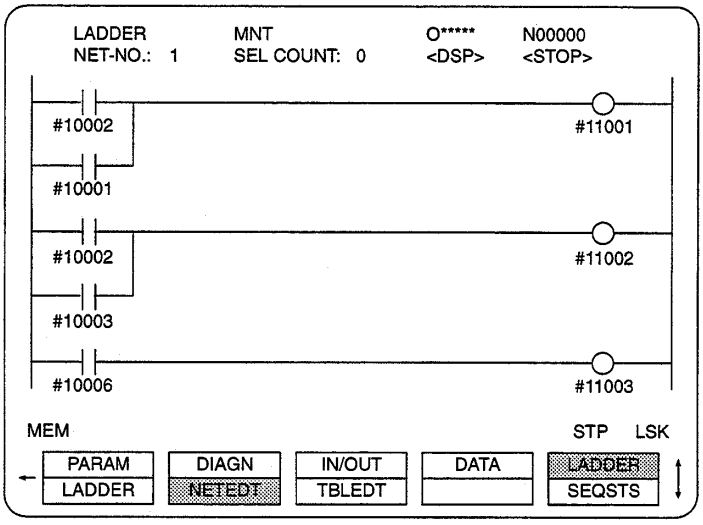


Fig. 8.8 Ladder Display Screen (After Deletion)

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.

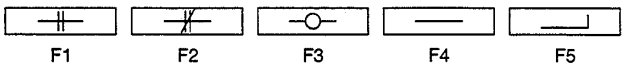
“CONTACT OVER!”

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.

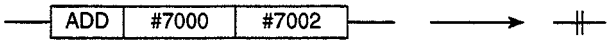
(1) Cursor Keys

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.
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.
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.
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

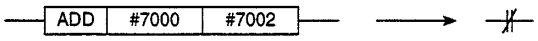
: Replaces the block where the cursor is positioned with the NO contact.

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.



: Replaces the block where the cursor is positioned with the NC contact.

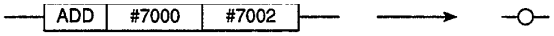
If this key is pressed when the cursor is positioned on the block which includes a register instruction, the remaining block is deleted as indicated below.



: Replaces the block where the cursor is positioned with the OUT coil.

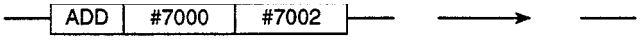
If this key is pressed when the cursor is positioned on the block which includes a register instruction, the remaining block is deleted as indicated below.

Note: An OUT coil can be input only at the 13th contact. If it is input at any other position, and error occurs and “INPUT ERROR” message us displayed.

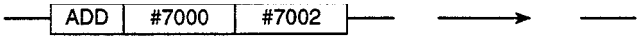


: Replaces the block where the cursor is positioned with a horizontal line “—”.

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.



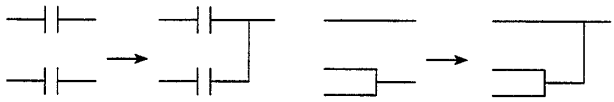
If this key is pressed when the cursor is positioned on the block of the horizontal line “—”, the horizontal line is deleted.



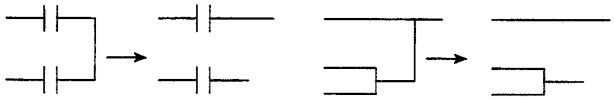
: Draws a vertical line on the block where the cursor is positioned and also on the block above this block.

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

ERASE: Deletes the instruction on which the cursor is positioned.

8.4.3 Inputting Contacts

(1) Inputting Contacts

① When a function soft-key presenting a contact is pressed, the message asking the input of the contact (switch) number is displayed.

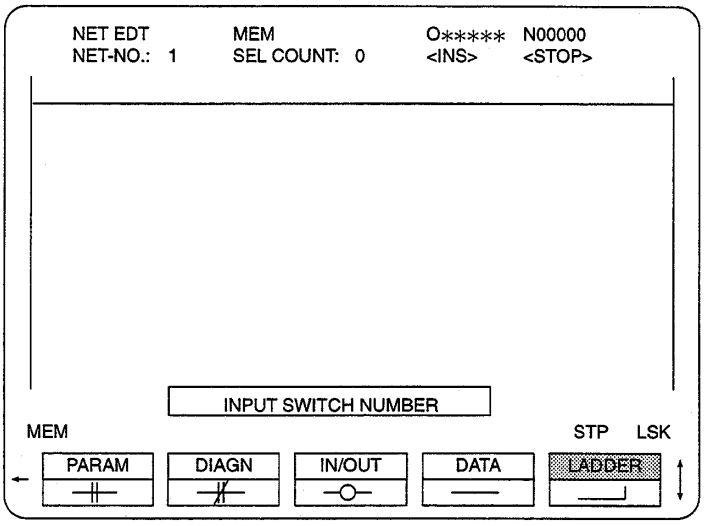


Fig. 8.9 Contact Input Screen

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, #1100 to #1175, #1200 to #1299, #1400 to #1699
⊣/⊢ #1800 to #2999, #3500 to #3699, #7000 to #7999.

⊣○⊢ : #1100 to #1163, #1200 to #1299, #1400 to #1699 #1800 to #2999, #3000 to #3159, #7000 to #7999.

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 ENTER:Correct input

#10000 ENTER: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.

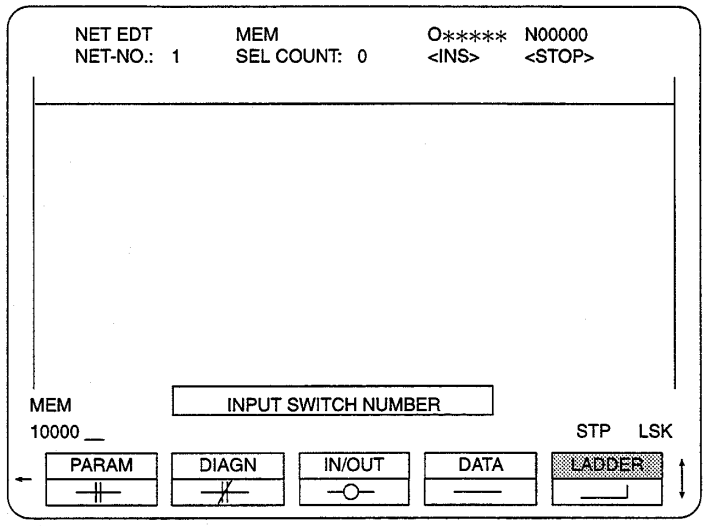
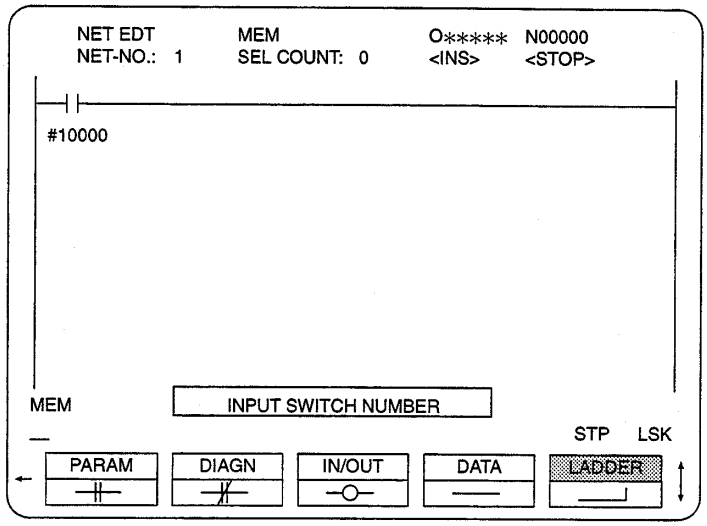


Fig. 8.10 Contact Input Screen

③ Enter “10000” and press the ENTER key. 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.



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 of 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 varietal/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.

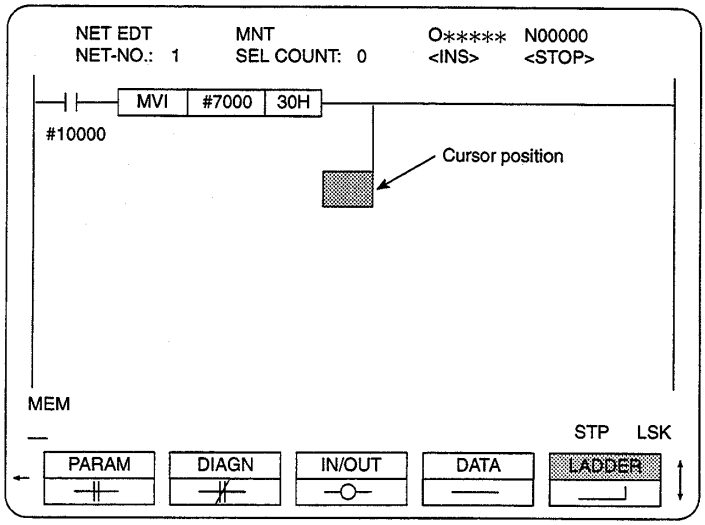


Fig. 8.12 Drawing a Vertical Line

8.4.5 Inputting Register Instructions

By keying in the first letter of 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 require register instruction can be written in this manner.

Table 8.4.5.1 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	XTI, 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	
ADC		COI		DST		IPSH		MULW	
ANR		CMP		DIN		IPSHD		MVLW	
ADD		CPI		DIVW				MCR	
ADDW		CPR		DSTW					
ADR		COR		DCRW					
APSH		CLRW							
		CMRW							
		CORW							
		CPRW							

[S]		[T]		[O]		[X]	
SUB	SUBP23	TIM		ORR		XRI	
SBI	SUBP25	TMR		ORI		XRR	
SUBW	SUBP27	TPSH				XOR	
SET	SUBP31					XNR	
SUBP3	SUBP32						
SUBP4	SUBP34						
SUBP5	SUBP35						
SUBP6	SUBP36						
SUBP7	SUBP37						
SUBP9	SUBP38						
SUBP11	SUBP39						
SUBP14	SUBP40						
SUBP17							
SUBP17							
SUBP18							

(3) Example of Input-CMP Instruction

The operation and screen display are explained below using the input of CMP instruction as an example.

① Key-in “C” to the key buffer area.

The pop-up menu showing five instructions beginning with “C” is displayed. The character “C” which has been keyed-in is not displayed in the key buffer display area.

To cancel the pop-up menu, press the [RST] key.

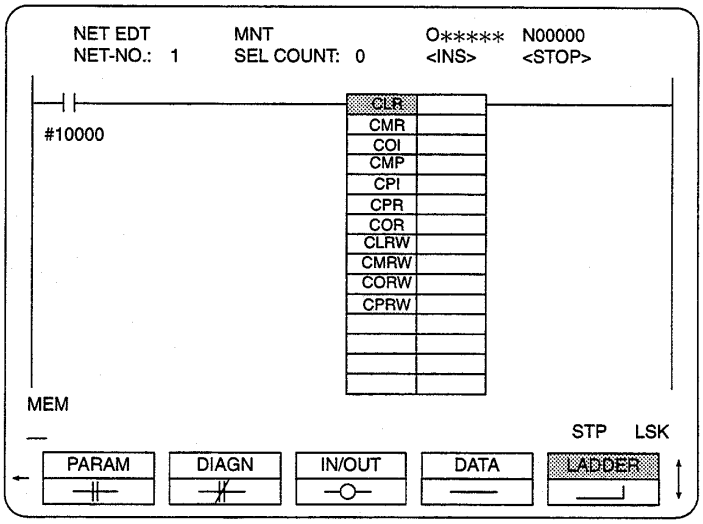


Fig. 8. 13 Pop-up Menu Screen after Keying-in “C”

② The CMP instruction format is displayed in the pop-up screen when “C” 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 CMP 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. 8.14. 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.

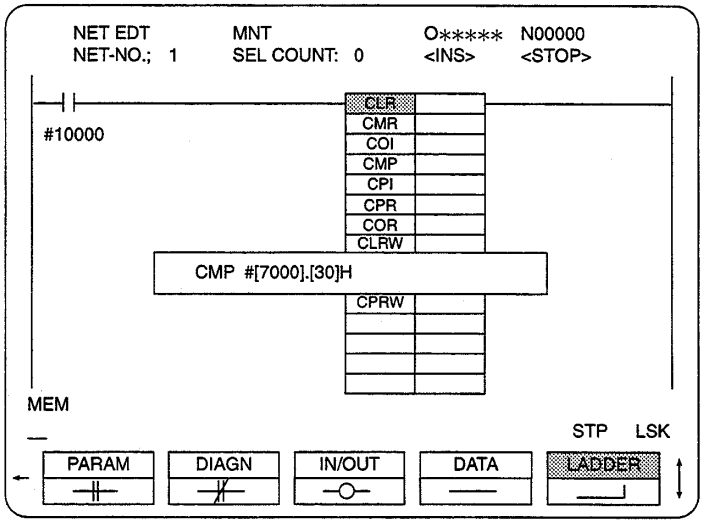


Fig. 8.14 CMP Instruction Input Screen

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.

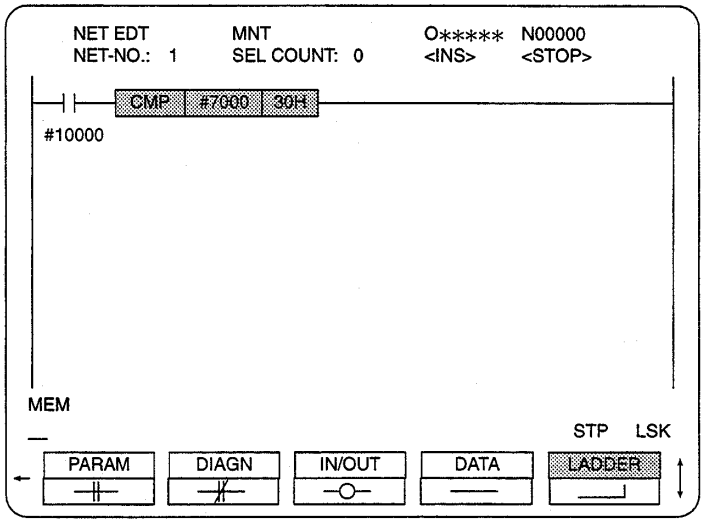


Fig. 8.15 Ladder Display after Inputting CMP Instruction.

(4) Operand Input Pop-up Menu Screen

(a) Types of operand input pop-up menus

Depending on the input register instruction, the corresponding operand input pop-up menu appears on the screen.

For the operands, check is made whether or not the input is allowed.

Table 8.4 Operand Input Pop-up Menu

Screen	Operand
JMP [1]	JMP, ADR
IPSH [7000]	IPSH, IPSHD, SUBP
INRW N [7000]	INR, DCR, CLR, CMR, INRW, DCRW, CLRW, APSH, PUSH, XOR, XNR
INRW N [9023]	TPSH
CPI# [7000], #[ ]H	TMR, ADI, SBI, ANI, ORI, XRI, DEC, COI, CMP, CPI, MVI
ADDW # [7000], #[ ]	TIM, ADD, SUB, ANR, OPR, XRR, CPR, COR, MOV, ADC, ADDW, SUBP, MULW, DIVW, CORW, CPRW, MVIW
DIN #[7000]. #[7000], [ ]H	DST, DIN, DSTW

(b) Resisters 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 8.5.

If a register outside the allowable range is input, the following warning message is displayed.

“INPUT ERROR!”

Table 8.4.5.2 Range of Registers that can be input for Individual Register instructions

Resister 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 8.6.

Table 8.4.5.3 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 256C		
IPSH	0 - FFFFH		
APSH	C		
PUSH	C		
TPSH	B		
IPSHD	-999999999 to 999999999		
SUBP	5, 6, 7, 9, 10, 11, 14, 17, 18, 23, 31, 32, 34, 35, 36, 37, 38, 39, 40		

(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”.

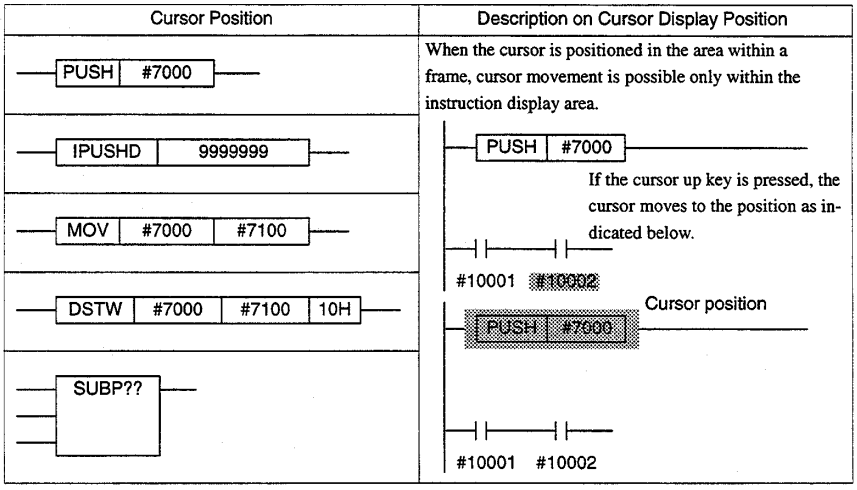
64_ Regarded as “64”.

(5) 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 8.7 Cursor Movement on the Register or SUBP instructions



(6) Patterns that does not Allow Register Instruction

Input of a register instruction is not allowed in the positions indicated below.

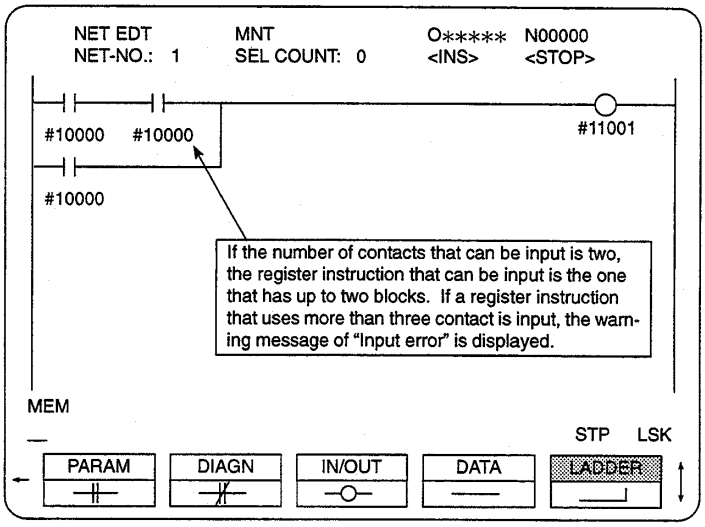


Fig. 8.16 Input Position of Register Instruction

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 Net Edit Function

(1) Operation Procedure

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.

① Check 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.

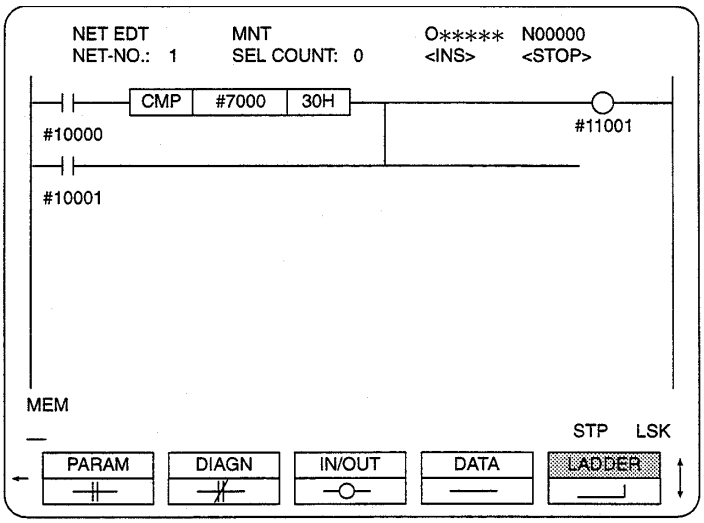


Fig. 8.17 Line Disconnection Error.

Example 2: In the case of the ladder shown in Fig. 8.18, the ladder is not one net and therefore connection error occurs.

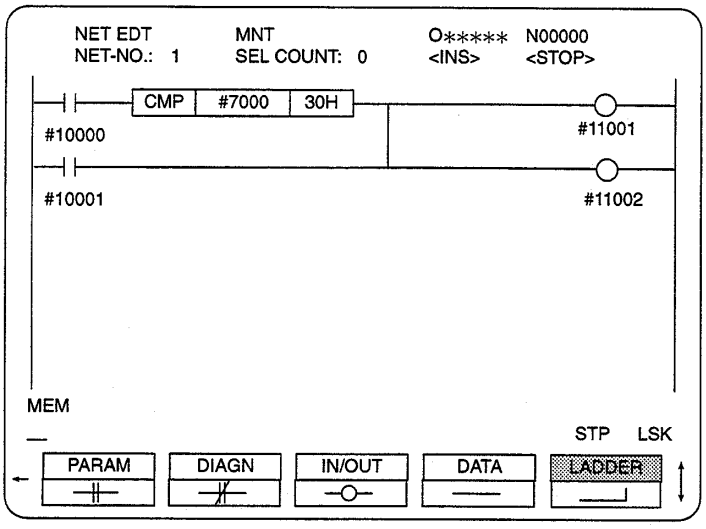


Fig. 8.18 Incorrect Net (More than One Ladder)

② Check the start instruction.

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.

The instructions to be checked are as follows:

CMP, DEC, PUSH, ADR, SET, RET, END, POP, IPSH, NOP, LD, LD-NOT, TPSH, APSH, SUBP23, IPSHD

③ Check the double-instruction.

The NC checks that no instruction is designated with any of the following instructions. If designated, the “DOUBLE INST ERR!” is displayed.

The instructions to be checked are as follows:

END, RET, RTH, IPSH, APSH, PUSH, TPSH, ADR, IPSHD

④ Check the single-instruction.

The NC checks whether some instruction is designated with any of the following instructions. If not designated, the “SINGLE INST ERR!” is displayed.

The instructions to be checked are as follows:

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

⑤ Check the analysis possibility.

The NC checks if the edited net can be converted into a sequence program.
If conversion is not possible, the “INVALID NET!” is displayed.

⑥ Check the sequence program size

The NC checks that a sequence program is within the allowable size.
The size of sequence program differs between J300 and J100.

The maximum size of a PCNC sequence program is 393,216 bytes which are approximately 24,500 steps (1step = 16bytes). If the size of the sequence exceeds 393,216 bytes the “SIZE OVER!” is displayed.

⑦ 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.

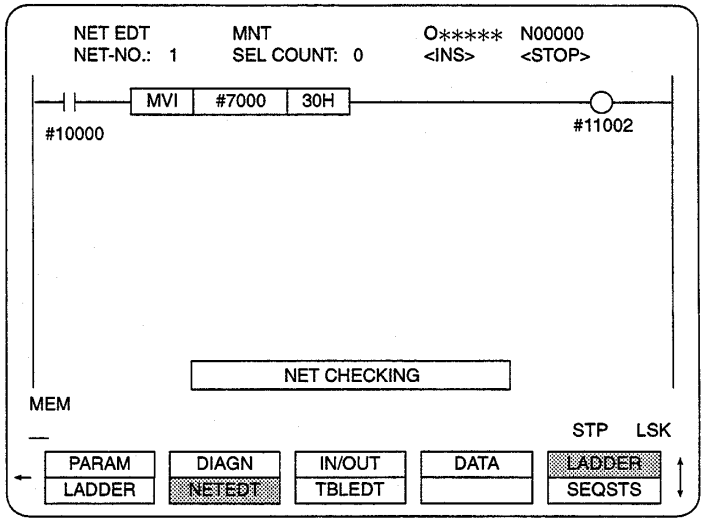


Fig. 8.19 Screen during Net Check

(2) 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 poser 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 Conversion Table

The conversion data that has 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 256bytes

N9008 - N9023 128 bytes

“FFH” is displayed for undefined conversion data.

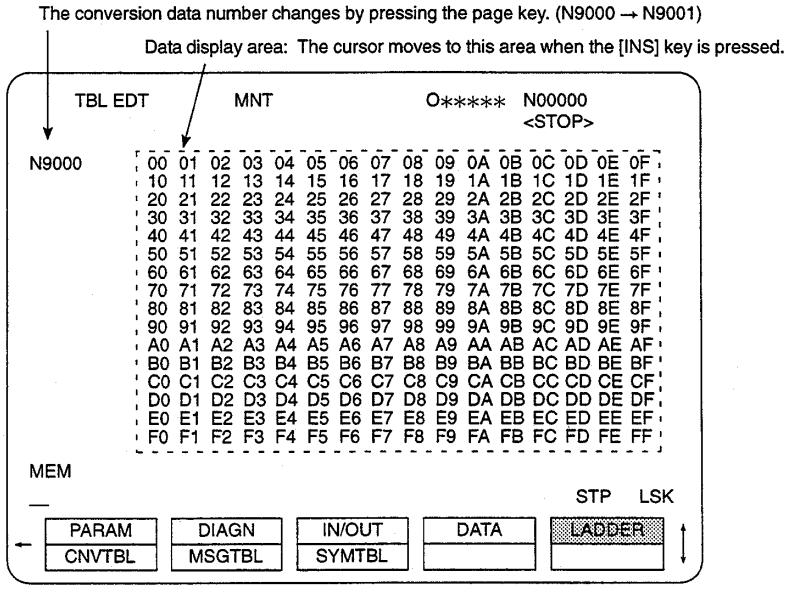


Fig. 8.20 Conversion Data Edit Screen

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 enter key.

Example: [4][2][W]

③ Press the [INS] key.

8.5.2 Editing the Data in Message Table

The message data that has 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 #9024 to #9323.

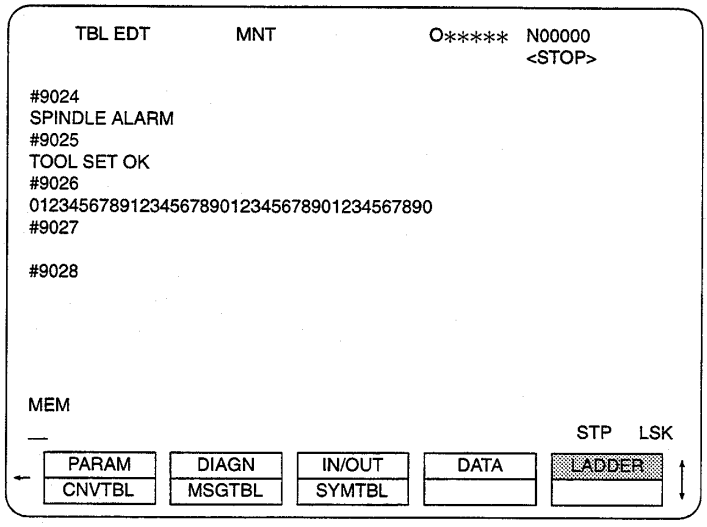


Fig. 8.21 message Data Edit Screen

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 numeric value and press the ENTER key.

8.5.3 Editing the Data in Symbol Table

The symbol name that has been set by using the pseudo instructon “SYMBOL” can be edited. As a symbol name, a character-string of upto 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.

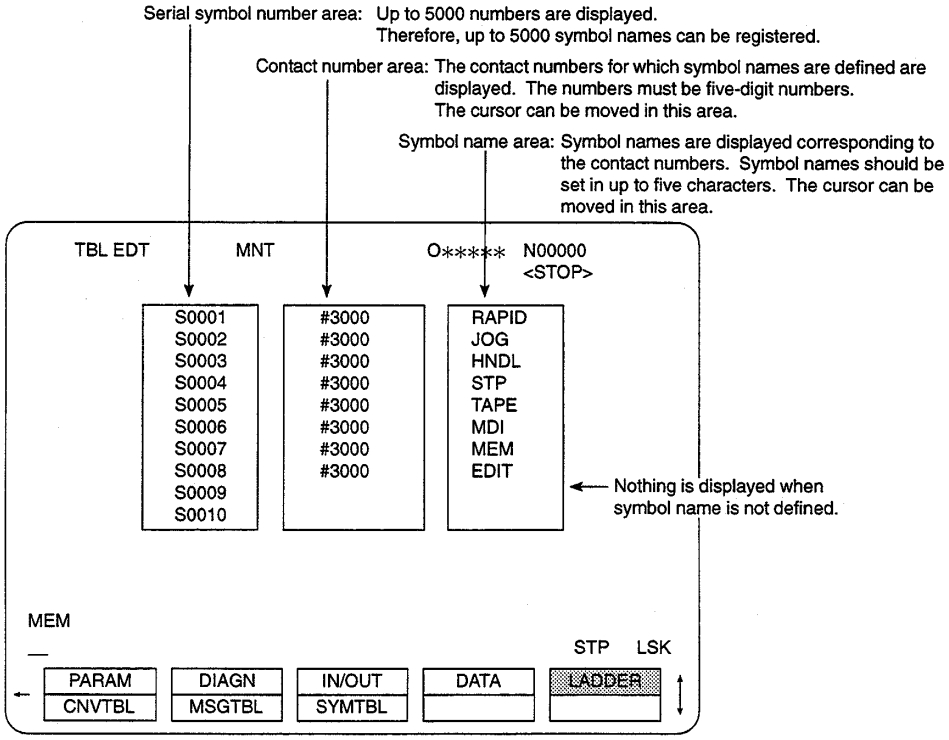


Fig. 8.22 Symbol Name Edit Screen

The symbol name editing procedure is explained below.

① Move the cursor to the contact number area 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.

SUPPLEMENT

1. When inputting a register number, it is not necessary to input “#”.
2. For a register number, only a numerical 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

From the CMOS area, you able to upload and download both Binary and Text files of the ladder sequence program.

8.6.1 Downloading the Sequence Program

The procedure used for down loading the sequence program (execution module *.BIN) stored in the Floppy disc to the NC is explained below.

① Once you have debugged you sequence, go to Utilities and Log in at Machinist level or higher.

② Select “Backup and Restore” icon.

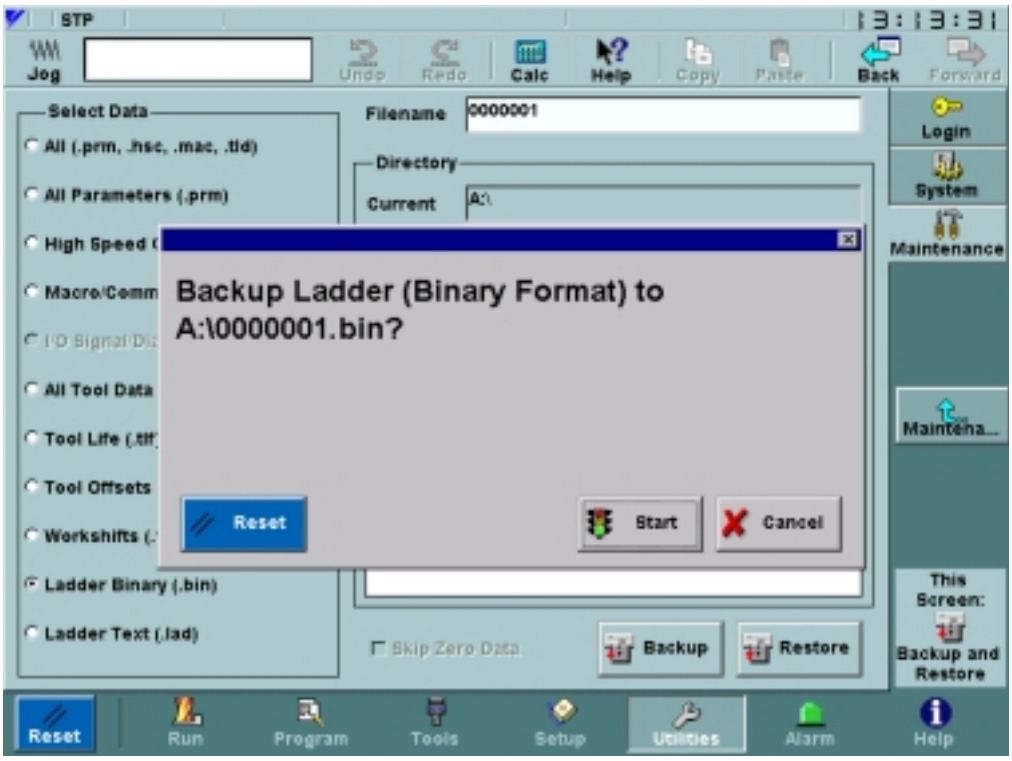


Fig. 8.23 IN/OUT FARM Function Screen

③ Select either Ladder Bin or Ladder Text for Backup and Restore functions to CMOS area.

8.8 List of Messages

Messages displayed during on-line editing are given in Tables 8.8, 8.9 and 8.10

8.8.1 List of Messages

Table 8.8.1.1 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 8.8.2.1 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 8.8.3.1 List of Alarm Messages

Message	Description
BACKUP THE SEQUENCE POGRAM	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

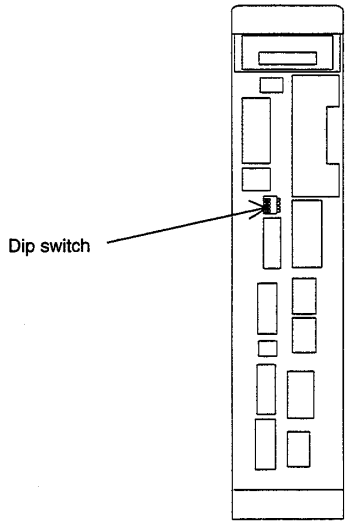
Downloading & Uploading Sequence Program

Chapter 9 describes the procedure for downloading and updating the sequence program from the flash ROM

9.1 Downloading Sequence Program (Floppy Disk → Flash Rom)

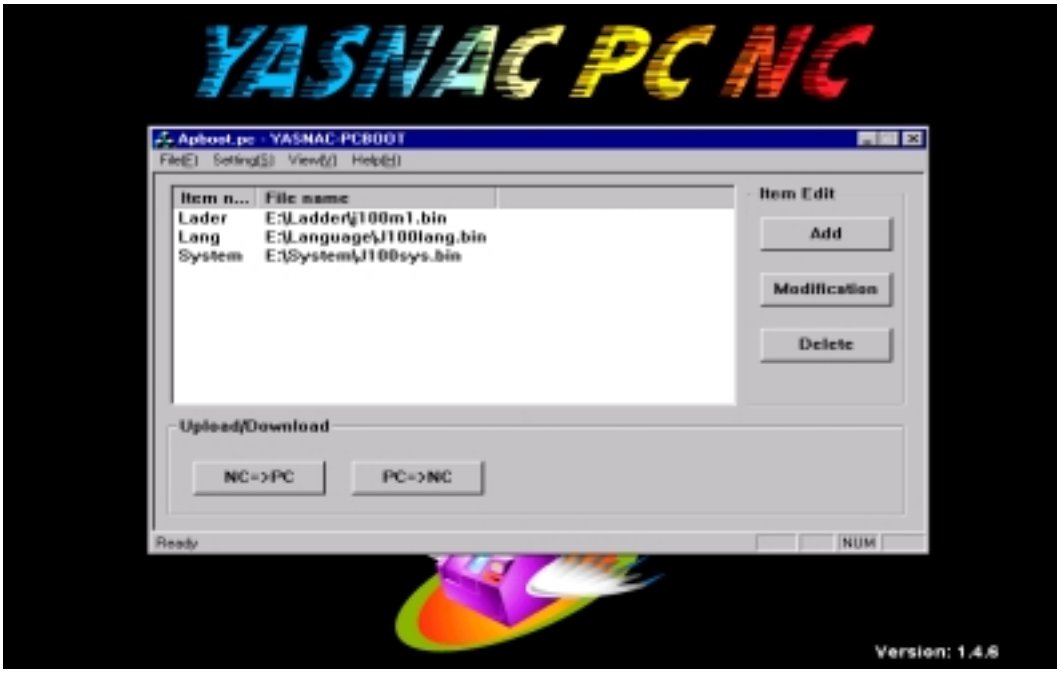
The procedure for downloading the PCNC sequence program from Floppy Disk to flash ROM is indicated below.

① Set the SW1 (dip switch) to “1” on the NC unit.



② Turn the power ON.

③ Insert a floppy disk with the Bin. file that you want to Download into Floppy Drive.



④ Highlight the “lader” path name. Hit the “Modification” button and make the path point to the a:\ drive.

⑤ Once you found your binary file hit enter and again highlight “lader” path name.

⑥ Next hit “PC -> NC” button to start download.

⑦ When finished hit “End” and reboot.

9.2 Uploading Sequence Program (Flash ROM→ Floppy Disk)

The procedure for uploading the sequence program from floppy disk to PC card is indicated below.

① Carry out steps ① to ⑤ described in section 9.1.

② Press the [NC → PC] function soft-key.

③ Turn the SW1 to OFF and turn the power OFF.

10

YASNAC Paradym-31 Sequence Program Development Environment Setup

Chapter 10 describes the YASNAC Paradym-31 sequence program development environment setup.

10.1 Cautions

- Some diagrams in this manual illustrate the removal of the cover or safety shield for detailed explanation purposes. When operating this machine, the cover or safety shield must be installed in the required position, and operation must be performed following the manual instructions.
- The diagrams and pictures in this manual represent typical examples, they may differ from the products received by users.
- This manual may be amended due to product retrofit, specification change, as well as manual ease-of-use improvement.
- If amended, the revised manual will be issued with a new document number.
- Product retrofitted by the user is beyond the scope of quality compensation provided by our company. Our company will not assume the responsibility for any injury or damage caused by retrofitted product.

Before usage (operation, maintenance, inspection), this manual and other relative documents must be read carefully to ensure that the machine will be operated correctly and safely. Furthermore, machine knowledge, safety information and cautionary items must also be fully understood before operating the machine.

After reading, this manual should be kept in a convenient location to ensure that it can always be readily available to the user during machine operation.

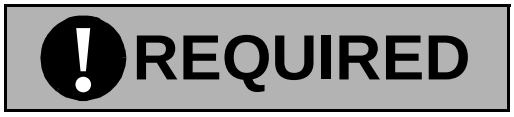
10.2 Warning Symbols

In this manual the following symbols for safe operation are used. Since the contents of the symbols are very important, they must be followed.

This symbol indicates that there is a risk of death or serious bodily injury resulting from improper operation.

This symbol indicates that there is a risk of minor or moderate injury to personnel and damage to equipment resulting improper operation.
The items indicated with this symbol may result in serious injury or equipment damage in some cases.)

This symbol indicates an action which is prohibited. For example, a symbol is used to indicate that smoking is prohibited for safety reasons.



This symbol indicates a required action. For example, this symbol indicates that grounding is required.

The **WARNING** symbol varies depending on ISO and JIS standards.

ISO Standards	JIS Standards
	

In this manual the **ISO standard symbol** is used.
Product warning display labels may use either the ISO or the JIS standard; however, both should be followed in the same way.

10.3 Icons

In this manual, the following icons are used, where necessary, to enable the user to quickly understand the contents.

This icon is used to identify an item which is important and is *necessary* to remember during operation.

It also indicates inputs or operations which should not be made because they may cause an alarm or malfunction, but not machine damage.

This icon indicates a program or operation example.

This icon indicates supplementary items or functions which should be remembered.

10.4 Preface

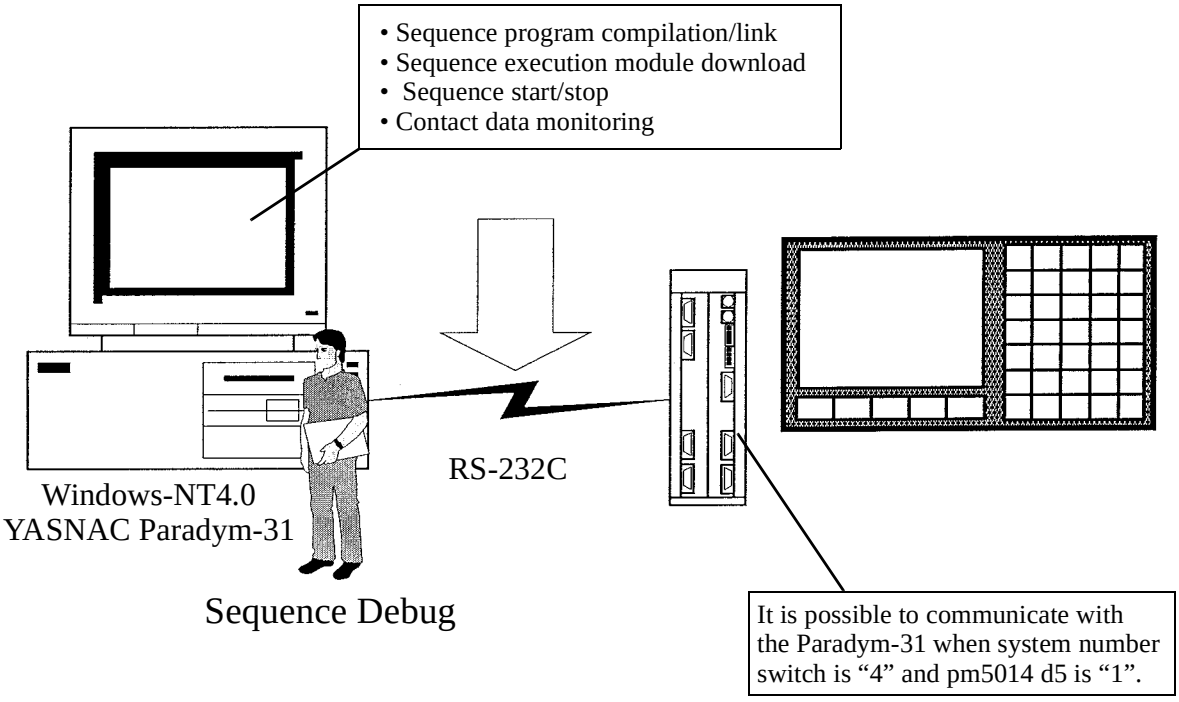
This document explains the procedures and restrictions for performing an on-line debug based on software YASNAC Paradym-31 (hereinafter, Paradym-31) used for sequence program development. Refer to **Paradym-31 Users Guide/Reference Book** for general Paradym-31 operation methods.

10.5 Function Outline

The sequence program debug can be performed by connecting a Paradym-31-installed PC and an NC device (YASNAC PCNC).

Possible debug functions performed by the Paradym-31 are as follows.

- Sequence program compilation and link
- Sequence execution module download
- Start/Stop of the sequence downloaded in the YASNAC PCNC
- Contact data monitoring



Sequence Debug

10.6 Software and Hardware Preparation

Necessary Software and Hardware

Provider	Software	Hardware
YASKAWA	• CNC system software used for YASNAC PCNC	• CNC system kit • RS-232C cable used for sequence program development
Outside Source	• Windows NT 4.0	• IBM PC compatible PC which runs WindowsNT4.0

Note: * Microsoft Windows NT: registered trademark of Microsoft Corporation.
* IBM PC/AT: registered trademark of International Business Machine Corporation.

RS-232C Cable Specification

RS-232C Connection of PC ← → CNC Device

PC Side	Pin No.			Pin No.	CNC Device Side
1st Serial Port (9pin)	3	SD	SD	7	CN13 (1st Port)
	2	RD	RD	6	
	7	RS	RS	5	
	8	CS	CS	4	
	6	DR	DR	3	
	4	ER	ER	2	
	5	SG	SG	1	

Cable Parts Specification

	Name	Model	Maker	Note
PC Side	Connector	17JE-13090-02 (D8A2) (soldered, with connector hood)	Daiichi Electronics (DDK)	D-Sub 9pin socket
CNC Side	Connector	10120-3000VE (soldered)	Sumitomo 3M (Inc.)	Half pitch 20pin
	Connector hood	10320-52A0-008	Sumitomo 3M (Inc.)	
	Cable	UL20276 AWG28 × 2 pairs with Shields		

10.7 Debug Preparation

Software Installation

First, the Paradym-31 must be installed on the PC to be used.
Refer to the **Paradym-31 Users Guide/Reference Book** for Paradym-31 installation methods.

Hardware Setup

(1) RS-232C Cable Connection
The RS-232C cable must be connected to the connector CN13 (1st port) of the CNC side and a usable port (port1 or port2) of the PC side.

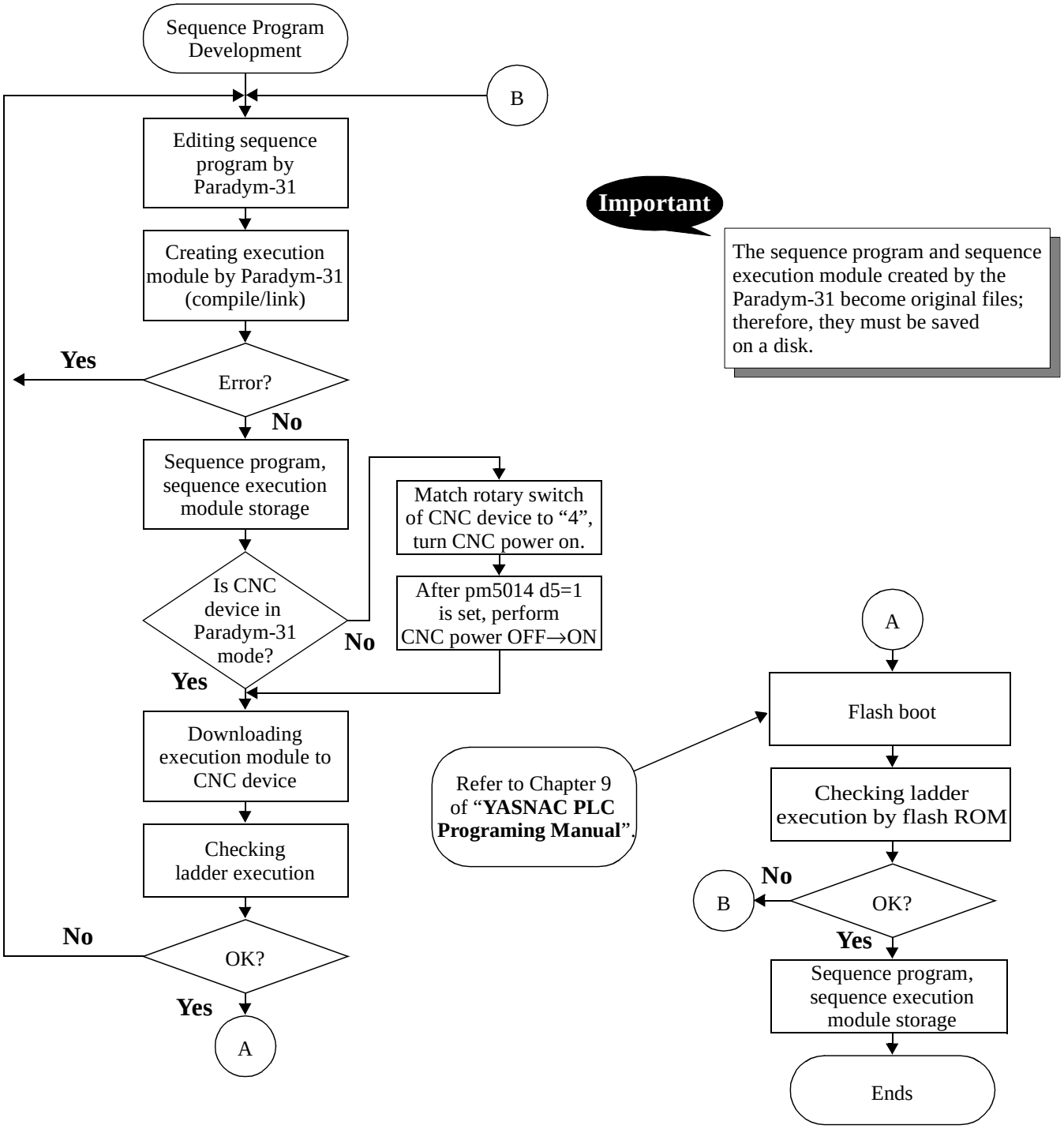
(2) Paradym-31 Debug Mode Set-up
Perform the following settings to set the Paradym-31 debug mode on the NC side.

- Match the system number switch to “4”.
- Set NC parameter pm5014 d5 as “1” (Paradym-31 communication effectiveness).

When pm5014 d5=1 is set, an NC alarm “**ALM0050: PARAM REQUIRING POWER OFF**” is displayed. Therefore, after the power is turned off, turn on the NC power again.

(3) RS-232C Communication Set-up
In the RS-232C communication set-up, if the port number used to connect the RS-232C is selected as a communication port number, communication speed will be automatically set-up. The CNC side will be automatically set-up by the above Paradym-31 debug mode set-up.

10.8 Sequence Development Procedures Based on the Paradym-31



10.9 Restrictions in Paradym-31 Debug Mode

The following restrictions exist when the CNC device is in the Paradym-31 debug mode.

(1) The ladder display and ladder on-line edit function of the CNC device are disabled. When the ladder display key is pressed, the warning message "**P31 is being selected**" is shown on the CNC display.

(2) The input/output of the NC data (parameter, working program, offset data) based on the Paradym-31 is disabled. (In the case of YASNAC J100M, J100L, parameter input/output based on PC card is possible).

10.10 Additional Parameters

Important

pm5014 d5: Paradym-31 debug mode selection parameter

0: Invalidate the communication between the Paradym-31 and CNC device
1: Validate the communication between the Paradym-31 and CNC device

- When this parameter is changed, the NC alarm "**ALM0050: PARAM REQUIRING POWER OFF**" is displayed. Therefore, after the NC power is turned off, turn the NC power on again.
- This parameter is enabled only when the rotary switch used for system set-up is "4".

11

YASNAC Paradym-31 Specifications of Dynamic Link Library (DDL)

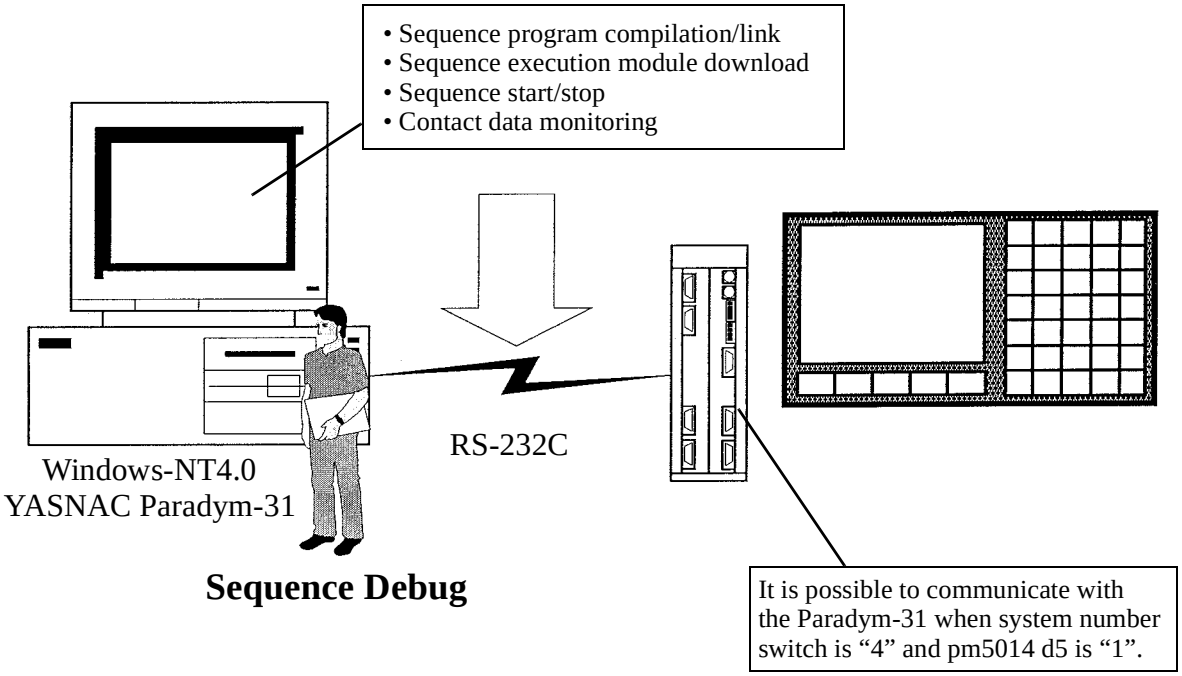
Chapter 11 describes the YASNAC Paradym-31 specifications of Dynamic Link Library (DDL) used for sequence development

11.1 Outline

This specification sheet explains the **D**ynamic **L**ink **L**ibrary (hereinafter, **DLL**) (for Windows) which enables the connection and on-line debug of sequence program development software YASNAC Paradym-31 (hereinafter, Paradym-31) and a CNC device (YASNAC PCNC series). (The Paradym-31 has three debug methods: dual port, Ethernet, and RS-232C. YASNAC uses only the RS-232C.)

11.2 System Outline Diagram

The YASNAC sequence program debug is performed by connecting and communicating the Paradym-31 with the CNC device via the RS-232C.
Refer to **YASNAC Paradym-31 Sequence Program Development Environment Set-up Manual** for the cable specification.



11.3 Paradym-31 Communication Effectiveness and Ineffectiveness

In order to communicate with the Paradym-31, the following conditions must be satisfied.

- CNC system number switch must be set to “4”.
- Paradym-31 communication effectiveness parameter (pm5014 d5=1) must be ON.
  When the pm5014 d5 is changed, the NC alarm **“ALM0050: PARAM REQUIRING POWER OFF”** is displayed. Therefore, it is necessary to turn on NC power after it has been turned off.
  Once the above conditions are satisfied, CNC device communication is automatically set.

  Restrictions during Paradym-31 communication effectiveness:
- Ladder display does not occur when the Paradym-31 communication is performed.
  If the ladder display function key is pressed, the warning “**P31 is being selected!**” occurs.

11.4 Where is the DLL Placed?

The following diagram illustrates where the DLL is placed.

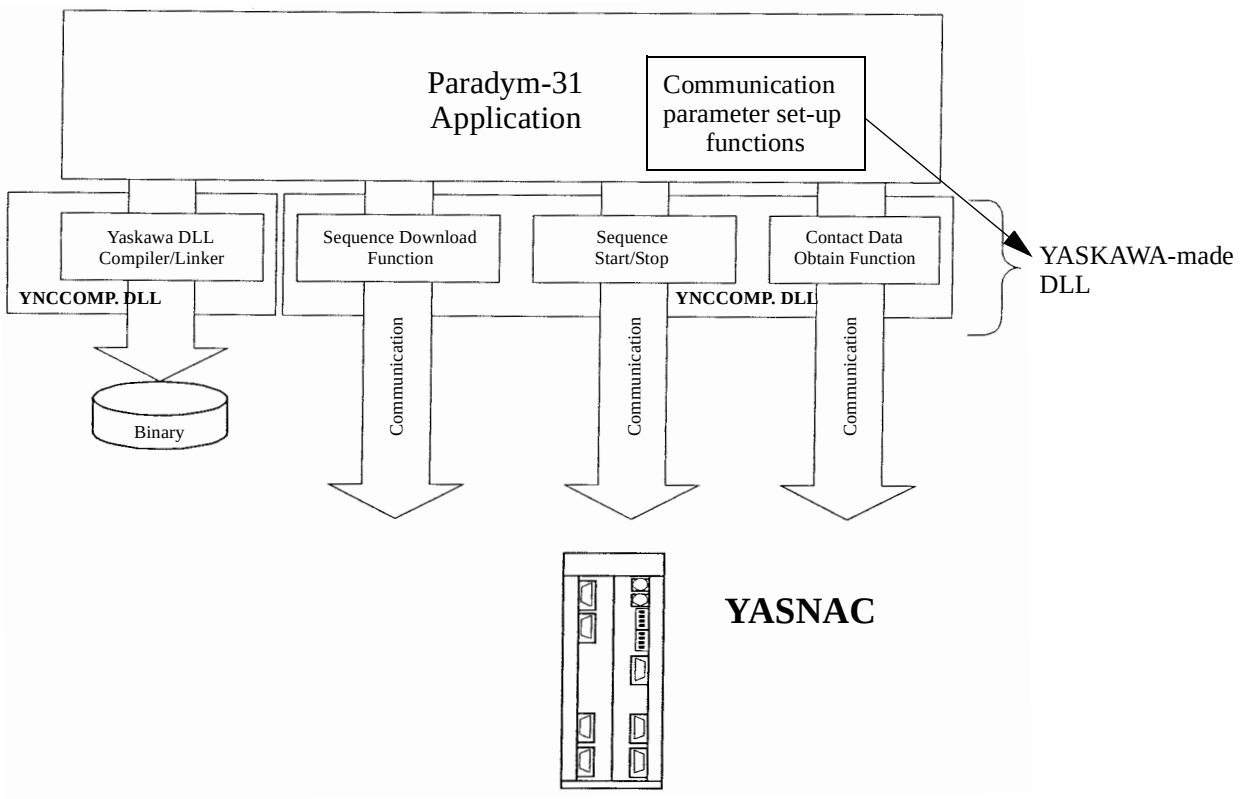


Figure 1: Placement of Paradym-31 DLL

11.5 Types of DLL and Functions that Compose the DLL

The following two types of DLL are to be created.

11.6 DLL for YASNAC Sequence Compile/Link

ynccomp.dll

Function Name	Meaning
lcomp()	Ladder compiler
llink()	Ladder linker

11.7 DLL for Communication Debug

ynccomm.dll

Function Name	Meaning
YncSetupComm()	Communication port set-up
Download()	Sequence download
ControlProg()	Ladder start/stop function
Getvalue()	Contact, byte data obtain

11.8 Each DLL

11.8.1 Ladder Compiler

INT STS = lcomp(char*psrcfile,char*pobjfile,char*perrfile);

psrcfile[64]:	Name of the file to be compiled
pobjfile[64]:	Name of newly compiled file
perrfile[64]:	Name of the file that saves compile error data

STS =

0:	Successful
Other than 0:	Number of error or warning

Basically, the ladder compiler contains the YASNAC J100 specification; however, since the following changes have been added, it is considered the Paradym-31 DLL ladder compiler.

(1) Compatible with 16-bit Message Data
16-bit characters are converted to 32-bit SHIFT-JIS and saved in an execution module, in order to correspond to 16-bit message data in the DLL ladder compiler.
(In conventional specifications, the message data is created by the 32-bit SHIFT-JIS.)

(2) Possible to duplicate symbol name definition for one contact number
The Paradym-31 can give more than two (a maximum of 5000) symbol definition names to one contact number. Therefore, the compiler for the YASNAC is also capable of this function.

For example: #10000 **STEP**
#10000 **STEP2**

(In conventional specifications, the compilation error "**Variable number has duplicated contact definition**" occurs.

Process When An Error Occurs
When a compilation error occurs, the file that saves compiler error data is automatically opened from a memo pad.

11.8.2 Ladder Linker

INT STS=llink(char*plnkfile,char*pbinfile,char*perrfile);

lnkfile[64]:Name of the file to be linked
binfile[64]:Name of newly created binary file
perrfile[64]:Name of the file that saves link error data

STS=

0: Successful
Other than 0: Error

When a link error occurs, the file that saves link error data is automatically opened from a memo pad.

When the link is successful, the sum value of a binary file is displayed in the message box.

11.8.3 Communication Port Set-up

BOOL yncSetupComm(WIZCOMM*,HWND)

BOOL STS=TRUE/FALSE
struct{
BOOL OPEN/CLOSE; TRUE=OPEN FALSE=CLOSE
INT COMMTYPE; RS232C,ETHERNET,DUALPORT
#DEFINE SERIALCOMM1; Serial communication selection
#DEFINE DUALPORTCOMM2;Communication selection via bus (unused)
#DEFINE ETHERNETCOMM3;Communication selection via Ethernet (unused)
INT PORTNO Can be designated by 1 ~ 4
#DEFINE COM11; Port number for RS232C
#DEFINE COM22; Port number for RS232C
#DEFINE COM33; Port number for RS232C
#DEFINE COM44; Port number for RS232C
CHAR IP_ADDR[16];15 characters + termination
}WIZCOM
HWND:Handling number of WIZDOM's main window
Main window handling is obtained from WIZDOM application. It is used for Windows display during error occurrence.

The communication speed is automatically set-up. When an error occurs, the following message is displayed in the message box.

Message Contents	Meaning
Already Used	It has already been used.
Communication Error	An error occurs during communication.
Communication Time Out	There is no response.
Communication Parameter Error	An error occurs in a communication parameter.

11.8.4 Sequence Download

BOOL Download(CHAR*pfile_name, HWND,Char*pProjname,charpProgname)

BOOL STS=TRUE/FALSE

pfile_name[64];	Name of download file with path (i.e. c:¥test¥test¥main.bin)
pProjname[64];	Project name, no extension name
pPrognname[64];	Program name, such as main, no extension name
HWND	Handling number of WIZDOM's main window

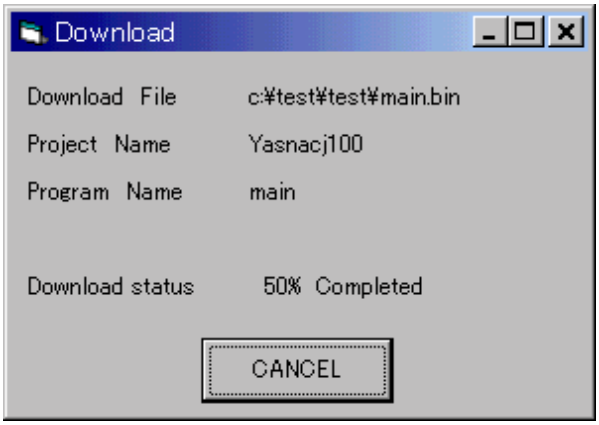
HWND: WIZDOM handling is obtained from the WIZDOM application.

When an error occurs, the following message is displayed in the message box.

Message Contents	Meaning
Ladder Check Sum Error	Sequence sum value is wrong.
Communication Error	An error occurred during communication.
Communication Time Out	There is no response.
Port Open Error	Port is not open.
Download File Open Error	Download file cannot be opened.
Ladder Running	Sequence is executing.
Already Used	It has already been used.

Download Status Display

The download status is shown as follows:



11.8.5 Ladder Start/Stop Functions

BOOL ControlProg(BOOL TRUE/FALSE,HWND)

BOOL STS=TRUE/FALSE
TRUE=Start
FALSE=Stop
HWND: Handling number of WIZDOM's main window (for message display)

When an error occurs, the following message is displayed in the message box.

Message Contents	Meaning
Ladder Start	Ladder execution began.
Already Started	Start process is complete.
Ladder Stop	Ladder execution stopped.
Already Stopped	Stop process is complete.
Ladder Check Sum Error	Sequence sum value is wrong.
Communication Error	An error occurs during communication.
Communication Time Out	There is no response.
Port Open Error	Port is not open.
Already Used	It has already been used.

11.9 Functions for Obtaining Contact and Byte Data

BOOL Getvalue(Word*Address,Byte*Bitnum,Byte*Datatype,Dword*pData,HWND)

*Address: A maximum of 100 contacts in 4-digits is possible
Array is ended by 0.
*Bitnum: It is possible to set up within 0 ~ 7 (effective when Datatype is 0.).
*Datatype: 0=bit,1=byte,2=word,3=dword
*pData: Obtain contact data pointer
1 data is fixed as 4 bytes.
Answer data during bit requesting returns by 0 or 1.
HWND: Handling number of WIZDOM's main window
Main window handling is obtained from WIZDOM application.
The status of start/stop is notified.

When an error occurs, the following message is displayed in the message box.

Message Contents	Meaning
Communication Error	An error occurred during communication.
Communication Time Out	There is no response.
Port Open Error	Port is not opened.
Request Data Error	Requested contact is wrong.
Already Used	It has already been used.

Note: Although data can be obtained that extends over the range of the I/O area or the NC standard signal diagnosis number area using the contact and byte data obtaining function (Getvalue), it is indefinite.

11.10 Error Message that Occurred in DLL Function

In each DLL, an error status is displayed in the message box during error occurrence.
For example:

11.11 Additional Parameters

pm5014 d5: Paradym-31 debug mode selection parameter
0: Invalidate the communication between the Paradym-31 and CNC device
1: Validate the communication between the Paradym-31 and CNC device.

- This parameter set-up requires power-off.
- This parameter is enabled only when the system number switch is “4”.

11.12 Other Restrictions

(1) If NC power is input when system number switch is “4”, and the pm5014 d5 is “1”, the ladder display is disabled.

1

Paradym-31 Reference Table

Appendix 1 describes the reference table in Paradym-31

APPENDIX 1.1

1	TIM	PV-Timer setting (decimal value) CV-Timer output (variable)
2	TMR	PV-Timer setting (variable) CV-Timer output (variable)
3	INR	D – register (variable)
4	DCR	D – register (variable)
5	CLR	D – register (variable)
6	CMR	D – register (variable)
7	ADI	A – Numeric Value (decimal value) B – Register (variable)
8	SBI	A – Numeric Value (decimal value) B – Register (variable)
9	ANI	A – Numeric Value (decimal value) B – Register (variable)
10	ORI	A – Numeric Value (decimal value) B – Register (variable)
11	XRI	A – Numeric Value (decimal value) B – Register (variable)
12	DEC	A – Numeric Value (decimal value) B – Register (variable)
13	COI	A – Numeric Value (decimal value) B – Register (variable)
14	CMP	A – Numeric Value (decimal value) B – Register (variable)
15	CPI	A – Numeric Value (decimal value) B – Register (variable)
16	MVI	A – Numeric Value (decimal value) B – Register (variable)
17	ADD	A – R1 (variable) B – R2 (variable)
18	SUB	A – R1 (variable) B – R2 (variable)

19	ANR	A – R1 (variable) B – R2 (variable)
20	ORR	A – R1 (variable) B – R2 (variable)
21	XRR	A – R1 (variable) B – R2 (variable)
22	CPR	A – R1 (variable) B – R2 (variable)
23	COR	A – R1 (variable) B – R2 (variable)
24	MOV	A – R1 (variable) B – R2 (variable)
25	DST	A – Numeric value (decimal value) B – R1 (variable) C – R2 (variable)
26	DIN	A – Numeric value (decimal value) B – R1 (variable) C – R2 (variable)
27	ADC	A – R1 (variable) B – R2 (variable)
28	ADDW	A –WR1 (variable) B – WR2 (variable)
29	SUBW	A –WR1 (variable) B – WR2 (variable)
30	MULW	A – R1 (variable) B – WR2 variable)
31	DIVW	A – R1 (variable) B – WR2 variable)
32	INRW	D – register (variable)
33	DCRW	D – register (variable)
34	CLRW	D – register (variable)
35	CMRW	D – register (variable)
36	CORW	A –WR1 (variable) B – WR2 (variable)
37	CPRW	A –WR1 (variable) B – WR2 (variable)
38	MVIW	A – Numeric value (decimal number)

		B – Register (variable)
39	DSTW	A – Numeric value (decimal number) B – WR1 (variable) C – WR2 (variable)
40	NOP	
41	MCR	
42	END	
43	RTI	
44	JMP	Label (decimal number)
45	ADR	Label (decimal number)
46	SUBP003	A – working address (Variable)
47	SUBP004	A – working address(Variable)
48	SUBP005	A – Preset value (decimal number or variable) B – working address (variable) C – Counter address (variable)
49	SUBP006	A – Target position address (variable) B – Preset position address (variable) C – No of positioning points (numeric value) D – Result address (variable)
50	SUBP007	N – No of conversion data items (decimal number) A – Conversion standard data address (variable) B – No of PLC (decimal number) C – Output address (variable)
51	SUBP009	A – Write Pattern (decimal number or a variable) B – Number of Bytes (decimal value)
52	SUBP011	A – Register (variable)
53	SUBP014	A – Input data address (variable) B – Output data address (variable)
54	SUBP17	N – Data table size (decimal number) A – Start address of data table (variable)
55	SUBP18	N – Data table size (decimal number) A – Start address of data table (variable) B – Address storing the number in table (variable) C – I/O data storing address (variable)
56	SUBP023	A – Message control address (variable) B – size of message control address (decimal number) C – Decimal screen number (decimal number) D – Start address of PLC table (decimal number)

57 SUBP025 A – Code data area (variable)
B – Start address of the decode table (variable)
C – Max. decode number (numeric value)

58 SUBP027 A – Size of conversion table (decimal number)
B – Conversion standard data address (variable)
C – Table number of conversion data (decimal number)
D – Byte/Word/DWord (decimal number)
E – Conversion data output address (variable)

59 SUBP031 A – Conversion data address (variable)
B – Converted data address (variable)

60 SUBP032 A – Input data address (variable)
B – Comparison data address (variable)
C – Result output address (variable)
N – Byte/Word/Dword (decimal number)

61 SUBP034 N – Size of data table (decimal number)
A – Start address of data table (variable)
B – Search data address (variable)
C – Byte/Word/Dword (decimal number)
D – Result address (variable)

62 SUBP035 N – Size of data table (decimal number)
A – Start address of data table (variable)
B – Address storing the number in table (variable)
C – Byte/Word/Dword (decimal number)
D – I/O data storing address (variable)

63 SUBP036 A – Augend data address (variable)
B – Addend data address (variable or decimal number)
C – Operation type (decimal number)
D – Result output address (variable)

64 SUBP037 A – Minuend data address (variable)
B – Subtrahend data address (variable or decimal number)
C – Operation type (decimal number)
D – Result output address (variable)

65 SUBP038 A – Multiplicand data address (variable)
B – Multiplier data address (variable or decimal number)
C – Operation type (decimal number)
D – Result output address (variable)

66 SUBP039A – Dividend data address (variable)
B – divisor data address (variable or decimal number)
C – Operation type (decimal number)
D – Result output address (variable)

67SUBP040A – Setting data (numeric value)
B – Byte length (decimal number)
C – Output address (variable)

YASKAWA ELECTRIC AMERICA, INC.
Chicago-Corporate Headquarters 2121 Norman Drive South, Waukegan, IL 60085, U.S.A.
Phone: (847) 887-7000 Fax: (847) 887-7310 Internet: http://www.yaskawa.com

MOTOMAN INC.
805 Liberty Lane, West Carrollton, OH 45449, U.S.A.
Phone: (937) 847-6200 Fax: (937) 847-6277

YASKAWA ELECTRIC CORPORATION
New Pier Takeshiba South Tower, 1-16-1, Kaigan, Minatoku, Tokyo, 105-0022, Japan
Phone: 81-3-5402-4511 Fax: 81-3-5402-4580 Internet: http://www.yaskawa.co.jp

YASKAWA ELETRICO DO BRASIL COMERCIO LTDA.
Avenida Fagundes Filho, 620 Bairro Saude Sao Paolo-SP, Brasil CEP: 04304-000
Phone: 55-11-5071-2552 Fax: 55-11-5581-8795 E-mail: yaskawabrasil@originet.com.br

YASKAWA ELECTRIC EUROPE GmbH
Am Kronberger Hang 2, 65824 Schwalbach, Germany
Phone: 49-6196-569-300 Fax: 49-6196-888-301 Internet: http://www.yaskawa.de

MOTOMAN ROBOTICS AB
Box 504 S38525, Torsas, Sweden
Phone: 46-486-48800 Fax: 46-486-41410

MOTOMAN ROBOTEC GmbH
Kammerfeldstraβe 1, 85391 Allershausen, Germany
Phone: 49-8166-900 Fax: 49-8166-9039

YASKAWA ELECTRIC UK LTD.
1 Hunt Hill Orchardton Woods Cumbernauld, G68 9LF, Scotland, United Kingdom
Phone: 44-12-3673-5000 Fax: 44-12-3645-8182

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.
Head Office: 151 Lorong Chuan, #04-01, New Tech Park Singapore 556741, SINGAPORE
Phone: 65-282-3003 Fax: 65-289-3003

TAIPEI OFFICE (AND YATEC ENGINEERING CORPORATION)
10F 146 Sung Chiang Road, Taipei, Taiwan
Phone: 886-2-2563-0010 Fax: 886-2-2567-4677

YASKAWA JASON (HK) COMPANY LIMITED
Rm. 2909-10, Hong Kong Plaza, 186-191 Connaught Road West, Hong Kong
Phone: 852-2803-2385 Fax: 852-2547-5773

BEIJING OFFICE
Room No. 301 Office Building of Beijing International Club,
21 Jianguomanwai Avenue, Beijing 100020, China
Phone: 86-10-6532-1850 Fax: 86-10-6532-1851

SHANGHAI OFFICE
27 Hui He Road Shanghai 200437 China
Phone: 86-21-6553-6600 Fax: 86-21-6531-4242

SHANGHAI YASKAWA-TONJI M & E CO., LTD.
27 Hui He Road Shanghai 200437 China
Phone: 86-21-6533-2828 Fax: 86-21-6553-6677

BEIJING YASKAWA BEIKE AUTOMATION ENGINEERING CO., LTD.
30 Xue Yuan Road, Haidian, Beijing 100083 China
Phone: 86-10-6232-9943 Fax: 86-10-6234-5002

SHOUGANG MOTOMAN ROBOT CO., LTD.
7, Yongchang-North Street, Beijing Economic & Technological Development Area,
Beijing 100076 China
Phone: 86-10-6788-0551 Fax: 86-10-6788-2878

Yaskawa Electric America, Inc., July 1999 YEA-SIAZ-C844-0.1 Printed In U.S.A.