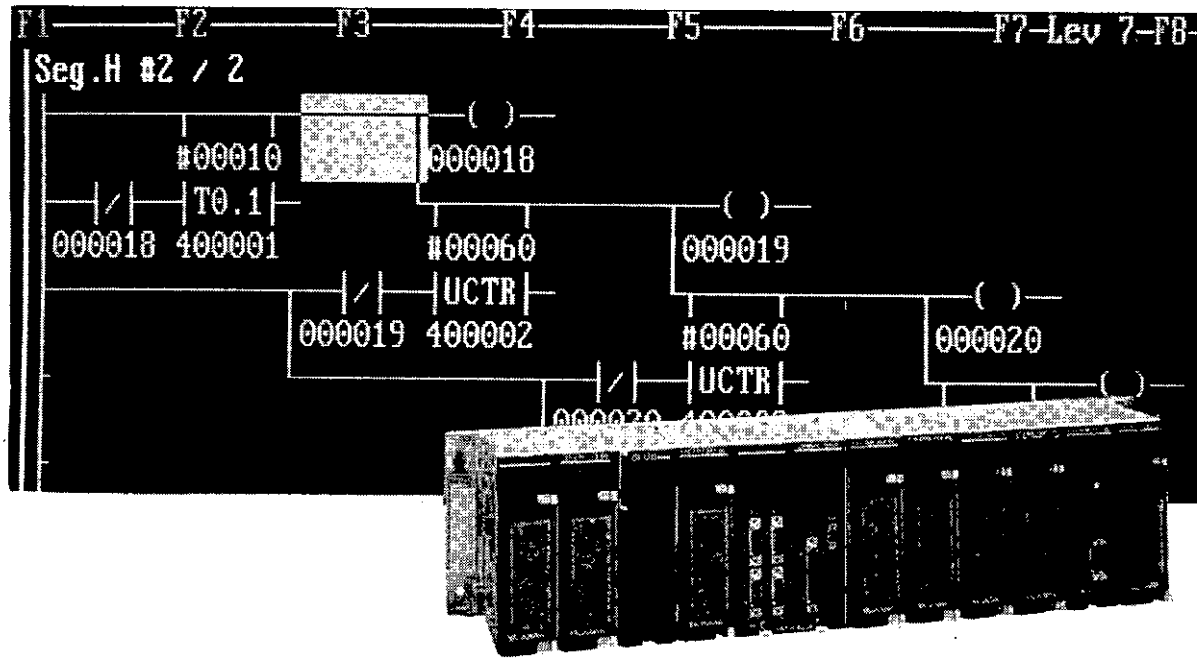
MEMOCON GL120, GL130

SOFTWARE USER'S MANUAL VOL.2



YASKAWA

MANUAL NO. SIEZ-C825-20.12B

Manual Contents

This manual describes the ladder logic programming instructions used to program the MEMOCON GL120 and GL130 Programmable Controllers (PLCs). Please read this manual carefully and be sure you understand the information provided before attempting to program a MEMOCON PLC.

Visual Aids

The following aids are used to indicate certain types of information for easier reference.

Indicates references for additional information.

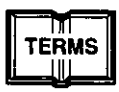
IMPORTANT — Indicates important information that should be memorized.

◀EXAMPLE▶ — Indicates application examples.

Indicates supplemental information.

SUMMARY — Indicates a summary of the important points of explanations.

Note — Indicates inputs, operations, and other information required for correct operation but that will not cause damage to the device.



Indicates definitions of terms used in the manual.

NOTICE

The following conventions are used to indicate precautions in this manual. Failure to heed precautions provided in this manual can result in injury to people or damage to the products.

⚠ **WARNING** — Indicates precautions that, if not heeded, could possibly result in loss of life or serious injury.

⚠ **Caution** — Indicates precautions that, if not heeded, could result in relatively serious or minor injury, damage to the product, or faulty operation.

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

Introduction and Precautions

This chapter introduces general information, including basic information precautions for the use of this manual and the software. **You must read this chapter before attempting to read the rest of the manual or using the product.**

I.1 Overview of Manual

- This manual describes the programming instruction used to create ladder programs for MEMOCON GL120 and GL130 Programmable Controllers. Expansion math, program control, communications, and motion control instructions, however, are described in other manuals.

- Read this manual carefully in order to use the instructions properly. Also, keep this manual in a safe place so that it can be used whenever necessary.

- Refer to the following related manuals.

	Manual Name	Manual Number	Content
CPU Module	MEMOCON GL120, GL130 Hardware User's Manual	SIEZ-C825-20.1	Describes the following for the GL120 and GL130: 1) System configuration 2) System components 3) Functions and specifications 4) Installation and wiring 5) Panel layout and hole dimensions 6) External dimensions
	MEMOCON GL120, GL130 Software User's Manual, Vol.1	SIEZ-C825-20.11	Describes the following for the GL120 and GL130. 1) Operating principles 2) I/O allocation 3) Overview of instructions 4) Instruction processing times
	MEMOCON GL120, GL130 Software User's Manual, Vol.3	SIEZ-C825-20.13	Describes expansion math instructions (floating point math instructions, etc.) used for the GL120 and GL130.
	MEMOCON GL120, GL130 Software User's Manual, Vol.4	SIEZ-C825-20.14	Describes process control instructions used for the GL120 and GL130.
I/O Modules	MEMOCON GL120, GL130 120-Series I/O Modules User's Manual	SIEZ-C825-20.22	Describes the functions, specifications, and usage of the 120-Series I/O Modules.
Special Purpose Modules	MEMOCON GL120, GL130 120-Series High-speed Counter Module User's Manual	SIEZ-C825-20.24	Describes the functions, specifications, and usage of the 120-Series High-speed Counter Module.
	MEMOCON GL120, GL130 120-Series Uniwire Interface Module User's Manual	SIEZ-C825-20.26	Describes the functions, specifications, and usage of the 120-Series Uniwire Interface Module.
Motion Modules	MEMOCON GL120, GL130 Motion Module MC10 User's Manual	SIEZ-C825-20.41	Describes the functions, specifications, and usage of the MC10 Motion Module (1 axis).
	MEMOCON GL120, GL130 Motion Module MC20 Hardware User's Manual	SIEZ-C825-20.51	Describes the functions, specifications, and usage of the MC20 Motion Module (4 axes).
	MEMOCON GL120, GL130 Motion Module MC20 Software User's Manual	SIEZ-C825-20.52	Describes the Motion Instructions and motion program language for the MC20 Motion Module (4 axes).

	Manual Name	Manual Number	Content
Man-machine Interface	MEMOCON GL120, GL130 Teach Pendant TB120 User's Manual	SIEZ-C825-60.3	Describes the functions, specifications, and usage of the TB120 Teach Pendant.
	MEMOCON GL120, GL130 MEMOSOFT for P120 Programming Panel User's Manual	SIEZ-C825-60.7	Describes the functions, specifications, and usage of the P120 Programming Panel with MEMOSOFT.
	MEMOCON GL120, GL130 MEMOSOFT for DOS User's Manual	SIEZ-C825-60.10	Describes the features and operating procedures of the DOS version of MEMOSOFT.
Communications Modules	MEMOCON GL120, GL130 PC Link Module User's Manual	SIEZ-C825-70.4	Describes the functions, specifications, FBUS communications instructions, and usage of the PC Link Module for the GL120 and GL130.
	MEMOCON GL120, GL130 MEMOBUS PLUS BASICS User's Manual	SIEZ-C825-70.5	Describes the functions, specifications, and usage of the MEMOBUS PLUS.
	MEMOCON GL120, GL130 Coaxial Remote I/O System User's Manual	SIEZ-C825-70.8	Describes the functions, specifications, and usage of the Coaxial Remote I/O System for the GL120 and GL130.
	MEMOCON GL120, GL130 MEMOBUS User's Manual	SIEZ-C825-70.13	Describes the functions, specifications, and usage of the MEMOBUS.
	MEMOCON GL120, GL130 COM Instructions User's Manual	SIEZ-C825-70.14	Describes the functions, specifications, and usage of the COM instructions. It also describes the specifications and usage of the MEMOBUS Module.

- Thoroughly check the specifications and conditions or restrictions of the product before using it.

I.2 Safety Precautions

- MEMOCON was not designed or manufactured for use in devices or systems that concern human lives. Users who intend to use the product described in this manual for special purposes such as devices or systems relating to transportation, medical, space aviation, atomic power control, or underwater use must contact Yaskawa Electric Corporation beforehand.

- This product has been manufactured under strict quality control guidelines. However, if this product is to be installed in any location in which a failure of MEMOCON involves a life and death situation or in a facility where failure may cause a serious accident, safety devices MUST be installed to minimize the likelihood of any accident.

- Any illustrations, photographs, or examples used in this manual are provided as examples only and may not apply to all product to which this manual is applicable.

- The products and specifications described in this manual or the content and presentation of the manual may be changed without notice to improve the product and/or the manual. A new version of the manual will be re-released under a revised document number when any changes are made.

- Contact your Yaskawa representative or a Yaskawa office listed on the back of this manual to order a new manual whenever this manual is damaged or lost. Please provide the document number listed on the front cover of this manual when ordering.

- Contact your Yaskawa representative or a Yaskawa office listed on the back of this manual to order new nameplates whenever a nameplate becomes worn or damaged.

- Yaskawa cannot make any quality guarantee for products which have been modified. Yaskawa assumes no responsibility for any injury or damage caused by a modified product.

I.3 Using this Manual

This manual is written for the following people:

- Workers responsible for designing GL120 or GL130 ladder programs.
- Workers responsible for testing GL120 or GL130 ladder programs.
- Workers responsible for debugging GL120 or GL130 ladder programs during trial operation.
- Workers responsible for maintaining GL120 or GL130 ladder programs.

- **Basic Terms**

  In this manual, the following terms have the meanings described below.

  - **PLC = Programmable (Logic) Controller**
  - **PP = Programming Panel**
  - **GL120, GL130 = MEMOCON GL120 and/or MEMOCON GL130 Programmable Controller**

- **Technical Terms**

  The bold technical terms in this manual are briefly explained in the **Glossary** provided at the bottom of the page. An example is shown below.

Glossary

The following types of terms are described.

- Specific sequence control terms required for explanation of functions.
- Terms that are specific to programmable controllers and electronic devices.

I.4 Reference Numbers

The types and ranges of reference numbers that can be used with each instruction are provided in this manual under the heading *Structural Elements*, as shown in the following example. These reference numbers are specified as follows:

1) The ranges listed in the tables are for initial values.

2) Two systems are used for reference numbers for coils, input relays, input registers, holding registers, and constant registers: Reference numbers beginning with numbers are called numeric reference numbers and those beginning with letters are called lettered reference numbers. In the *Structural Elements* tables, the lettered reference numbers are given in parentheses.

Structural Elements of REGISTER-TO-TABLE MOVE (R→T)

Element	Meaning	Possible Settings	
Top (S)	Source reference number	Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (P)	Pointer reference number	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 or R20001 to R21023
Bottom (Z)	Destination table size	Constant:	#00001 to #00999

Basic Instructions

1

This chapter describes the basic instructions used in programming.

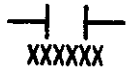
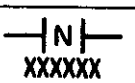
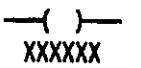
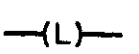
1.1 Relays

This section describes relay elements. It also provides precautionary information on designing and building relay circuits.

1.1.1 Relay Elements

The relay elements shown in the following table are combined to build relay circuits.

Table 1.1 Relay Elements

Name	Structure	Function	Reference No.
Normally Open (N.O.) Contact		Power is passed from left to right while the corresponding reference is ON.	Reference numbers for one of the following types of relay are specified for "XXXXXX". 1) Coils 2) Link coils 3) MC coils 4) MC control coils 5) Input relays 6) MC relays 7) MC control relays 8) M code relays
Normally Closed (N.C.) Contact		Power is passed from left to right while the corresponding reference is OFF.	
Positive Transitional Contacts		Power is passed from left to right for one scan each time the corresponding reference goes from OFF to ON.	
Negative Transitional Contacts		Power is passed from left to right for one scan each time the corresponding reference goes from ON to OFF.	
Horizontal Short		Shorts adjacent columns and passes power from left to right.	None (not necessary)
Vertical Short		Shorts adjacent rows and passes power from top to bottom or from bottom to top.	
Coil	1) Normal  2) Latched 	1) Output coils send the ON/OFF status to Digital Output Modules 2) Internal coils are used only to build logic in the ladder logic program. 3) Battery monitor coils are used to monitor the output voltage of the memory backup battery built into the CPU Module.	000001 to 008192 (O00001 to O08192)
Link Coil		Transfers ON/OFF data to/from another PLC in a PC link.	D10001 to D11024 D20001 to D21024
MC Coil		Outputs ON/OFF data to a Four-axis Motion Module (MC20).	Y10001 to Y10256 Y20001 to Y20256
MC Control Coil		Outputs overrides, MFIN signals, or other signals to a Four-axis Motion Module (MC20).	Q10001 to Q10160 Q20001 to Q20160

Note "XXXXXX" represents the reference number of a coil or relay.

1.1.2 Normally Open (N.O.) and Normally Closed (N.C.) Contacts

1. Function

1) Normally open (N.O.) contacts pass power from the left to the right when the corresponding reference is ON.

2) Normally closed (N.C.) contacts pass power from the left to the right when the corresponding reference is OFF.

2. Structure

N.O. contact: ⊣ ⊢ XXXXXX N.C. contact: ⊣/⊢ XXXXXX

1) ⊣ ⊢ is the symbol for a N.O. contact and ⊣/⊢ is the symbol for a N.C. contact.

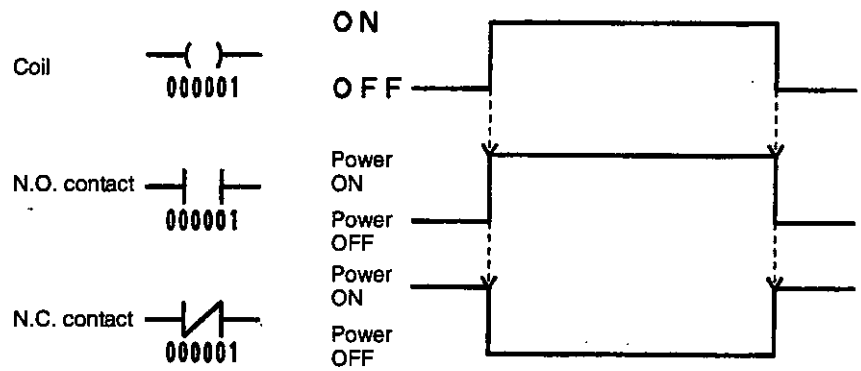
2) "XXXXXX" represents the reference number. The reference numbers listed in the following table can be specified for N.O. and N.C. contacts.

Table 1.2 Reference Numbers for N.O. and N.C. Contacts

Reference	Reference No.	Reference	Reference No.
Coil	000001 to 008192 (O00001 to O08192)	Input relay	100001 to 101024 (I00001 to I01024)
Link coil	D10001 to D11024 D20001 to D21024	MC relay	X10001 to X10256 X20001 to X20256
MC coil	Y10001 to Y10256 Y20001 to Y20256	MC control relay	P10001 to P10256 P20001 to P20256
MC control coil	Q10001 to Q10160 Q20001 to Q20160	M code relay	M10001 to M10096 M20001 to M20096

3. Operation

◀EXAMPLE▶ Examples of the operation of N.O. and N.C. contacts are shown in the following illustration.



1.1.3 Positive and Negative Transitional Contacts

1. Function

1) Positive transitional contacts pass power from left to right for one scan each time the corresponding reference goes from OFF to ON.

2) Negative transitional contacts pass power from left to right for one scan each time the corresponding reference goes from ON to OFF.

2. Structure

Positive transitional contact: —| P |— XXXXXX

Negative transitional contact: —| N |— XXXXXX

1) —| P |— is the symbol for a positive transitional contact and —| N |— is the symbol for a negative transitional contact.

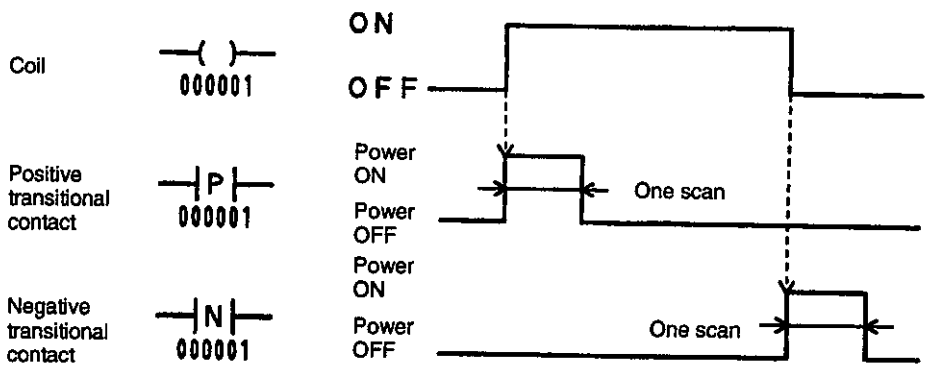
2) "XXXXXX" represents the reference number. The reference numbers listed in the following table can be specified for positive and negative transitional contacts.

Table 1.3 Reference Numbers for Transitional Contacts

Reference	Reference No.	Reference	Reference No.
Coil	000001 to 008192 (O00001 to O08192)	Input relay	100001 to 101024 (I00001 to I01024)
Link coil	D10001 to D11024 D20001 to D21024	MC relay	X10001 to X10256 X20001 to X20256
MC coil	Y10001 to Y10256 Y20001 to Y20256	MC control relay	P10001 to P10256 P20001 to P20256
MC control coil	Q10001 to Q10160 Q20001 to Q20160	M code relay	M10001 to M10096 M20001 to M20096

3. Operation

◀EXAMPLE▶ Examples of the operation of positive and negative contacts are shown in the following illustration.



1.1.4 Horizontal and Vertical Shorts

1. Function

1) Horizontal shorts connect adjacent columns and pass power from left to right.

2) Vertical shorts connect adjacent rows and pass power from top to bottom or from bottom to top.

2. Structure

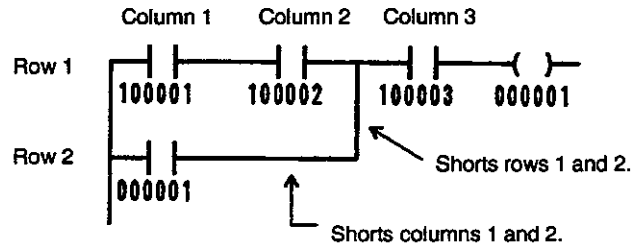
1) Horizontal short: ———

   Vertical short: |

2) Reference numbers are not required.

3. Application Example

◀EXAMPLE▶ An example of the operation of horizontal and vertical shorts is shown in the following illustration.



1.1.5 Coils

Output coils, internal coils, and battery monitor coils are collectively called coils.

1. Output Coils

A. Function

1) Coils that can be used to output ON/OFF status to Digital Output Modules are called output coils.

2) The desired coils can be specified as output coils when I/O is allocated.

3) There is a limit, however, to number of output coils that can be defined. The limit depends on the model of CPU Module that is being used, as follows:

   • CPU20: Number of input relays + number of output coils $\leq$ 1,024

   • CPU30: Number of input relays + number of output coils $\leq$ 4,096

4) There are two types of output coils that vary in the way they operate during the power-up sequence performed when power is turned ON, as described below. The operation of these two types of output coils is the same during the program scan.

- **Normal Output Coils**
  During the power-up sequence, normal output coils are always turned OFF.

- **Latched Output Coils**
  During the power-up sequence, latched output coils always maintain the status they have before power was turned OFF.

B. Structure

Normal output coil: —( )— XXXXXX Latched output coil: —(L)— XXXXXX

1) —( )— is the symbol for a normal output coil and —(L)— is the symbol for a latched output coil.

2) "XXXXXX" represents the reference number. Any reference number between 000001 to 008192 (O00001 to O08192) that is specified as an output coil in I/O allocation can be used as the reference number for an output coil.

C. Operation of Enabled Output Coils

1) Operation on Power Application

a) Normal output coils are turned OFF during the power-up sequence performed when power is turned ON and the OFF status is output to the Digital Output Module.

b) Latched output coils are set to the status they had before power was turned OFF during the power-up sequence performed when power is turned ON and the previous status is output to the Digital Output Module.

2) Operation During the Scan Cycle

There is no difference in the operation of normal and latched output coils during the scan. Both types of output coils operate as follows:

a) Output coils are ON while power is being supplied and the ON status is output to the Digital Output Unit. When an output coil goes from OFF to ON, contacts with the same reference number as the output coil behave as follows:

- N.O. contacts pass power from left to right.

- N.C. contact interrupt power flow.

- Positive transitional contacts pass power from left to right only for the scan in which the output coil goes from OFF to ON.

- Negative transitional contacts will interrupt power flow (i.e., will normally not change)

b) Output coils are OFF while power is not being supplied and the OFF status is output to the Digital Output Unit. When an output coil goes from ON to OFF, contacts with the same reference number as the output coil behave as follows:

- N.O. contact interrupt power flow.

- N.C. contacts pass power from left to right.

- Positive transitional contacts will interrupt power flow (i.e., will normally not change)

- Negative transitional contacts pass power from left to right only for the scan in which the output coil goes from ON to OFF.

D. Operation of Disabled Output Coils

1) Operation on Power Application

There is no difference in the operation of disabled normal and latched output coils. When disabled, output coils return to the status they had before the scan operation was interrupted and that status is output to the Digital Output Module.

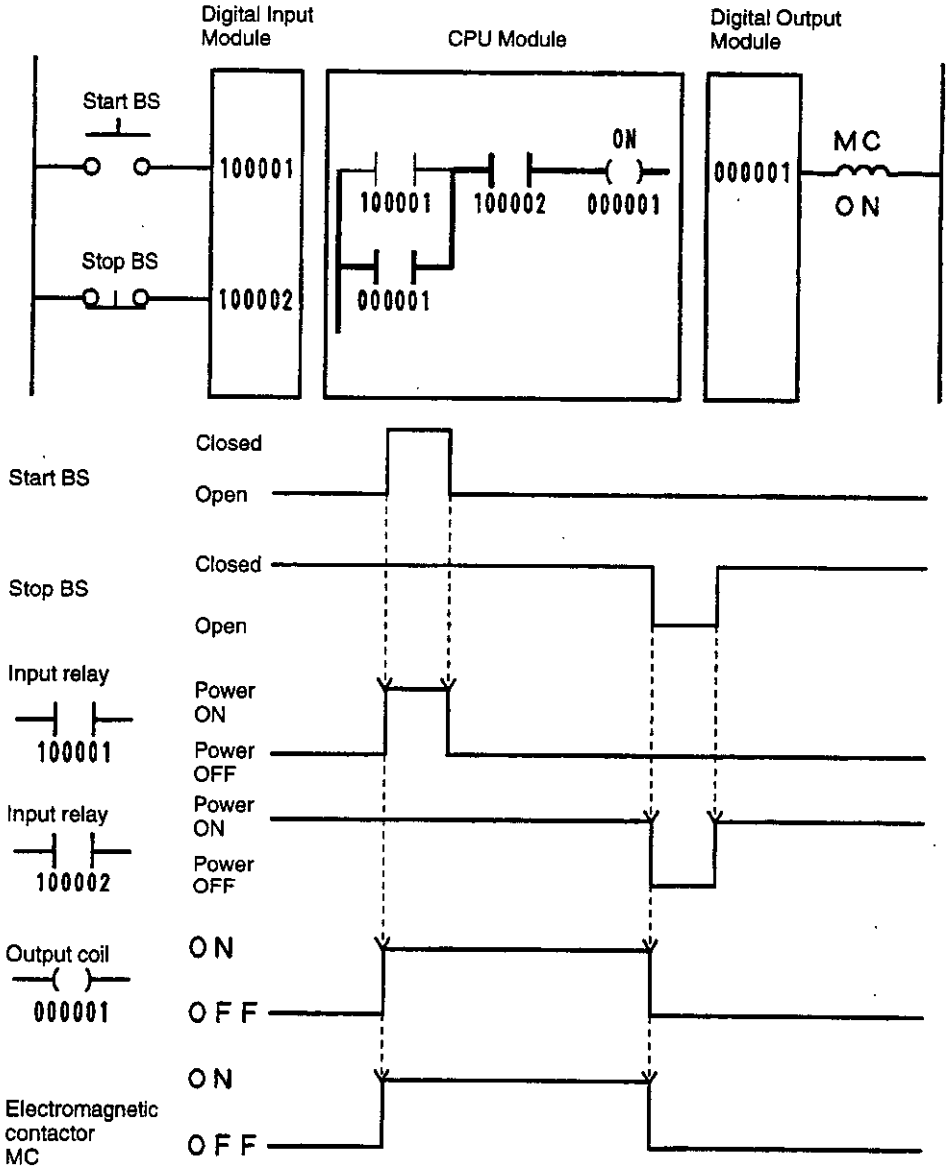
2) Operation During the Scan Cycle

During the scan, disabled output coils are turned ON if forced ON from the MEMOSOFT, are turned OFF if forced OFF from the MEMOSOFT, and the status is output to the Digital Output Module. The operation of coils in response to output coils status is the same as for enabled output coils.

E. Output Coil Application Examples

◀EXAMPLE▶ **Example 1: Using Normal Output Coils**

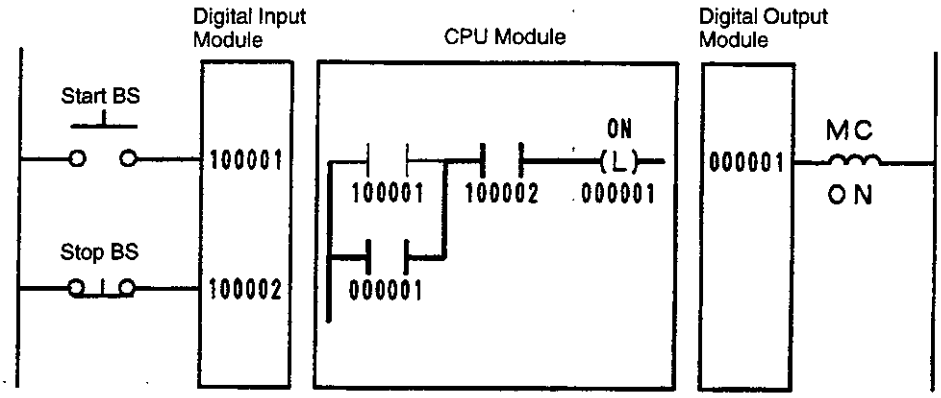
1) When start BS is pressed, coil 000001 turns ON, turning ON the electromagnetic contactor MC. The ON status is maintained even if the start BS is released because of the self-holding action of a N.O. contact for coil 000001. The electromagnetic contactor MC will turn OFF when the stop BS is pressed.



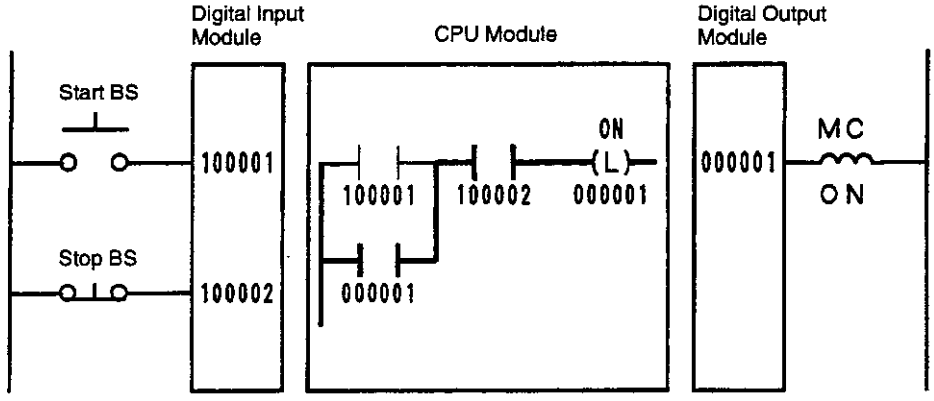
2) If power is interrupted and then restarted with coil 000001 in the above state, coil 000001 will be turned OFF in the power-up sequence, releasing the self-holding circuit. The start BS will have to be pressed again to turn ON coil 000001 again.

◀EXAMPLE▶ **Example 2: Using Latched Output Coils**

1) Assume that latched output coil 000001 is in the state shown in the following illustration before power is interrupted.



2) When power is turned back ON, coil 000001 will be restored to ON in the power-up sequence, as shown in the following illustration, and start BS will not have to be pressed again to turn ON coil 000001.



2. Internal Coils

A. Function

1) Internal coils are used to build logic in the ladder logic program.

2) Internal coils cannot be used to output ON/OFF status to Digital Output Modules.

3) All coils that are not specified as output coils are internal coils.

4) There are two types of internal coils: normal and latched.

B. Structure

Normal internal coil: —( )— XXXXXX Latched internal coil: —(L)— XXXXXX

1) —( )— is the symbol for a normal internal coil and —(L)— is the symbol for a latched internal coil.

2) "XXXXXX" represents the reference number. Any reference number between 000001 to 008192 (O00001 to O08192) that is not specified as an output coil in I/O allocation can be used as the reference number for an internal coil.

C. Operation of Enabled Internal Coils

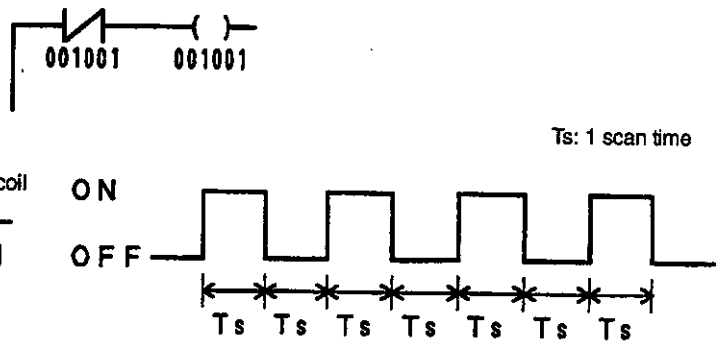
The operation of enabled internal coils is the same as that of output coils, except that the ON/OFF status is not output to Digital Output Modules.

D. Operation of Disabled Internal Coils

The operation of disabled internal coils is the same as that of output coils, except that the ON/OFF status is not output to Digital Output Modules.

E. Using Internal Coils

◀EXAMPLE▶ In the following circuit, coil 001001 will change continuously between ON and OFF each scan.



3. Battery Monitor Coil

A. Function

1) A battery monitor coil is used to monitor the output voltage of the memory backup battery built into the CPU Module.

2) The battery monitor coil can also be specified as an output coil to directly output the ON/OFF status to a Digital Output Module.

B. Structure

The battery monitor coil is initially set to $\dashv(\ )\vdash_{008192}$. This coil cannot be used within the network; use the reference number in contacts.

C. Operation of Enabled Battery Monitor Coil

1) The output voltage of the memory backup battery in the CPU Module will be monitored and the battery monitor coil will behave as described below both during the power-up sequence when power is turned on and after moving to the scan cycle.

   a) The battery monitor coil will be ON as long as the output voltage is within the proper range.

   b) The battery monitor coil will turn OFF if the output voltage is not within the proper range.

2) The operation of contacts for the ON/OFF status of the battery monitor coil is the same as the operation of output coils or internal coils during the scan cycle.

D. Operation of Disabled Battery Monitor Coil

1) Operation on Power Application

The battery monitor coil is set to the status that it had before the scan operation stopped.

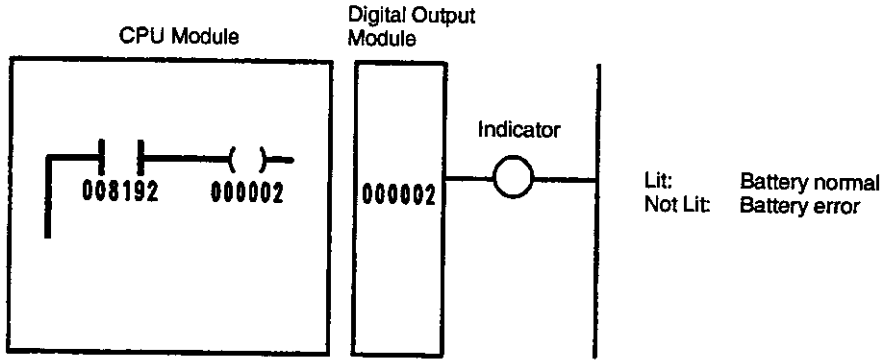
2) Operation During the Scan Cycle

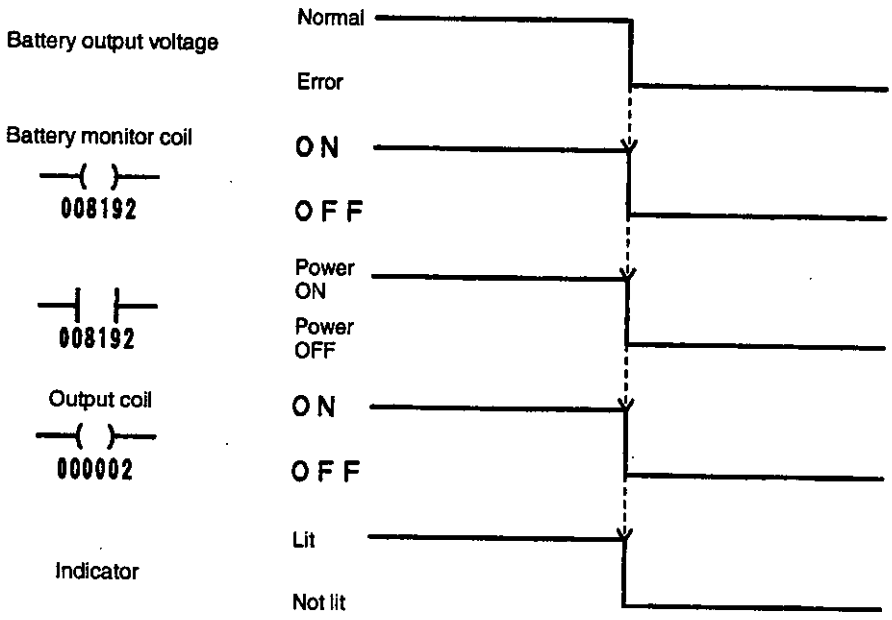
During the scan, disabled output coils are turned ON if forced ON from the MEMOSOFT, are turned OFF if forced OFF from the MEMOSOFT, and the status is output to the Digital Output Module. The operation of coils in response to output coils status is the same as for enabled output coils.

E. Using the Battery Monitor Coil

◀EXAMPLE▶

As long as the battery output voltage is within the proper range, the battery monitor coil 008192 will remain ON and the indicator lamp will remain lit.





Note The battery monitor coil can also be specified as an output coil to directly output the ON/OFF status to a Digital Output Module.

1.1.6 Link Coils

The use of link coils is enabled by setting the number of PC Link Modules to either 1 or 2 on the system configuration table display of the MEMOSOFT.

1. Function

The CPU Module uses link coils and corresponding contacts to send and receive ON/OFF data with other PLCs on the PC Link.

2. Structure

Normal link coil: —( )— XXXXXX Latched link coil: —(L)— XXXXXX

1) —( )— is the symbol for a normal link coil and —(L)— is the symbol for a latched link coil.

2) "XXXXXX" represents the reference number. The reference numbers listed in the following table can be used.

Table 1.4 Reference Numbers for Link Coils

Number of PC Link Modules	Channel No.	Reference No.
1	1	D10001 to D11024
2	1	D10001 to D11024
	2	D20001 to D21024

3. Operation

Refer to the following manual for details on the operation of link coils.

MEMOCON GL120, GL130 PC Link Module User's Manual (SIEZ-C825-70.4)

1.1.7 MC Coils and MC Control Coils

The use of MC coils and MC control coils is enabled by setting the number of MC20 Modules to either 1 or 2 on the system configuration table display of the MEMOSOFT.

1. Function

1) The CPU Module uses MC coils to output ON/OFF data to 4-axis Motion Modules (MC20).

2) The CPU Module uses MC control coils to output overrides, MFIN signals, and other signals to 4-axis Motion Modules (MC20).

2. Structure

Normal MC or MC control coil: —( )— XXXXXX

Latched MC or MC control coil: —(L)— XXXXXX

1) —( )— is the symbol for a normal MC or MC control coil and —(L)— is the symbol for a latched MC or MC control coil.

2) "XXXXXX" represents the reference number. The reference numbers listed in the following table can be used.

Table 1.5 Reference Numbers for MC and MC Control Coils

Number of MC20 Modules	Module No.	Reference No.	
		MC Coil	MC Control Coil
1	1	Y10001 to Y10256	Q10001 to Q10160
2	1	Y10001 to Y10256	Q10001 to Q10160
	2	Y20001 to Y20256	Q20001 to Q20160

3. Operation

Refer to the following manual for details on the operation of MC and MC control coils.

MEMOCON GL120, GL130 Motion Module MC20 Software User's Manual (SIEZ-C825-20.52)

1.1.8 Relay Circuit Design Example

An example of a relay circuit design for the GL120 or GL130 is provided next.

◀EXAMPLE▶ 1) We will consider the GL120/GL130 relay circuit design equivalent to the following contact relay circuit.

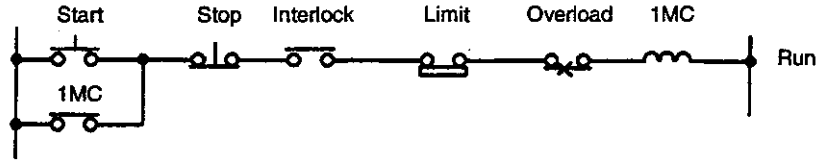


Figure 1.1 Contact Relay Circuit

2) Reference numbers for I/O signals are allocated as shown in *Figure 1.2.*

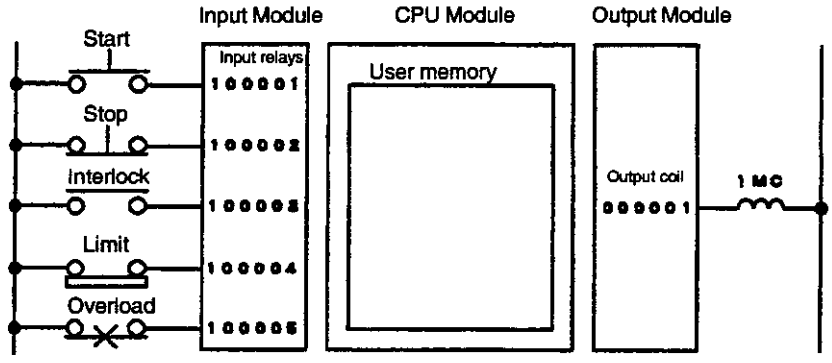


Figure 1.2 Allocation of Reference Numbers to I/O Signals

3) The relay circuit that is equivalent to the above contact relay circuit is shown below. This circuit would be input in the user memory of the GL120/GL130 CPU Module.

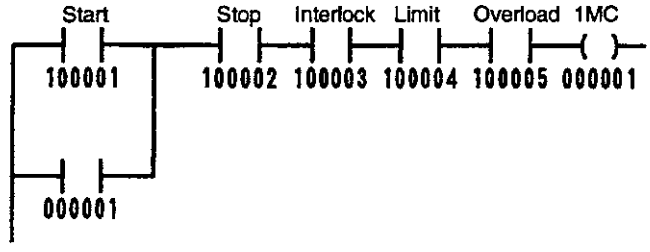


Figure 1.3 Equivalent GL120/GL130 Relay Circuit

Note In this example, normally ON states have been used to input the stop, interlock, limit, and overload signals to the Input Module for safety reasons. In actual system applications, system design specifications would need to be considered to determine if N.C. or N.O. contacts should be used.

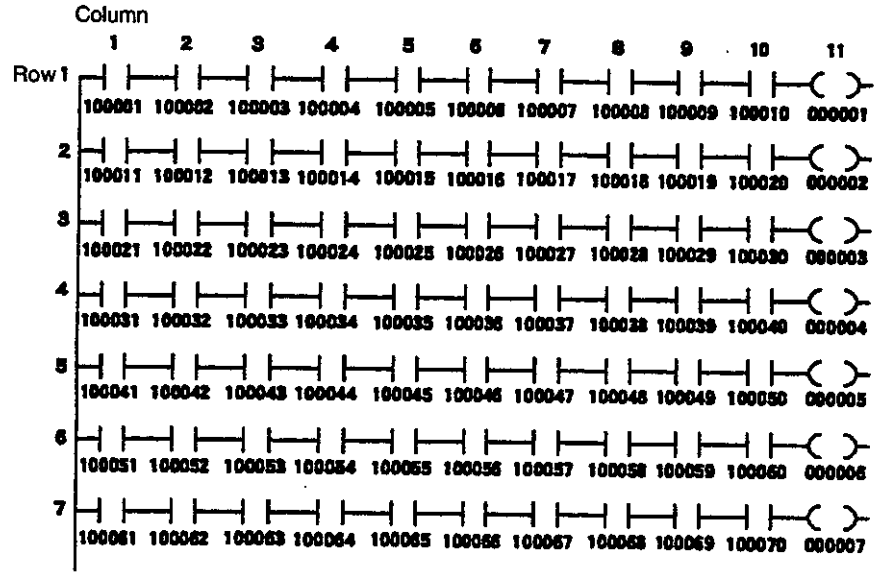
1.1.9 Building Relay Circuits

1) Storage Locations on Networks

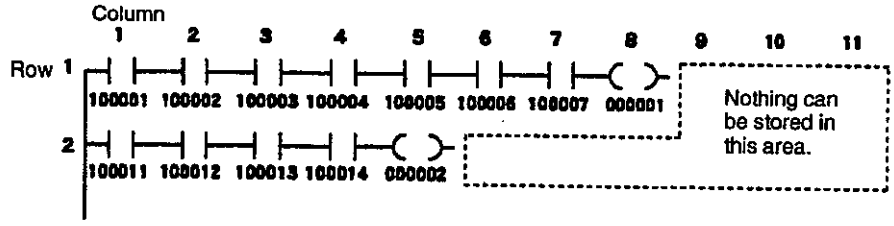
a) Contacts and Horizontal Shorts

Contacts (including N.O., N.C., positive transitional, and negative transitions contacts) and horizontal shorts can be stored horizontally anywhere on a 7-row by 10-column matrix (rows 1 through 7 and columns 1 through 10). Contacts cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

Example 1
Up to 70 contacts and 7 coils can be stored in one network.



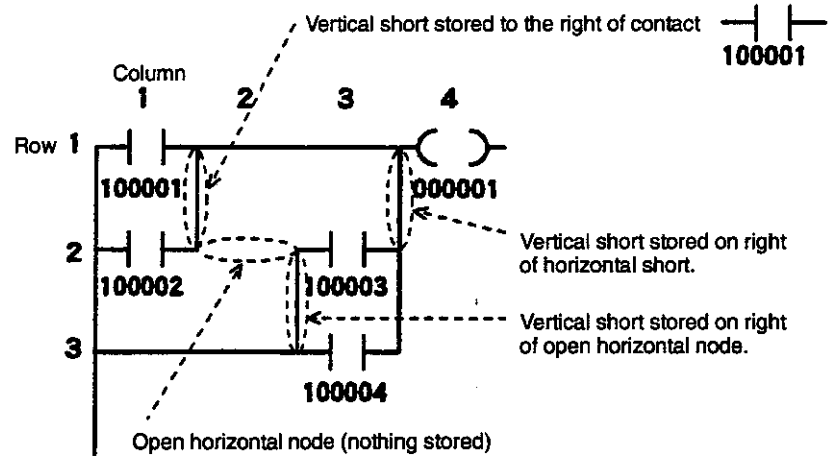
Example 2
Nothing can be stored to the right of coils.



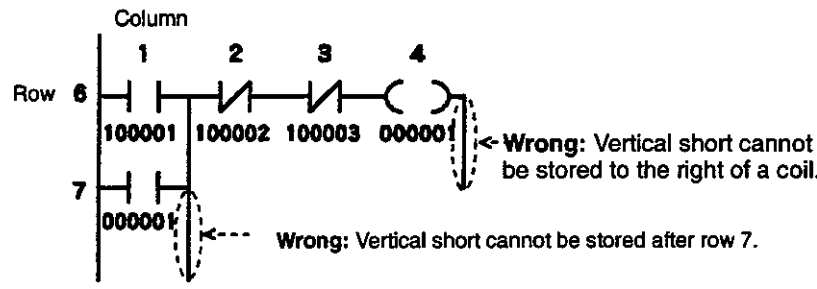
b) Vertical Shorts

Vertical shorts are stored vertically between one row and the row below it. Vertical shorts cannot, however, be stored after row 7, in column 11, or to the right of coils.

Example 1: Correct Application



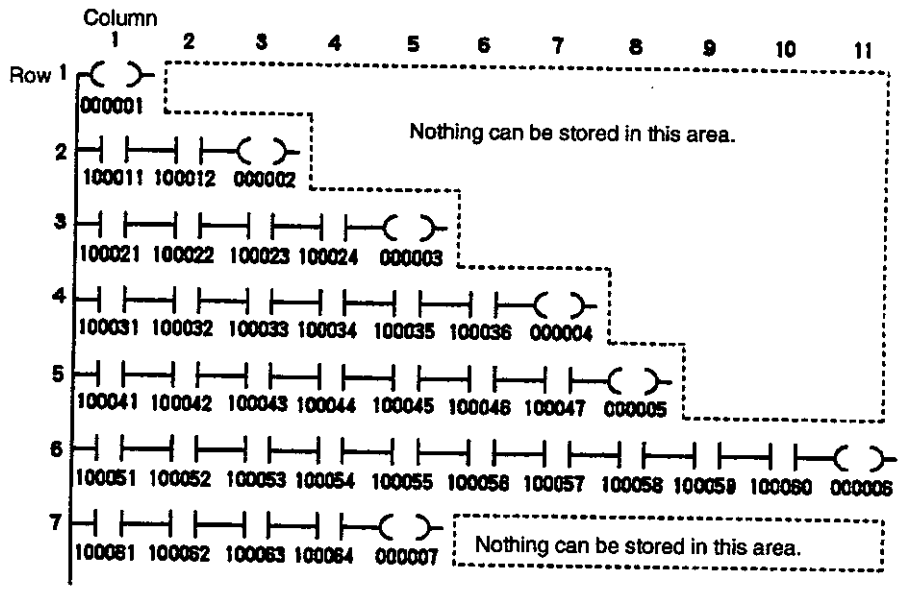
Example 2: Incorrect Application



c) Coils

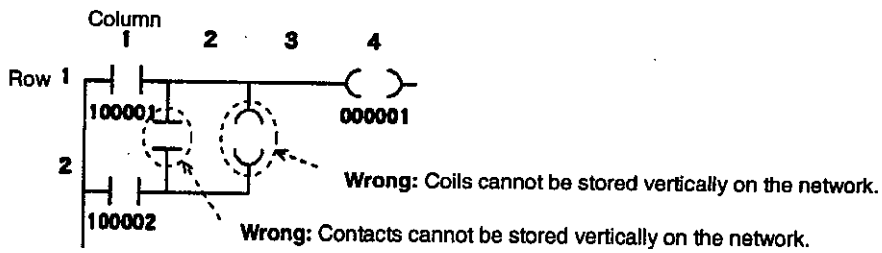
Coils (including output coils, internal coils, link coils, MC coils, and MC control coils) can be stored horizontally anywhere on a 7-row by 11-column matrix (rows 1 through 7 and columns 1 through 11). Only one coil, however, can be stored on each row, and nothing can be stored to the right of a coil.

Example
Up to 7 coils can be stored in one network.



d) Contacts and coils are stored horizontally on a network. They cannot be stored vertically, as illustrated below.

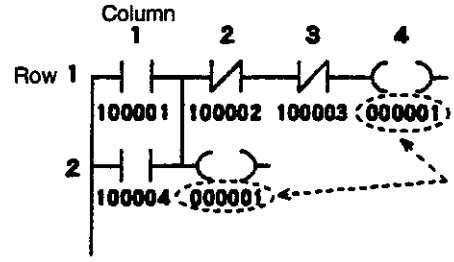
Example: Incorrect Application



2) Using Reference Numbers

a) The same reference number cannot be used for more than one coil, i.e., duplicate coils cannot be used.

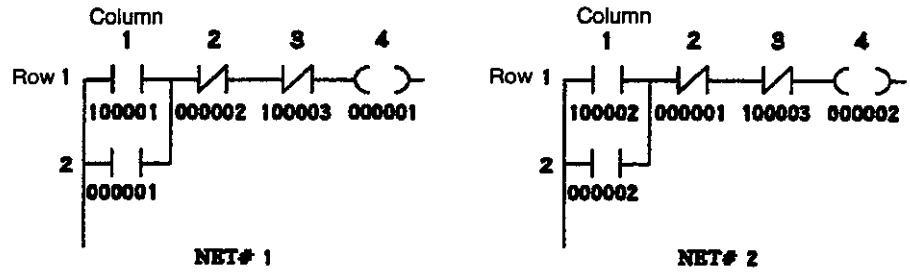
Example: Incorrect Application



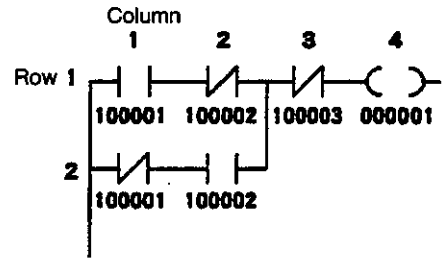
Wrong: The same reference number cannot be used for more than one coil.

b) The same reference number can be used as many times as required for different contacts.

Example 1

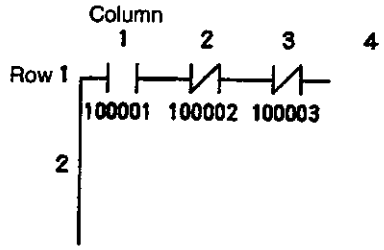


Example 2



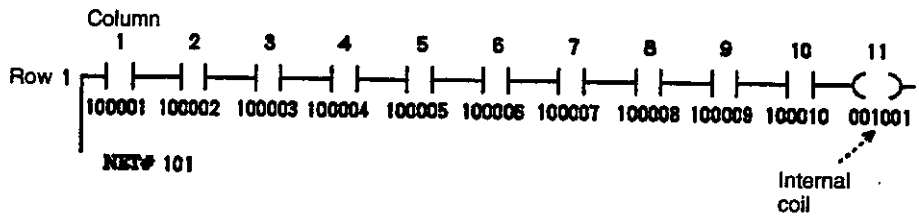
3) Although relay circuits without coils are not technically mistakes, they are meaningless.

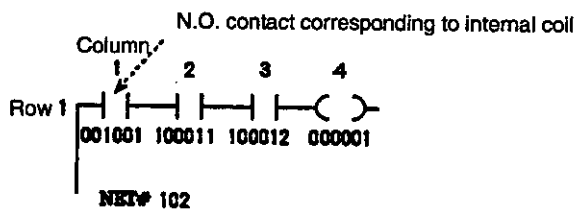
Example



4) An internal coil and a N.O. contact with the same reference number can be used to connect more than 10 contacts in series to the same coil. Examples 1 and 2 show how to connect N.O. contacts for input relays 100001 to 100012 to coil 000001.

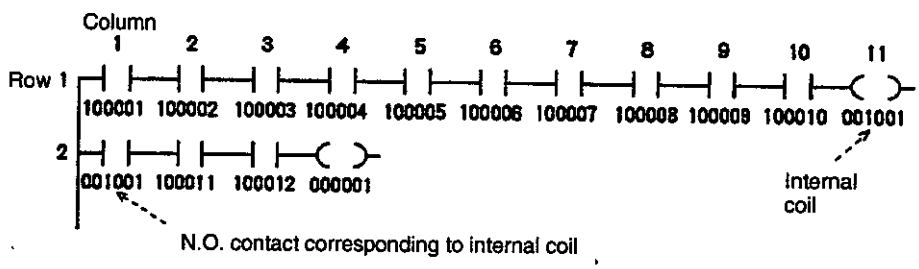
Example 1: Programming in Two Different Networks





Coil 001001 will turn ON in the same scan that input relays 100001 to 100012 turn ON.

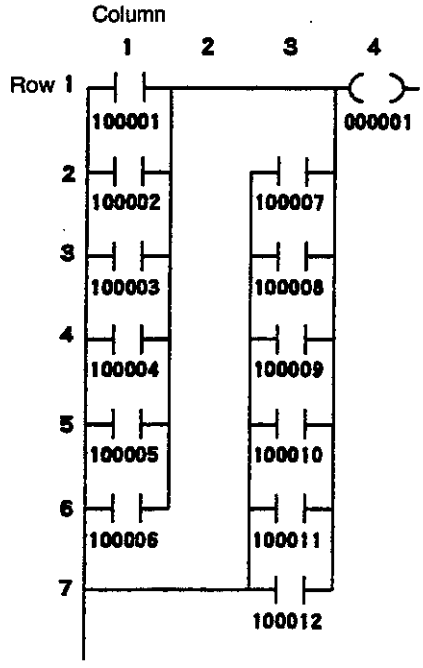
Example 2: Programming in the Same Network



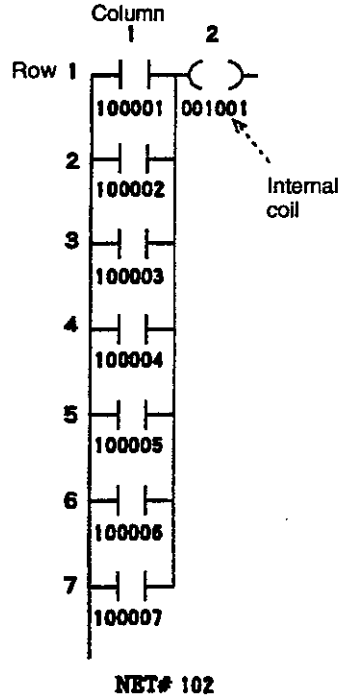
Coil 001001 will turn ON in the same scan that input relays 100001 to 100012 turn ON, but coil 000001 will not turn ON until the next scan and will turn ON then only if input relays 100011 to 100012 are still ON.

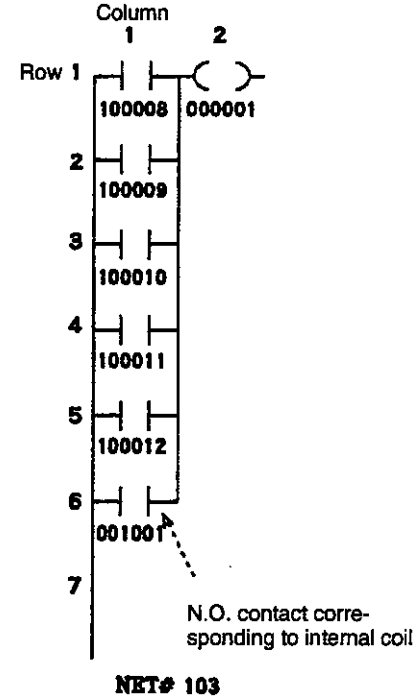
5) Examples 1 and 2 show how to connect more than 7 contact in parallel to the same coil. Both of these examples show how to connect N.O. contacts for input relays 100001 to 100012 to coil 000001.

Example 1: Programming in One Network (No Internal Coil Necessary)



Example 2: Programming in Two Different Networks (Using Internal Coil)



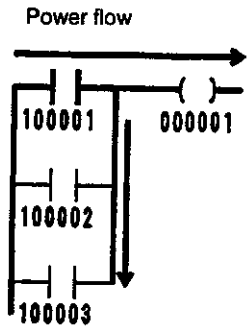


6) Power flows within a network according to the following rules.

a) Power always flows to the right from the power rail and never flows backward, i.e., never flows from the right to the left.

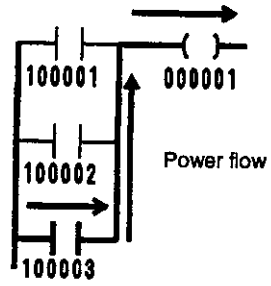
b) Power flows both from the top to the bottom and from the bottom to the top within a column.

Example 1



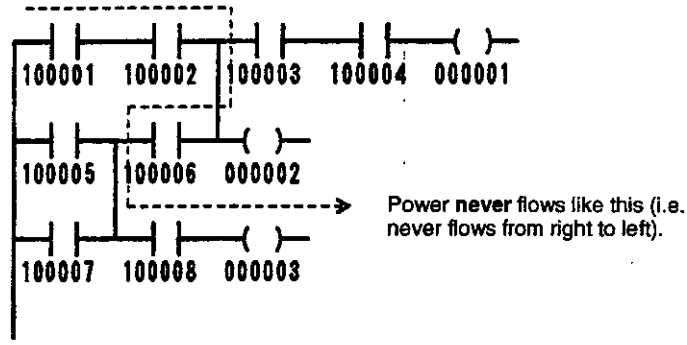
Power will flow as illustrated by the arrows in the above diagram whenever input relay 100001 is ON, causing coil 000001 to turn ON.

Example 2

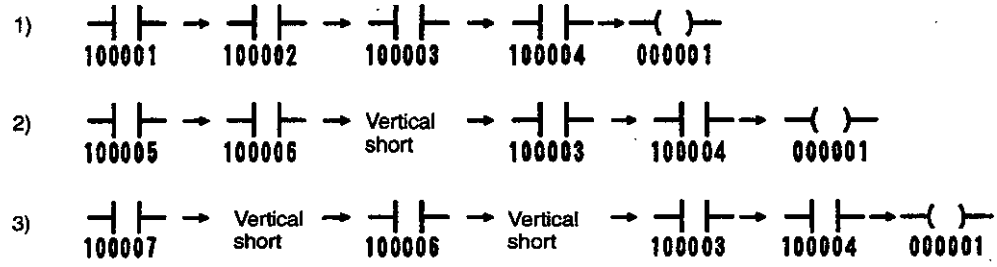


Power will flow as illustrated by the arrows in the above diagram whenever input relay 100003 is ON, causing coil 000001 to turn ON.

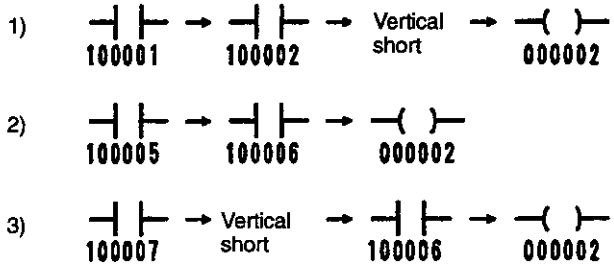
Example 3



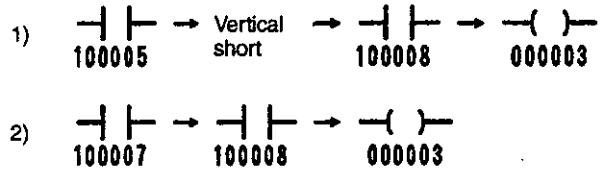
(1) Possible Power Flows to Turn ON Coil 000001



(2) Possible Power Flows to Turn ON Coil 000002



(3) Possible Power Flows to Turn ON Coil 000003



(4) Coil 000003 will never turn ON for the power flow shown by the arrow with the dotted lines. There are thus no worries about reverse power flow.

1.2 Timers

This section describes the timer instructions available for programming, including their functions, structure, operation, and application. Building timer circuits is also discussed and related precautions are provided.

1.2.1 Timer Instructions

The following table list the four timer instructions. As many of the following instructions can be used as long as the user memory, holding register, or link register capacity is not exceeded.

Table 1.6 Timer Instructions

Instruction Name	Symbol	Timing Unit (*1)	Timing Range (*2)
1-SECOND TIMER	T1.0	1 s	1 to 65,535 s
0.1-SECOND TIMER	T0.1	0.1 s	0.1 to 6,553.5 s
0.01-SECOND TIMER	T.01	0.01 s	0.01 to 655.35 s
0.001-SECOND TIMER	T1MS	0.001 s	0.001 to 65.535 s

*1: The timing unit is the unit used by the timer to measure time. For example, 1-SECOND TIMER measures time in 1-s intervals.

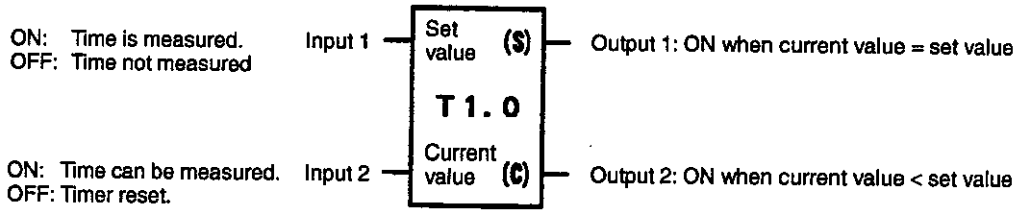
*2: The timing range is the range in which the timer can measure time. For example, 1-SECOND TIMER measures time between 1 and 65,535 s.

1.2.2 1-SECOND TIMER (T1.0)

1. Function

1-SECOND TIMER is used to measure time in 1-s intervals between 1 and 65,535 s.

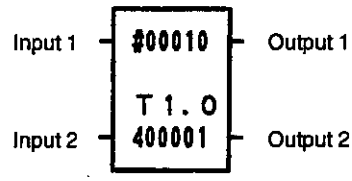
2. Structure



1) T1.0 is the symbol for 1-SECOND TIMER.

2) T1.0 requires two elements, one top element and one bottom element, located vertically on the network. *Table 1.7* lists the register reference numbers and constants that can be specified.

Example



#00010: Set value (10)
400001: Reference number of holding register to store the current value.

Table 1.7 Structural Elements of T1.0

Element	Meaning	Possible Settings	
Top (S)	The value of the constant or the contents of the register with the specified reference number will be used as the set value (S) of the timer. The value of S must be between 0 and 65,535.	Constant:	#00000 to #65535
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 R20001 to R21024
Bottom (C)	The current value of the timer (C) will be stored in the specified register. The value of C will be between 0 and the set value.	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 R20001 to R21024

3. Operation

1) If input 2 is ON, the following timing operation will be performed by T1.0 while input 1 is ON.

   a) The current value will be incremented by 1 every second. As long as the current value is less than the set value, output 1 will remain OFF and output 2 will remain ON.

   b) When the current value reaches the set value, the current value will no longer be incremented, output 1 will turn ON, and output 2 will turn OFF.

2) If input 1 turns ON and OFF repeatedly while input 2 is ON, the time that input 1 is ON will be measured and the operation described for item 1) will be performed.

3) If input 2 is OFF, timing will not be performed regardless of the status of input 1, the current value will be set to 0, output 1 will turn OFF, and output 2 will turn ON.

4) The following table summarizes the operation of T1.0.

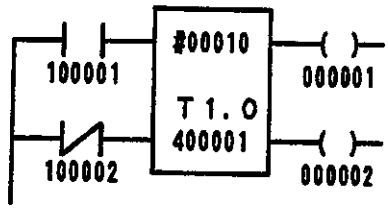
Table 1.8 Operation of T1.0

Inputs		Timer Status		Current Value	Outputs	
1	2				1	2
Any	OFF	Reset		0	OFF	**ON**
ON	**ON**	Running	Current value < Set value	Incrementing	OFF	**ON**
			Current value = Set value	Set value	**ON**	OFF
OFF	**ON**	Stopped	Current value < Set value	Not changed	OFF	**ON**
			Current value = Set value	Set value	**ON**	OFF

◀EXAMPLE▶

4. Application Example

1) Ladder Programming



Set value: 10
Timing range: 1 to 10 s

2) Operation

a) Timer operation is enabled when input relay 100002 is OFF.

b) The timer will start when input 100001 turns ON while input relay 100002 is OFF and the current value (contents of holding register 400001) will be incremented 1 each second. Coil 000001 will be OFF and coil 000002 will be ON.

c) When the current value reaches the set value (10), the timer will stop and the current value will no longer be incremented. Coil 000001 will turn ON and coil 000002 will turn OFF.

d) If input relay 100001 turns ON and OFF repeatedly while input relay 100002 is OFF, the time that input relay 100001 is ON will be measured and the operation described for items b) and c) will be performed.

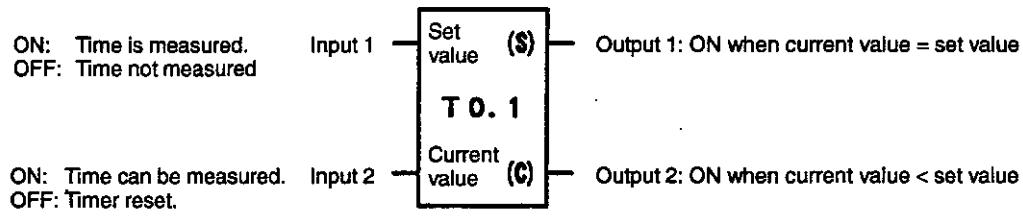
e) If input relay 100002 is ON, timing will not be performed regardless of the status of input relay 100001, the current value will be set to 0, coil 000001 will turn OFF, and coil 000002 will turn ON.

1.2.3 0.1-SECOND TIMER (T0.1)

1. Function

0.1-SECOND TIMER is used to measure time in 0.1-s intervals between 0.1 and 6,553.5 s.

2. Structure



1) T0.1 is the symbol for 0.1-SECOND TIMER.

2) T1.0 requires two elements, one top element and one bottom element, located vertically on the network. *Table 1.9* lists the register reference numbers and constants that can be specified.

Example

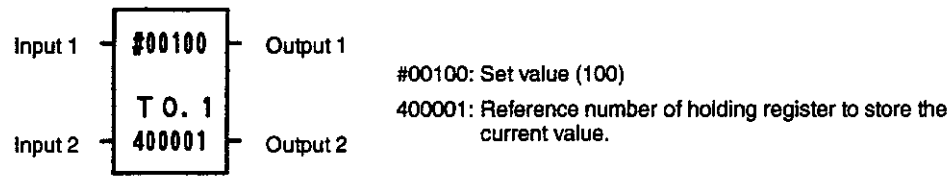


Table 1.9 Structural Elements of T0.1

Element	Meaning	Possible Settings	
Top (S)	The value of the constant or the contents of the register with the specified reference number will be used as the set value (S) of the timer. The value of S must be between 0 and 65,535.	Constant:	#00000 to #65535
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 R20001 to R21024
Bottom (C)	The current value of the timer (C) will be stored in the specified register. The value of C will be between 0 and the set value.	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 R20001 to R21024

3. Operation

1) If input 2 is ON, the following timing operation will be performed by T0.1 while input 1 is ON.

   a) The current value will be incremented by 1 every 0.1 second. As long as the current value is less than the set value, output 1 will remain OFF and output 2 will remain ON.

   b) When the current value reaches the set value, the current value will no longer be incremented, output 1 will turn ON, and output 2 will turn OFF.

2) If input 1 turns ON and OFF repeatedly while input 2 is ON, the time that input 1 is ON will be measured and the operation described for item 1) will be performed.

3) If input 2 is OFF, timing will not be performed regardless of the status of input 1, the current value will be set to 0, output 1 will turn OFF, and output 2 will turn ON.

4) The following table summarizes the operation of T0.1.

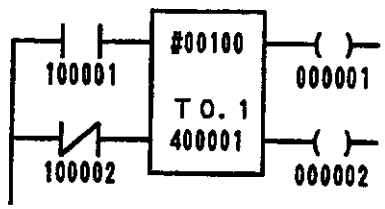
Table 1.10 Operation of T0.1

Inputs		Timer Status		Current Value	Outputs	
1	2				1	2
Any	OFF	Reset		0	OFF	**ON**
ON	**ON**	Running	Current value < Set value	Incrementing	OFF	**ON**
			Current value = Set value	Set value	**ON**	OFF
OFF	**ON**	Stopped	Current value < Set value	Not changed	OFF	**ON**
			Current value = Set value	Set value	**ON**	OFF

◀EXAMPLE▶

4. Application Example

1) Ladder Programming



Set value: 100
Timing range: 0.1 to 10.0 s

2) Operation

a) Timer operation is enabled when input relay 100002 is OFF.

b) The timer will start when input 100001 turns ON while input relay 100002 is OFF and the current value (contents of holding register 400001) will be incremented 1 each 0.1 second. Coil 000001 will be OFF and coil 000002 will be ON.

c) When the current value reaches the set value (100), the timer will stop and the current value will no longer be incremented. Coil 000001 will turn ON and coil 000002 will turn OFF.

d) If input relay 100001 turns ON and OFF repeatedly while input relay 100002 is OFF, the time that input relay 100001 is ON will be measured and the operation described for items b) and c) will be performed.

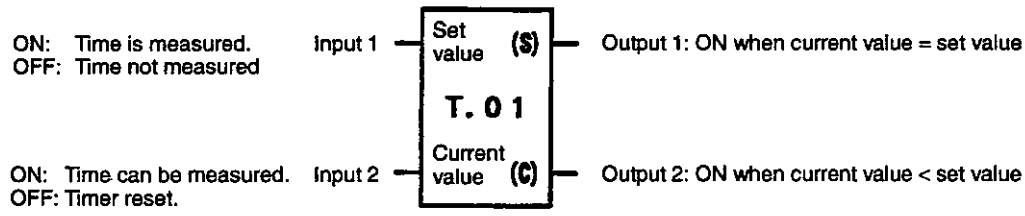
e) If input relay 100002 is ON, timing will not be performed regardless of the status of input relay 100001, the current value will be set to 0, coil 000001 will turn OFF, and coil 000002 will turn ON.

1.2.4 0.01-SECOND TIMER (T.01)

1. Function

0.01-SECOND TIMER is used to measure time in 0.01-s intervals between 0.01 and 655.35 s.

2. Structure



1) T.01 is the symbol for 0.01-SECOND TIMER.

2) T.01 requires two elements, one top element and one bottom element, located vertically on the network. *Table 1.11* lists the register reference numbers and constants that can be specified.

Example

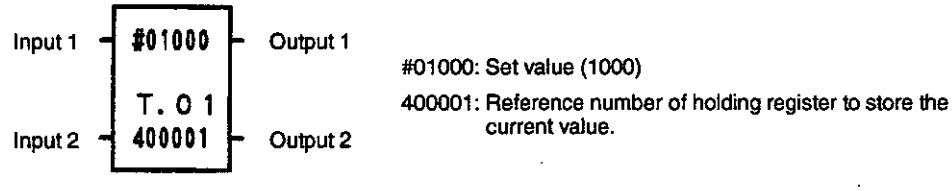


Table 1.11 Structural Elements of T.01

Element	Meaning	Possible Settings	
Top (S)	The value of the constant or the contents of the register with the specified reference number will be used as the set value (S) of the timer. The value of S must be between 0 and 65,535.	Constant: Input register: Holding register: Constant register: Link register:	#00000 to #65535 300001 to 300512 (Z00001 to Z00512) 400001 to 409999 (W00001 to W09999) 700001 to 704096 (K00001 to K04096) R10001 to R11024 R20001 to R21024
Bottom (C)	The current value of the timer (C) will be stored in the specified register. The value of C will be between 0 and the set value.	Holding register: Link register:	400001 to 409999 (W00001 to W09999) R10001 to R11024 R20001 to R21024

3. Operation

1) If input 2 is ON, the following timing operation will be performed by T.01 while input 1 is ON.

   a) The current value will be incremented by 1 every 0.01 second. As long as the current value is less than the set value, output 1 will remain OFF and output 2 will remain ON.

   b) When the current value reaches the set value, the current value will no longer be incremented, output 1 will turn ON, and output 2 will turn OFF.

2) If input 1 turns ON and OFF repeatedly while input 2 is ON, the time that input 1 is ON will be measured and the operation described for item 1) will be performed.

3) If input 2 is OFF, timing will not be performed regardless of the status of input 1, the current value will be set to 0, output 1 will turn OFF, and output 2 will turn ON.

4) The following table summarizes the operation of T.01.

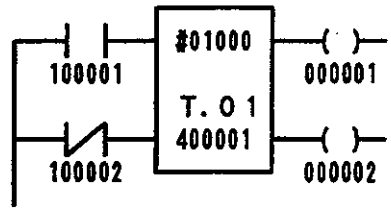
Table 1.12 Operation of T.01

Inputs		Timer Status		Current Value	Outputs	
1	2				1	2
Any	OFF	Reset		0	OFF	**ON**
ON	**ON**	Running	Current value < Set value	Incrementing	OFF	**ON**
			Current value = Set value	Set value	**ON**	OFF
OFF	**ON**	Stopped	Current value < Set value	Not changed	OFF	**ON**
			Current value = Set value	Set value	**ON**	OFF

◀EXAMPLE▶

4. Application Example

1) Ladder Programming



Set value: 1000
Timing range: 0.01 to 10.00 s

2) Operation

a) Timer operation is enabled when input relay 100002 is OFF.

b) The timer will start when input 100001 turns ON while input relay 100002 is OFF and the current value (contents of holding register 400001) will be incremented 1 each 0.01 second. Coil 000001 will be OFF and coil 000002 will be ON.

c) When the current value reaches the set value (1000), the timer will stop and the current value will no longer be incremented. Coil 000001 will turn ON and coil 000002 will turn OFF.

d) If input relay 100001 turns ON and OFF repeatedly while input relay 100002 is OFF, the time that input relay 100001 is ON will be measured and the operation described for items b) and c) will be performed.

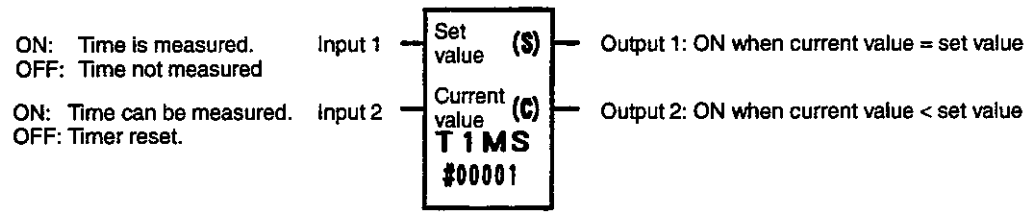
e) If input relay 100002 is ON, timing will not be performed regardless of the status of input relay 100001, the current value will be set to 0, coil 000001 will turn OFF, and coil 000002 will turn ON.

1.2.5 0.001-SECOND TIMER (T1MS)

1. Function

0.001-SECOND TIMER is used to measure time in 0.001-s intervals between 0.001 and 65.535 s.

2. Structure



1) T1MS is the symbol for 0.001-SECOND TIMER.

2) T1MS requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 1.13* lists the register reference numbers and constants that can be specified.

Example

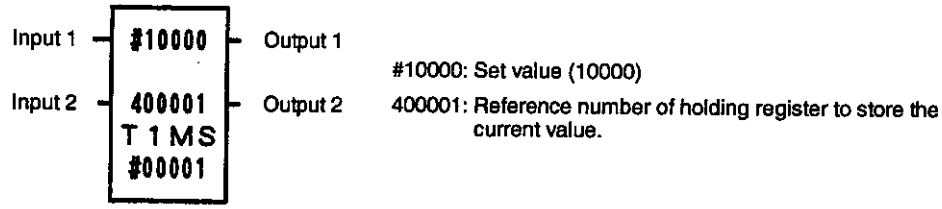


Table 1.13 Structural Elements of T1MS

Element	Meaning	Possible Settings	
Top (S)	The value of the constant or the contents of the register with the specified reference number will be used as the set value (S) of the timer. The value of S must be between 0 and 65,535.	Constant:	#00000 to #65535
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 R20001 to R21024
Middle (C)	The current value of the timer (C) will be stored in the specified register. The value of C will be between 0 and the set value.	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 R20001 to R21024
Bottom	—	Constant:	#00001

3. Operation

1) If input 2 is ON, the following timing operation will be performed by T1MS while input 1 is ON.

   a) The current value will be incremented by 1 every 0.001second. As long as the current value is less than the set value, output 1 will remain OFF and output 2 will remain ON.

   b) When the current value reaches the set value, the current value will no longer be incremented, output 1 will turn ON, and output 2 will turn OFF.

2) If input 1 turns ON and OFF repeatedly while input 2 is ON, the time that input 1 is ON will be measured and the operation described for item 1) will be performed.

3) If input 2 is OFF, timing will not be performed regardless of the status of input 1, the current value will be set to 0, output 1 will turn OFF, and output 2 will turn ON.

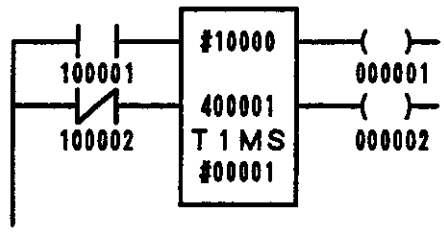
4) The following table summarizes the operation of T1MS.

Table 1.14 Operation of T1MS

Inputs		Timer Status		Current Value	Outputs	
1	2				1	2
Any	OFF	Reset		0	OFF	**ON**
ON	**ON**	Running	Current value < Set value	Incrementing	OFF	**ON**
			Current value = Set value	Set value	**ON**	OFF
OFF	**ON**	Stopped	Current value < Set value	Not changed	OFF	**ON**
			Current value = Set value	Set value	**ON**	OFF

◀EXAMPLE▶ **4. Application Example**

1) Ladder Programming



Set value: 10000
Timing range: 0.001 to 10.000 s

2) Operation

a) Timer operation is enabled when input relay 100002 is OFF.

b) The timer will start when input 100001 turns ON while input relay 100002 is OFF and the current value (contents of holding register 400001) will be incremented 1 each 0.001 second. Coil 000001 will be OFF and coil 000002 will be ON.

c) When the current value reaches the set value (10000), the timer will stop and the current value will no longer be incremented. Coil 000001 will turn ON and coil 000002 will turn OFF.

d) If input relay 100001 turns ON and OFF repeatedly while input relay 100002 is OFF, the time that input relay 100001 is ON will be measured and the operation described for items b) and c) will be performed.

e) If input relay 100002 is ON, timing will not be performed regardless of the status of input relay 100001, the current value will be set to 0, coil 000001 will turn OFF, and coil 000002 will turn ON.

1.2.6 Building Timer Circuits

1. Storage Locations on Networks

1) 1-SECOND TIMER, 0.1-SECOND TIMER, and 0.01-SECOND TIMER

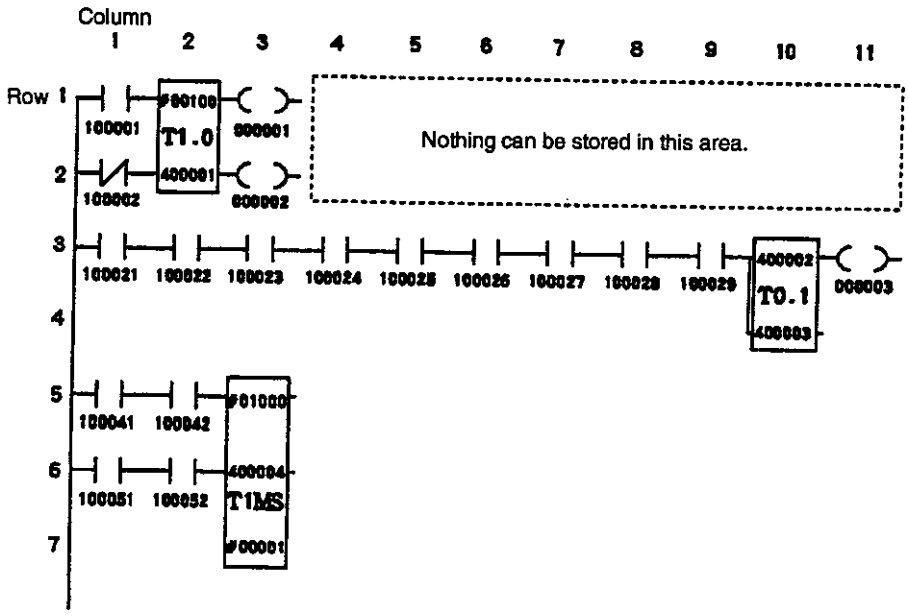
These three timers require two elements on a network, one top element and one bottom element. They can thus be stored anywhere on a 6-row by 10-column matrix (rows 1 through 6 and columns 1 through 10) on the network.

2) 0.001-SECOND TIMER

The 0.001-SECOND TIMER requires three elements on a network, one top element, one middle element, and one bottom element. It can thus be stored anywhere on a 5-row by 10-column matrix (rows 1 through 5 and columns 1 through 10) on the network.

Note Timers cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

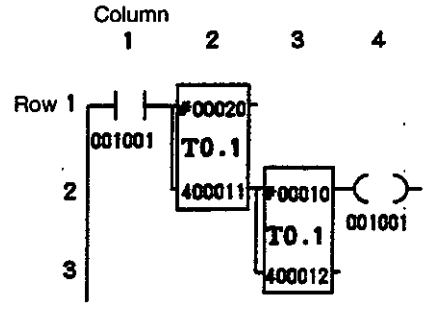
Example



2. Timer Inputs

Timer inputs 1 and 2 can be connected to contacts and/or outputs from other timers, counters, math instructions, other instructions, etc. Coils cannot be connected to inputs.

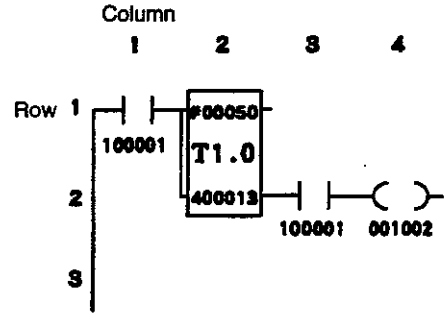
Example



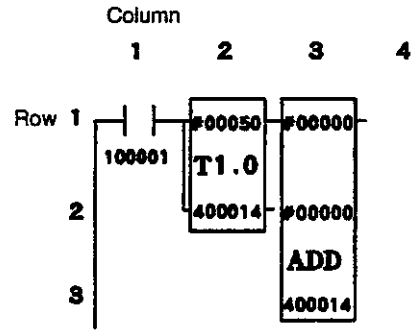
3. Timer Outputs

Timer outputs 1 and 2 can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to other instructions, etc.

Example 1



Example 2



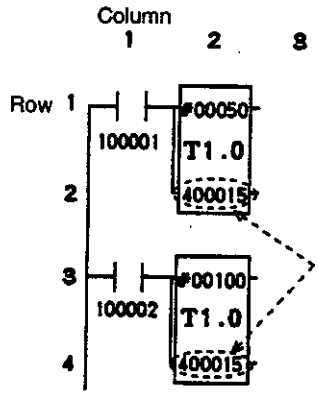
4. Storage Registers for Timer Current Values

Note Unless there is a very specific reason for doing so, never use the storage register of a timer instruction for the following purposes (see following illustration). Unexpected operation may result.

1) As the storage register for the current value of a counter or another timer.

2) As the storage register for the results of a math instruction or any other instruction.

Example: Incorrect Application

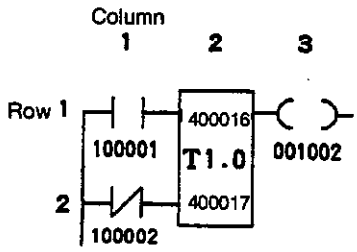


No: Do not use the same reference number in more than one instruction without a specific and planned reason.

5. Set Values for Timers

Note If zero is input as the set value for a timer, output 1 will turn ON and output 2 will turn OFF as long as input 2 is ON regardless of the status of input 1.

Example



Assuming that input relay 100002 is OFF and that the contents of holding register 400016 is set to 0, coil 001002 will turn ON regardless of the status of input relay 100001.

6. Timer Set Values and Current Values

In normal application, the current value of the timer will never exceed the set value. Numeric or other instructions can, however, be used to output values to a register holding a current value, thus possibly making the contents larger than the set value. If this happens, the current value will be reset to the set value the next time the timer is solved.

7. Timing Error

Note The following equation can be used to calculate the timing error of timers.

Max. error = Min. unit of set value + 1 scan time.

For example, if 1-SECOND TIMER is used, there will be a timing error of up to 1 s, i.e., the timer may time-out up to 1 s too early. If this possible error is a problem for the applications, use one of the other timer instructions to achieve greater accuracy.

8. Current Values of Timers at Power-up

Note (1) The present values of timers are saved in memory even if the operation of the CPU Module is temporarily stopped due to power interruptions or other causes. When CPU Module operation is restarted, the current value will be cleared to 0 or set to the value recorded in memory depending on the ON/OFF status of timer input 2.

(2) The following table lists the treatment of timer current values when power is resupplied and input relays or contact for coils are connected to timer input 2.

Table 1.15 Timer Present Value Treatment for Restarts

Contact Connected to Input 2*2				Timer Current Value First Scan After Restarting
Type of Reference		Status	Contact type	
Input relay Latched coil		ON	N.O.	Value in memory*3
			N.C.	0
		OFF	N.O.	0
			N.C.	Value in memory
Normal coil	n≦m*1	ON	N.O.	Value in memory
			N.C.	0
		OFF	N.O.	0
			N.C.	Value in memory
	n>m*1	OFF	N.O.	0
			N.C.	Value in memory

*1: "n" is the number of the network where the coil corresponding to the contact connected to input 2 is programmed and "m" is the number of the network where the timer is programmed.

*2: If more than one contact is connected to input 2, the above logic can be applied to the results of the contacts.

*3: The value in memory is the value saved when power was interrupted.

1.3 Counters

This section describes the counter instructions available for programming, including their functions, structure, operation, and application. Building counter circuits is also discussed and related precautions are provided.

1.3.1 Counter Instructions

The following table list the two counter instructions. As many of the following instructions can be used as long as the user memory, holding register, or link register capacity is not exceeded.

Table 1.16 Counter Instructions

Instruction Name	Symbol	Counting Unit*1	Counting Range*2
UP COUNTER	UCTR	1 pulse	1 to 65,535 pulse
DOWN COUNTER	DCTR		

*1: The counting unit is the unit used to count pulses. Both UP COUNTER and DOWN COUNTER count pulses one at a time.

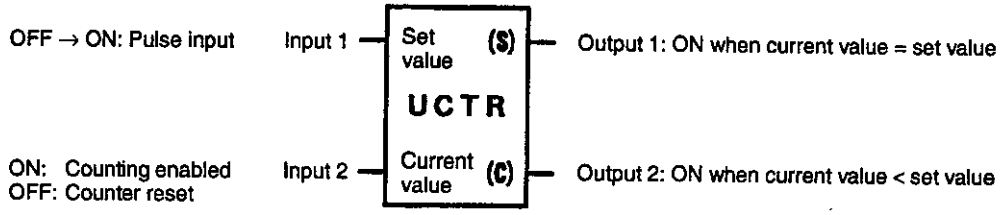
*2: The counting range is the range in which pulses are counted. UP COUNTER and DOWN COUNTER count pulses between 1 and 65,535.

1.3.2 UP COUNTER (UCTR)

1. Function

UP COUNTER counts pulses and increments the current value one at a time. The counting range is between 1 and 65,535.

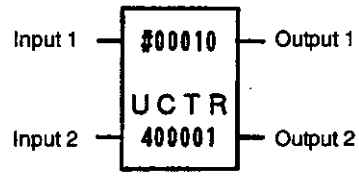
2. Structure



1) UCTR is the symbol for UP COUNTER.

2) UCTR requires two elements, one top element and one bottom element, located vertically on the network. *Table 1.17* lists the register reference numbers and constants that can be specified.

Example



#00010: Set value (10)

400001: Reference number of holding register to store the current value.

Table 1.17 Structural Elements of UCTR

Element	Meaning	Possible Settings	
Top (S)	The value of the constant or the contents of the register with the specified reference number will be used as the set value (S) of the counter. The value of S must be between 0 and 65,535.	Constant:	#00000 to #65535
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 R20001 to R21024
Bottom (C)	The current value of the counter (C) will be stored in the specified register. The value of C will be between 0 and the set value.	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 R20001 to R21024

3. Operation

1) If input 2 is ON, the following counting operation will be performed by UCTR.

   a) The current value will be incremented by 1 each time input 1 goes from OFF to ON. As long as the current value is less than the set value, output 1 will remain OFF and output 2 will remain ON.

   b) When the current value reaches the set value, the current value will no longer be incremented, output 1 will turn ON, and output 2 will turn OFF.

2) If input 2 is OFF, counting will not be performed regardless of the status of input 1, the current value will be set to 0, output 1 will turn OFF, and output 2 will turn ON.

3) The following table summarizes the operation of UCTR.

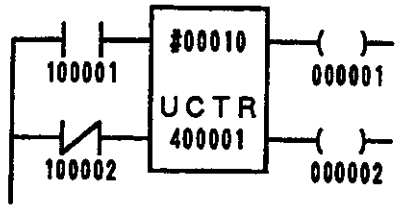
Table 1.18 Operation of UCTR

Inputs		Counter Status		Current Value	Outputs	
1	2				1	2
Any	OFF	Reset		0	OFF	**ON**
OFF→**ON**	**ON**	Running	Current value < Set value	Incrementing	OFF	**ON**
			Current value = Set value	Set value	**ON**	OFF
OFF ON ON→OFF	**ON**	Stopped	Current value < Set value	Not changed	OFF	**ON**
			Current value = Set value	Set value	**ON**	OFF

◀EXAMPLE▶

4. Application Example

1) Ladder Programming



2) Operation

a) Counter operation is enabled when input relay 100002 is OFF.

b) The counter will increment the current value (contents of holding register 400001) by 1 each time that input relay 100001 goes from OFF to ON while input relay 100002 is OFF. Coil 000001 will be OFF and coil 000002 will be ON.

c) When the current value reaches the set value (10), the counter operation will stop and the current value will no longer be incremented. Coil 000001 will turn ON and coil 000002 will turn OFF.

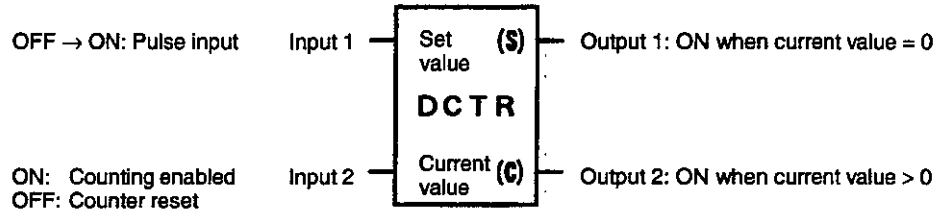
d) If input relay 100002 is ON, counting will not be performed regardless of the status of input relay 100001, the current value will be set to 0, coil 000001 will turn OFF, and coil 000002 will turn ON.

1.3.3 DOWN COUNTER (DCTR)

1. Function

DOWN COUNTER counts pulses and decrements the current value one at a time. The counting range is between 1 and 65,535.

2. Structure



1) DCTR is the symbol for DOWN COUNTER.

2) DCTR requires two elements, one top element and one bottom element, located vertically on the network. *Table 1.19* lists the register reference numbers and constants that can be specified.

Example

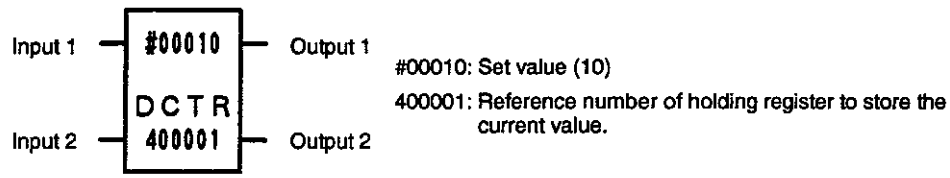


Table 1.19 Structural Elements of DCTR

Element	Meaning	Possible Settings	
Top (S)	The value of the constant or the contents of the register with the specified reference number will be used as the set value (S) of the counter. The value of S must be between 0 and 65,535.	Constant:	#00000 to #65535
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 R20001 to R21024
Bottom (C)	The current value of the counter (C) will be stored in the specified register. The value of C will be between 0 and the set value.	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 R20001 to R21024

3. Operation

1) If input 2 is ON, the following counting operation will be performed by DCTR.

   a) The current value will be decremented by 1 each time input 1 goes from OFF to ON. As long as the current value is greater than zero, output 1 will remain OFF and output 2 will remain ON.

   b) When the current value reaches zero, the current value will no longer be decremented, output 1 will turn ON, and output 2 will turn OFF.

2) If input 2 is OFF, counting will not be performed regardless of the status of input 1, the current value will be equal to the set value, output 1 will turn OFF, and output 2 will turn ON.

3) The following table summarizes the operation of DCTR.

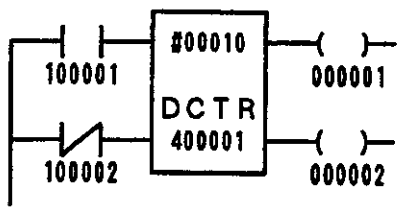
Table 1.20 Operation of DCTR

Inputs		Counter Status		Current Value	Outputs	
1	2				1	2
Any	OFF	Reset		Set value	OFF	**ON**
OFF→**ON**	**ON**	Running	Current value > 0	Decrementing	OFF	**ON**
			Current value = 0	0	**ON**	OFF
OFF ON ON→OFF	**ON**	Stopped	Current value > 0	Not changed	OFF	**ON**
			Current value = 0	0	**ON**	OFF

◀EXAMPLE▶

4. Application Example

1) Ladder Programming



2) Operation

a) Counter operation is enabled when input relay 100002 is OFF.

b) The counter will decrement the current value (contents of holding register 400001) by 1 each time that input relay 100001 goes from OFF to ON while input relay 100002 is OFF. Coil 000001 will be OFF and coil 000002 will be ON.

c) When the current value reaches the set value (10), the counter operation will stop and the current value will no longer be decremented. Coil 000001 will turn ON and coil 000002 will turn OFF.

d) If input relay 100002 is ON, counting will not be performed regardless of the status of input relay 100001, the current value will be set to 10, coil 000001 will turn OFF, and coil 000002 will turn ON.

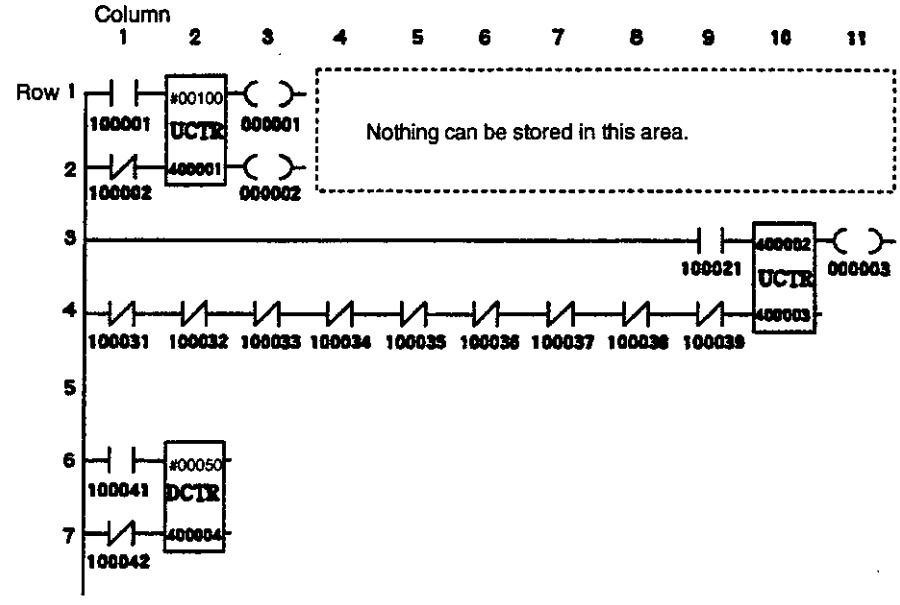
1.3.4 Building Counter Circuits

1. Storage Locations on Networks

Both UCTR and DCTR require two elements on a network, one top element and one bottom element. They can thus be stored anywhere on a 6-row by 10-column matrix (rows 1 through 6 and columns 1 through 10) on the network.

Note Counters cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

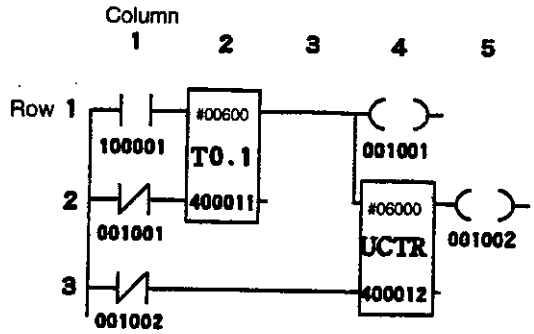
Example



2. Counter Inputs

Counter inputs 1 and 2 can be connected to contacts and/or outputs from other counters, timers, math instructions, other instructions, etc.

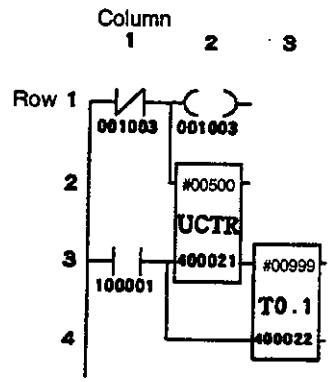
Example



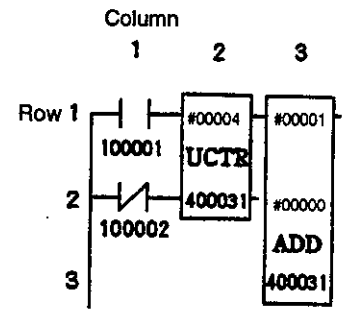
3. Counter Outputs

Counter outputs 1 and 2 can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to other instructions, etc.

Example 1



Example 2



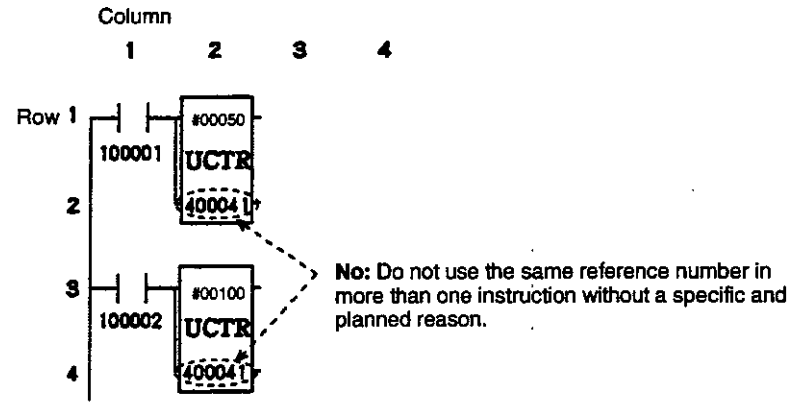
4. Storage Registers for Counter Current Values

Note Unless there is a very specific reason for doing so, never use the storage register for a counter instruction for the following purposes (see following illustration). Unexpected operation may result.

1) As the storage register for the current value of another counter or timer.

2) As the storage register for the results of a math instruction or any other instruction.

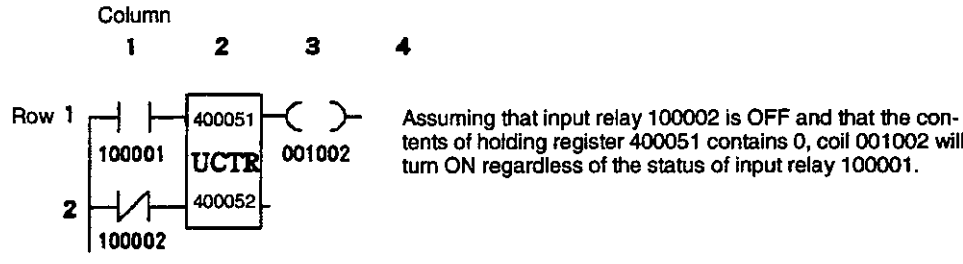
Example: Incorrect Application



5. Set Values for Counters

Note If zero is input as the set value for a counter, output 1 will turn ON and output 2 will turn OFF as long as input 2 is ON regardless of the status of input 1.

Example



6. Counter Set Values and Current Values

In normal application, the current value of the counter will never exceed the set value. Numeric or other instructions can, however, be used to output values to a register holding a current value, thus possibly making the contents larger than the set value. If this happens, the current value will be reset to the set value the next time the counter is solved.

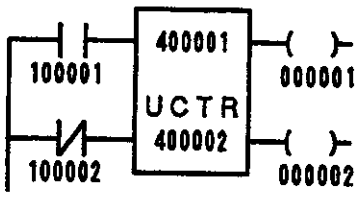
7. Pulse Width

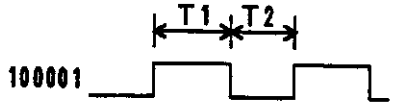
Note When using counter instruction to count signals input from external devices through Digital Input Modules, the signal ON and OFF times must both be longer than the value of T computed using the following equation:

T = 1 scan time + input delay of the Input Module (ms)

If the input signal ON time or OFF time is shorter than T, input pulses will not be counted properly.

Example





$T_1, T_2 > T$
T = 1 scan time + input delay of the Input Module (ms)

8. Current Values of Counters at Power-up

Note (1) The present values of counters are saved in memory even if the operation of the CPU Module is temporarily stopped due to power interruptions or other causes. When CPU Module operation is restarted, the current value will be set to 0, the set value, or the value recorded in memory depending on the ON/OFF status of counter input 2.

(2) The following table lists the treatment of counter current values when power is resupplied and input relays or contacts for coils are connected to counter input 2.

Table 1.21 Counter Present Value Treatment for Restarts

Contact Connected to Input 2*2				Counter Current Value First Scan After Restarting	
Type of Reference		Status	Contact type	UP COUNTER	DOWN COUNTER
Input relay Latched coil		ON	N.O.	Value in memory*3	Value in memory
			N.C.	0	Set value
		OFF	N.O.	0	Set value
			N.C.	Value in memory	Value in memory
Normal coil	n ≤ m*1	ON	N.O.	Value in memory	Value in memory
			N.C.	0	Set value
		OFF	N.O.	0	Set value
			N.C.	Value in memory	Value in memory
	n > m*1	OFF	N.O.	0	Set value
			N.C.	Value in memory	Value in memory

*1: "n" is the number of the network where the coil corresponding to the contact connected to input 2 is programmed and "m" is the number of the network where the counter is programmed.

*2: If more than one contact is connected to input 2, the above logic can be applied to the results of the contacts.

*3: The value in memory is the value saved when power was interrupted.

Math Instructions

2

This chapter describes instuctions used to perform math operations.

2.1 Math Instructions

This section introduces the various math instructions that can be used in programming.

1) There are two groups of math instructions:

- Group 1 Math Instructions
- Group 2 Math Instructions

a) Group 1 Math Instructions

(1) Group 1 math instructions include those supported by the GL60H and other older PLCs.

(2) Group 1 math instructions are outlined in *Table 2.1.*

(3) The use of group 1 instructions is recommended in the following cases:

i. To preserve compatibility with user programs for older PLCs.

ii. When square root or trigonometric functions are required.

b) Group 2 Math Instructions

(1) Group 2 math instructions are new for the GL120 and GL130.

(2) Group 2 math instructions are outlined in *Table 2.2.*

(3) The use of group 2 instructions is recommended in the following cases:

i. To preserve compatibility with user programs for older PLCs is not important.

ii. When the following Modules are being used:

Analog I/O Module
Counter Modules
1-axis Motion Modules
4-axis Motion Modules

2) Math instructions are separated into two groups because the expressions used to handle numbers are very different. For details, refer to *2.2 Expressing Numbers.* Be sure that you understand the required numeric expressions before you attempt to use a math instruction.

3) Group 1 math instructions are outlined in *Table 2.1*, below. In the table V1 represents the first operand (operand 1) and V2 represents the second operand (operand 2).

Table 2.1 Group 1 Math Instructions

Class	Name	Symbol	Operands	V1	V2	Results
Unsigned, 4-digit, decimal arithmetic instructions	UNSIGNED SINGLE PRECISION DECIMAL ADDITION	ADD	V1 + V2	0 to 9,999		
	UNSIGNED SINGLE PRECISION DECIMAL SUBTRACTION	SUB	V1 – V2 Comparison			
	UNSIGNED SINGLE PRECISION DECIMAL MULTIPLICATION	MUL	V1 × V2	0 to 9,999		0 to 99,980,001
	UNSIGNED SINGLE PRECISION DECIMAL DIVISION	DIV	V1 ÷ V2	0 to 9,999 or 0 to 99,989,999	0 to 9,999	
Unsigned, 8-digit, decimal arithmetic instructions	UNSIGNED DOUBLE PRECISION DECIMAL ADDITION	DADD	V1 + V2	0 to 99,999,999		
	UNSIGNED DOUBLE PRECISION DECIMAL SUBTRACTION	DSUB	V1 – V2 Comparison			
	UNSIGNED DOUBLE PRECISION DECIMAL MULTIPLICATION	DMUL	V1 × V2	0 to 99,999,999		0 to 9,999,999,800,000,001
	UNSIGNED DOUBLE PRECISION DECIMAL DIVISION	DDIV	V1 ÷ V2	0 to 9,999,999,899,999,999	0 to 99,999,999	
Signed, 4-digit, decimal arithmetic instructions	SIGNED SINGLE PRECISION DECIMAL ADDITION	SADD	V1 + V2	–9,999 to 9,999		
	SIGNED SINGLE PRECISION DECIMAL SUBTRACTION	SSUB	V1 – V2			
	SIGNED SINGLE PRECISION DECIMAL MULTIPLICATION	SMUL	V1 × V2	–9,999 to 9,999		–99,980,001 to 99,980,001
	SIGNED SINGLE PRECISION DECIMAL DIVISION	SDIV	V1 ÷ V2	–99,989,999 to 99,989,999	–9,999 to 9,999	
Signed, 8-digit, decimal arithmetic instructions	SIGNED DOUBLE PRECISION DECIMAL ADDITION	SDAD	V1 + V2	–99,999,999 to 99,999,999		
	SIGNED DOUBLE PRECISION DECIMAL SUBTRACTION	SDSB	V1 – V2			
Decimal square root instructions	SINGLE PRECISION DECIMAL SQUARE ROOT	SQRT	$\sqrt{V}$	0 to 9,999		0 to 99.9949
	DOUBLE PRECISION DECIMAL SQUARE ROOT	DSQR		0 to 99,999,999		0 to 9999.9999

Class	Name	Symbol	Operands	V1	V2	Results
Decimal trigonometric instruction	DECIMAL SINE	SIN	|SINθ|	0.0000° to 360.0000°		0.0000 to 1.0000
	DECIMAL COSINE	COS	|COSθ|			

4) Group 2 math instructions are outlined in *Table 2.2*, below. In the table V1 represents the first operand (operand 1) and V2 represents the second operand (operand 2).

Table 2.2 Group 2 Math Instructions

Class	Name	Symbol	Operands	V1	V2	Results
16-bit arithmetic instructions	16-BIT ADDITION	AD16	V1 + V2	1) Unsigned: 0 to 65,535 2) Signed: –32,768 to 32,767		
	16-BIT SUBTRACTION	SU16	V1 – V2 Comparison			
	16-BIT MULTIPLICATION	MU16	V1 × V2	1) Unsigned: 0 to 65,535 2) Signed: –32,768 to 32,767		1) Unsigned: 0 to 4,294,967,295 2) Signed: –2,147,483,648 to 2,147,483,647
	16-BIT DIVISION	DV16	V1 ÷ V2	1) Unsigned: 0 to 4,294,967,295 2) Signed: –2,147,483,648 to 2,147,483,647	1) Unsigned: 0 to 65,535 2) Signed: –32,768 to 32,767	
32-bit arithmetic instructions	32-BIT ADDITION	AD32	V1 + V2	1) Unsigned: 0 to 4,294,967,295 2) Signed: –2,147,483,648 to 2,147,483,647		
	32-BIT SUBTRACTION	SU32	V1 – V2 Comparison			
	32-BIT COMPARE	TEST	Comparison	• **16-bit Comparison** 1) Unsigned: 0 to 65,535 2) Signed: –32,768 to 32,767 • **32-bit Comparison** 1) Unsigned: 0 to 4,294,967,295 2) Signed: –2,147,483,648 to 2,147,483,647		

2.2 Expressing Numbers

This section describes the expressions used in math instructions.

2.2.1 Numeric Expressions

1) Two different types of numeric expressions are used for the GL120 and GL130, as described below.

a) Group 1 Numeric Expressions

Group 1 numeric expressions are the same as those used for the GL60H and other previously released PLCs and they are used for the group 1 math instructions described in the previous section.

b) Group 2 Numeric Expressions

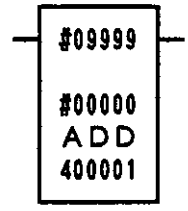
Group 2 numeric expressions are the used for the new GL120 and GL130 instructions, and they are thus used for the group 2 math instructions described in the previous section.

2) Group 1 Numeric Expressions

a) Constants

Constants are handled as 4-digit, decimal integers and 0, i.e., 0 to 9,999.

Example: UNSIGNED SINGLE PRECISION DECIMAL ADDITION

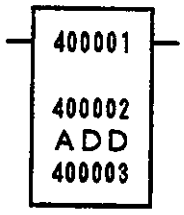


#09999: Augend (9,999)
#00000: Addend (0)
400001: Stores the results (9,999).

b) Storing Numeric Values in Registers

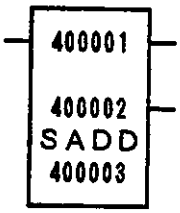
(1) One register can store a single 4-digit, decimal value, i.e., –9,999 to 9,999.

Example 1: UNSIGNED SINGLE PRECISION DECIMAL ADDITION



400001: Stores the augend between 0 and 9,999.
400002: Stores the addend between 0 and 9,999.
400003: Stores the results between 0 and 9,999.

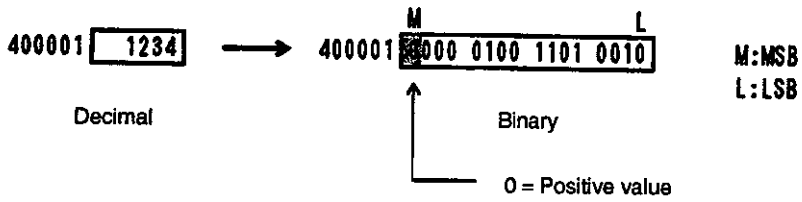
Example 2: SIGNED SINGLE PRECISION DECIMAL ADDITION



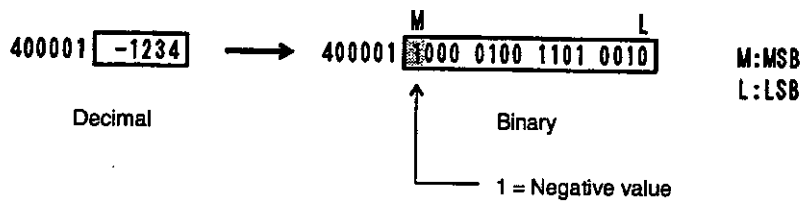
400001 :Stores the augend between –9,999 and 9,999.
400002: :Stores the addend between –9,999 and 9,999.
400003: :Stores the results between –9,999 and 9,999.

(2) The sign of the numeric value is expressed by separating the sign and the absolute value of the number.
Positive values: MSB of register is set to 0.
Negative values: MSB of register is set to 1.

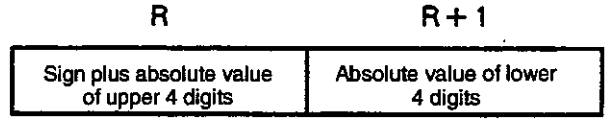
Example 1: Storing a Positive, 4-digit, Decimal Integer (+1,234)



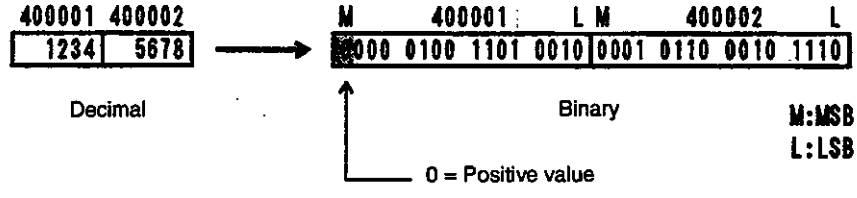
Example 2: Storing a Negative, 4-digit, Decimal Integer (–1,234)



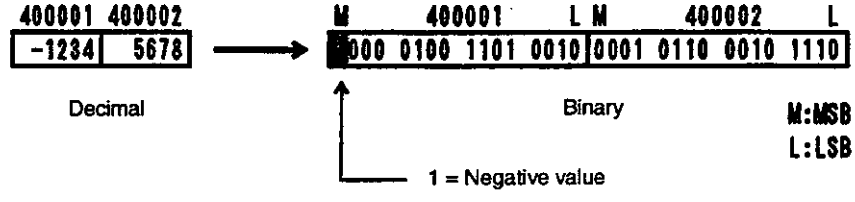
(3) Reference numbers for two consecutive registers, R and R+1, can be used to store 8-digit, decimal values. Register R is the register with the lower reference number and it holds the sign plus the absolute value of the upper 4 digits. Register R + 1 is the register with the higher reference number and it holds the absolute value of the lower 4 digits.



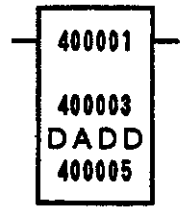
Example 1: Storing a Positive, 8-digit, Decimal Integer (+12,345,678)



Example 2: Storing a Negative, 8-digit, Decimal Integer (–12,345,678)



Example 3: UNSIGNED DOUBLE PRECISION DECIMAL ADDITION



V1 (0 to 99,999,999) is stored as follows:

400001	400002
Upper 4 digits	Lower 4 digits

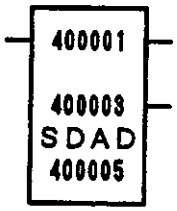
V2 (0 to 99,999,999) is stored as follows:

400003	400004
Upper 4 digits	Lower 4 digits

The results (0 to 99,999,999) is stored as follows:

400005	400006
Upper 4 digits	Lower 4 digits

Example 4: SIGNED DOUBLE PRECISION DECIMAL ADDITION



V1 (–99,999,999 to 99,999,999) is stored as follows:

400001	400002
Sign + Upper 4 digits	Lower 4 digits

V2 (–99,999,999 to 99,999,999) is stored as follows:

400003	400004
Sign + Upper 4 digits	Lower 4 digits

The results (–99,999,999 to 99,999,999) is stored as follows:

400005	400006
Sign + Upper 4 digits	Lower 4 digits

(4) Reference numbers for four consecutive registers, R, R+1, R+2, and R+3, can be used to store 16-digit, decimal values. A 16-digit, decimal positive integer is stored as follows:

R	R+1	R+2	R+3
Most upper 4 digits	Upper 4 digits	Lower 4 digits	Most lower 4 digits

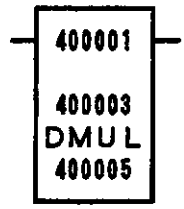
Example 1: Storing a Positive, 16-digit, Decimal Integer (+1,234,567,890,123,456)

	400001	400002	400003	400004	
	1234	5678	9012	3456	Decimal
	M			L	
400001	0000	0100	1101	0010	Most upper 4 digits
400002	0001	0110	0010	1110	Upper 4 digits
400003	0010	0011	0011	0100	Lower 4 digits
400004	0000	1101	1000	0000	Most lower 4 digits

Binary

M:MSB
L:LSB

Example 2: UNSIGNED DOUBLE PRECISION DECIMAL MULTIPLICATION



V1 (0 to 99,999,999) is stored as follows:

400001	400002
Upper 4 digits	Lower 4 digits

V2 (0 to 99,999,999) is stored as follows:

400003	400004
Upper 4 digits	Lower 4 digits

The results (0 to 9,999,999,800,000,001) is stored as follows:

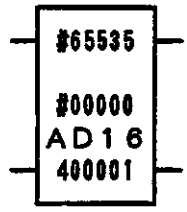
400005	400006	400007	400008
Most upper 4 digits	Upper 4 digits	Lower 4 digits	Most lower 4 digits

3) Group 2 Numeric Expressions

a) Constants

Constants are handled as 16-bit, binary numbers, i.e., 0 to 65,535.

Example: 16-BIT ADDITION for Unsigned Numbers

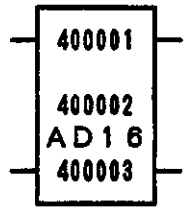


#65535: Augend (65,535)
#00000: Addend (0)
400001: Stores the results (65,535)

b) Storing Numeric Values in Registers

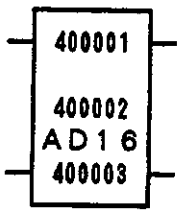
(1) One register can store a single 16-bit binary integer, i.e., 0 to 65,535 for unsigned values and –32,768 to 32,767 for signed values.

Example 1: 16-BIT ADDITION for Unsigned Numbers



400001: Stores the augend between 0 and 65,535.
400002: Stores the addend between 0 and 65,535.
400003: Stores the results between 0 and 65,535.

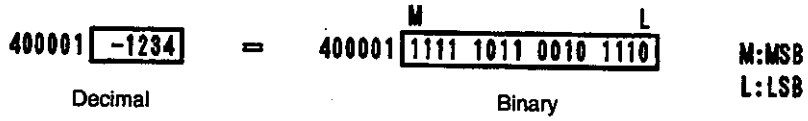
Example 2: 16-BIT ADDITION for Signed Numbers



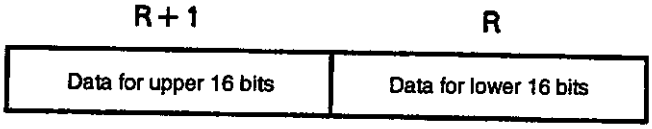
400001: Stores the augend between –32,768 to 32,767
400002: Stores the addend between –32,768 to 32,767
400003: Stores the results between –32,768 to 32,767

(2) The two's complement is used for negative numbers.

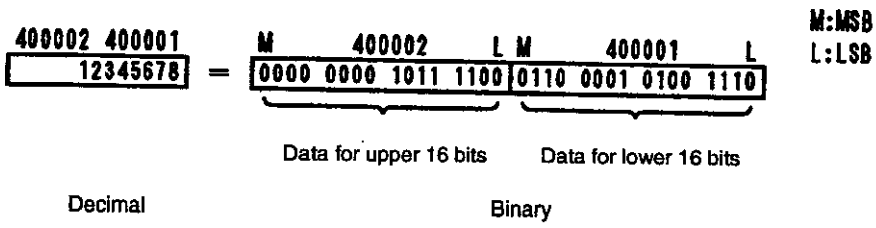
Example: Storing a Negative, 4-digit, Decimal Integer (–1,234)



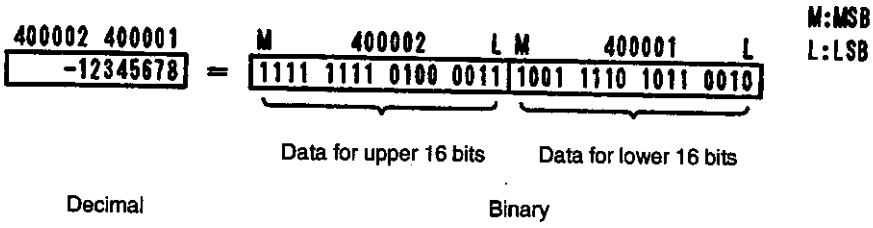
(3) Reference numbers for two consecutive registers, R and R+1, can be used to store 32-bit, binary values. Using two registers, 0 to 4,294,967,295 can be stored for unsigned values and –2,147,483,648 to 2,147,483,647 can be stored for signed values. Register R is the register with the lower reference number and it holds the sign plus the value of the upper 16 bits. Register R+1 is the register with the higher reference number and it holds the value of the lower 16 bits.



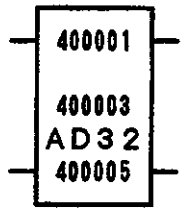
Example 1: Storing a Positive, 8-digit, Decimal Integer (12,345,678)



Example 2: Storing a Negative, 8-digit, Decimal Integer (–12,345,678)



Example 3: 32-BIT ADDITION for Unsigned Values



V1 (0 to 4,294,967,295) is stored as follows:

400002	400001
Upper 16 bits	Lower 16 bits

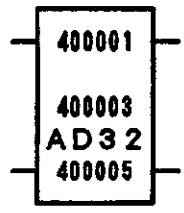
V2 (0 to 4,294,967,295) is stored as follows:

400004	400003
Upper 16 bits	Lower 16 bits

The results (0 to 4,294,967,295) is stored as follows:

400006	400005
Upper 16 bits	Lower 16 bits

Example 4: 32-BIT ADDITION for Signed Values



V1 (–2,147,483,648 to 2,147,483,647) is stored as follows:

400002	400001
Upper 16 bits	Lower 16 bits

V2 (–2,147,483,648 to 2,147,483,647) is stored as follows:

400004	400003
Upper 16 bits	Lower 16 bits

The results (–2,147,483,648 to 2,147,483,647) is stored as follows:

400006	400005
Upper 16 bits	Lower 16 bits

2.2.2 Converting Numeric Expressions

1) The two type of expressions used for math instructions of the GL120 and GL130 were described in the previous section. Although most applications will use either one or the other type of expression, some applications will require both forms, meaning that the data will have to be converted from one type of numeric expression to the other.

   a) Group 1 numeric expressions can be converted to group 2 numeric expressions.

   b) Group 2 numeric expressions can be converted to group 1 numeric expressions.

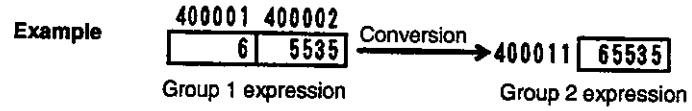
2) The following two data conversion instructions are provided to convert between the two types of numeric expression. These instructions are introduced here. Refer to chapter 7 *Data Conversion Instructions* for details.

   • SINGLE WORD DATA CONVERSION (CAST)

   • DOUBLE WORD DATA CONVERSION (DCST)

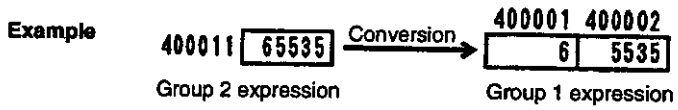
a) SINGLE WORD DATA CONVERSION (CAST)

(1) A 5-digit, unsigned, decimal integer (0 to 65,535) using group 1 numeric expression can be converted to group 2 numeric expression.

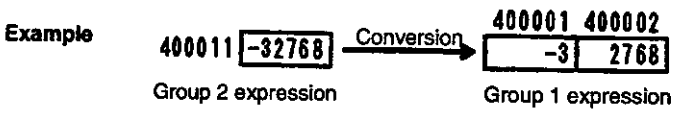


(2) A 5-digit, signed, decimal integer (–32,768 to 32,767) using group 1 numeric expression can be converted to group 2 numeric expression.

(3) A 5-digit, unsigned, decimal integer (0 to 65,535) using group 2 numeric expression can be converted to group 1 numeric expression.

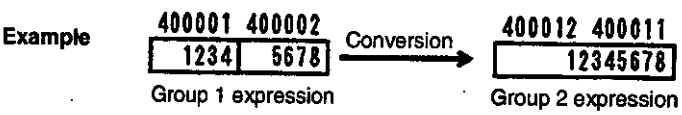


(4) A 5-digit, signed, decimal integer (–32,768 to 32,767) using group 2 numeric expression can be converted to group 1 numeric expression.

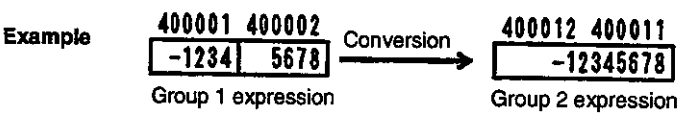


b) DOUBLE WORD DATA CONVERSION (DCST)

(1) An 8-digit, unsigned, decimal integer (0 to 99,999,999) using group 1 numeric expression can be converted to group 2 numeric expression.

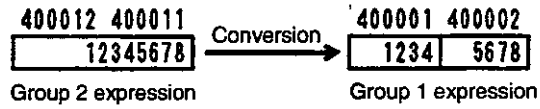


(2) An 8-digit, signed, decimal integer (–99,999,999 to 99,999,999) using group 1 numeric expression can be converted to group 2 numeric expression.



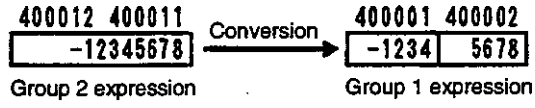
(3) An 8-digit, unsigned, decimal integer (0 to 99,999,999) using group 2 numeric expression can be converted to group 1 numeric expression.

Example



(4) An 8-digit, signed, decimal integer (–99,999,999 to 99,999,999) using group 2 numeric expression can be converted to group 1 numeric expression.

Example



2.3 Unsigned, Four-digit, Decimal Arithmetic Instructions

This section describes the functions, structures, and operation of the unsigned, 4-digit, decimal arithmetic instructions and provides simple examples of their application.

2.3.1 Instruction

Unsigned, 4-digit, decimal arithmetic instructions perform unsigned addition, subtraction, multiplication, and division on two 4-digit, decimal values, V1 and V2. The instructions that are available are shown in *Table 2.3.*

Table 2.3 Unsigned, 4-digit, Decimal Arithmetic Instructions

Name	Symbol	Operands	V1	V2	Result
UNSIGNED SINGLE PRECISION DECIMAL ADDITION	ADD	V1 + V2	0 to 9,999		
UNSIGNED SINGLE PRECISION DECIMAL SUBTRACTION	SUB	V1 – V2 Comparison	0 to 9,999		
UNSIGNED SINGLE PRECISION DECIMAL MULTIPLICATION	MUL	V1 × V2	0 to 9,999		0 to 99,980,001
UNSIGNED SINGLE PRECISION DECIMAL DIVISION	DIV	V1 ÷ V2	0 to 9,999 or 0 to 99,989,999*1	0 to 9,999	

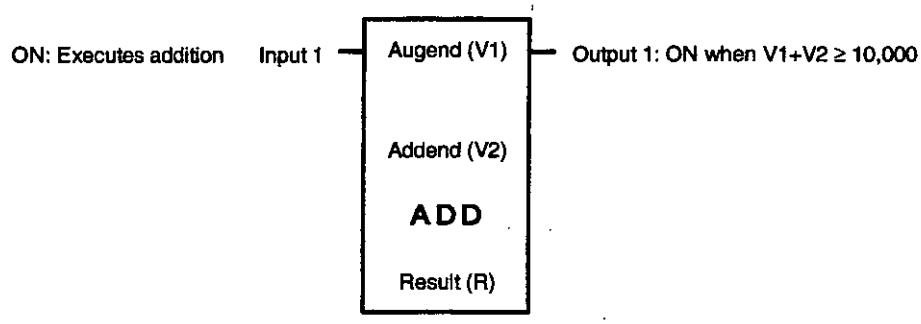
*1: The wider range for V1 is possible by using two continuous registers.

2.3.2 UNSIGNED SINGLE PRECISION DECIMAL ADDITION (ADD)

1. Function

Unsigned addition is performed between two 4-digit decimal numbers, V1 and V2.

2. Structure



1) ADD is the symbol for UNSIGNED SINGLE PRECISION DECIMAL ADDITION.

2) ADD requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.4* lists the register reference numbers and constants that can be specified.

Example

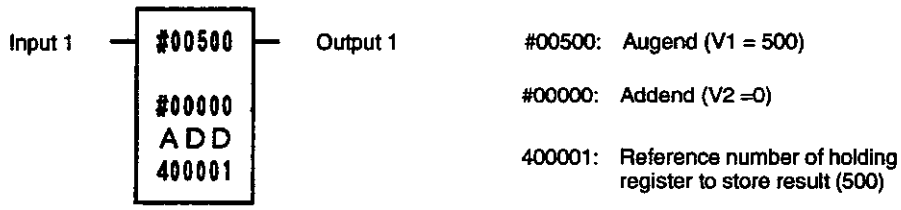


Table 2.4 Structural Elements of ADD

Element	Meaning	Possible Settings	
Top (V1)	Either the value of the constant or the contents of the register is used as the augend, V1. V1 must be between 0 and 9,999.	Constant: Input register: Holding register:	#00000 to #09999 300001 to 300512 (Z00001 to Z00512) 400001 to 409999 (W00001 to W09999)
Middle (V2)	Either the value of the constant or the contents of the register is used as the addend, V2. V2 must be between 0 and 9,999.	Constant register: Link register:	700001 to 704096 (K00001 to K04096) R10001 to R11024 R20001 to R21024
Bottom (R)	The result is stored in the register. The result must be between 0 and 9,999.	Holding register: Link register:	400001 to 409999 (W00001 to W09999) R10001 to R11024 R20001 to R21024

3. Operation

1) ADD adds V2 to V1 when input 1 is ON and process the result as follows:

   a) If $0 \leq V1 + V2 \leq 9,999$:

      (1) The result of V1 + V2 is stored in R.

      (2) Output 1 remains OFF.

   b) If $10,000 \leq V1 + V2 \leq 19,998$:

      (1) The result of V1 + V2 – 10,000 is stored in R.

      (2) Output 1 turns ON.

2) The result remains in R even if input 1 turns OFF.

3) The operation of ADD is summarized in the following table.

Table 2.5 Operation of ADD

Input 1	Condition	Operation	Output 1
ON	$0 \leq$ V1 + V2 $\leq$ 9,999	V1+V2 stored in R.	OFF
	$10,000 \leq$ V1 + V2 $\leq$ 19,998	V1+V2–10,000 stored in R.	ON
OFF	None	Nothing is done.	OFF

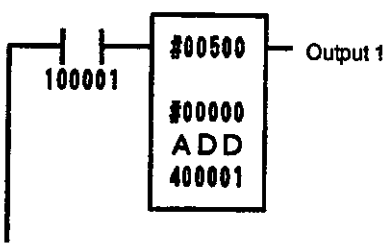
Note Both V1 and V2 must be between 0 and 9,999. ADD will not operate properly if V1 or V2 is not within this range.

◀EXAMPLE▶

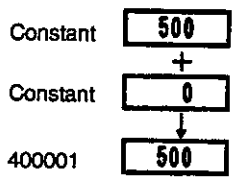
4. Application Examples

Example 1

1) Ladder Programming



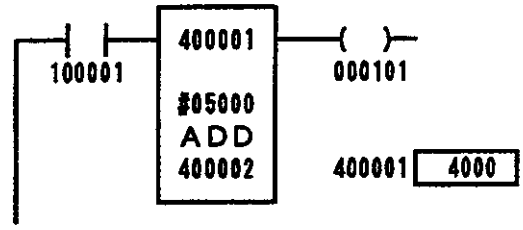
2) Operation



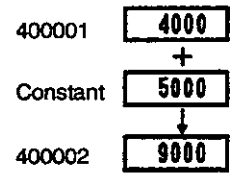
For the above ADD, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will remain OFF. The result will remain in holding register 400001 even after input relay 100001 turns OFF.

Example 2

1) Ladder Programming



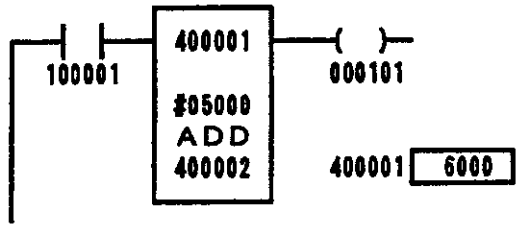
2) Operation



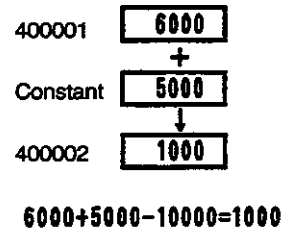
For the above ADD, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will remain OFF. The result will remain in holding register 400002 even after input relay 100001 turns OFF.

Example 3

1) Ladder Programming



2) Operation



6000+5000-10000=1000

For the above ADD, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON. When input relay 100001 turns OFF, coil 000101 will turn OFF, but the result will remain in holding register 400002.

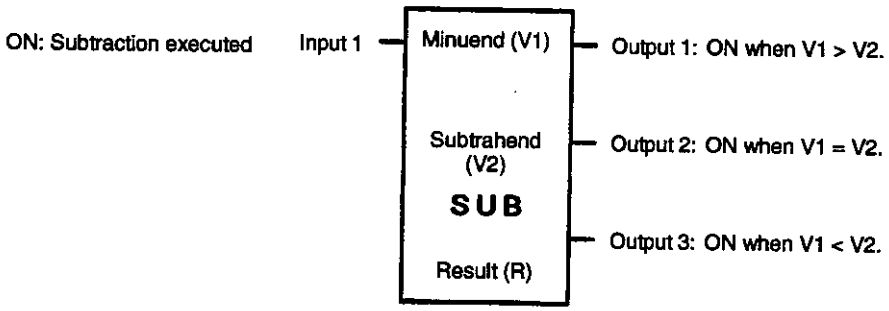
2.3.3 UNSIGNED SINGLE PRECISION DECIMAL SUBTRACTION (SUB)

1. Function

1) Unsigned subtraction is performed between two 4-digit decimal numbers, V1 and V2.

2) The sizes of V1 and V2 are compared.

2. Structure



1) SUB is the symbol for UNSIGNED SINGLE PRECISION DECIMAL SUBTRACTION.

2) SUB requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.6* lists the register reference numbers and constants that can be specified.

Example

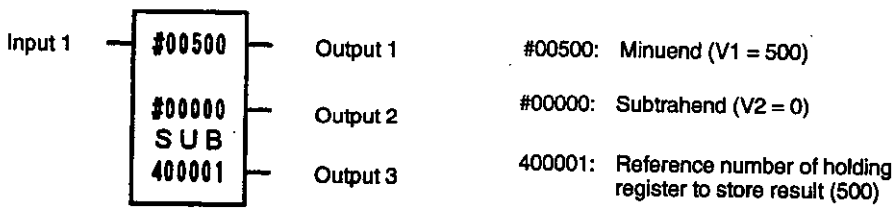


Table 2.6 Structural Elements of SUB

Element	Meaning	Possible Settings	
Top (V1)	Either the value of the constant or the contents of the register is used as the minuend, V1. V1 must be between 0 and 9,999.	Constant: Input register: Holding register: Constant register: Link register:	#00000 to #09999 300001 to 300512 (Z00001 to Z00512) 400001 to 409999 (W00001 to W09999) 700001 to 704096 (K00001 to K04096) R10001 to R11024 R20001 to R21024
Middle (V2)	Either the value of the constant or the contents of the register is used as the subtrahend, V2. V2 must be between 0 and 9,999.		
Bottom (R)	The result is stored in the register. The result must be between 0 and 9,999.	Holding register: Link register:	400001 to 409999 (W00001 to W09999) R10001 to R11024 R20001 to R21024

3. Operation

1) SUB will subtract V2 from V1 when input 1 is ON and process the result as follows:

   a) If V1 > V2:

      (1) The result of V1 – V2 is stored in R.

      (2) Output 1 turns ON.

   b) If V1 = V2:

      (1) Zero (0) is stored in R.

      (2) Output 2 turns ON.

   c) If V1 < V2:

      (1) The result of V2 – V1 is stored in R.

      (2) Output 3 turns ON.

2) The result remains in R even if input 1 turns OFF.

3) The operation of SUB is summarized in the following table.

Table 2.7 Operation of SUB

Input 1	Condition	Operation	Outputs		
			1	2	3
ON	V1 > V2	The result of V1 – V2 is stored in R.	**ON**	OFF	OFF
	V1 = V2	Zero (0) is stored in R.	OFF	**ON**	OFF
	V1 < V2	The result of V2 – V1 is stored in R.	OFF	OFF	**ON**
OFF	None	Nothing is done.	OFF	OFF	OFF

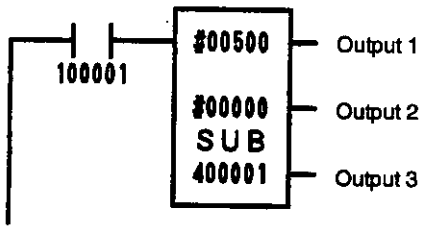
Note Both V1 and V2 must be between 0 and 9,999. SUB will not operate properly if V1 or V2 is not within this range.

◀EXAMPLE▶

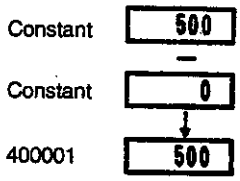
4. Application Examples

Example 1

1) Ladder Programming



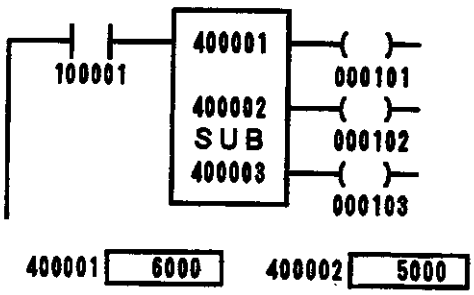
2) Operation



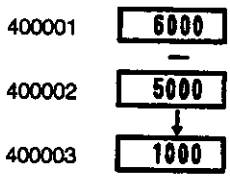
For the above SUB, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. When input relay 100001 turns OFF, output 1 will turn OFF, but the result will remain in holding register 400001.

Example 2

1) Ladder Programming



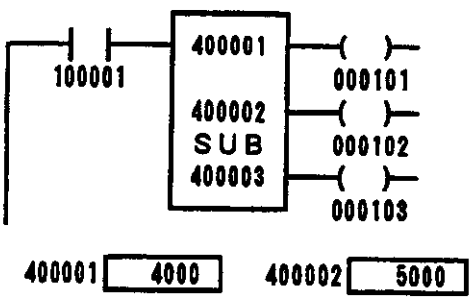
2) Operation



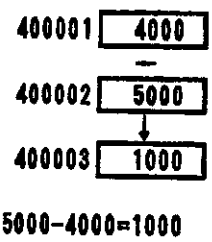
For the above SUB, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON. When input relay 100001 turns OFF, coil 000101 will turn OFF, but the result will remain in holding register 400003.

Example 3

1) Ladder Programming



2) Operation



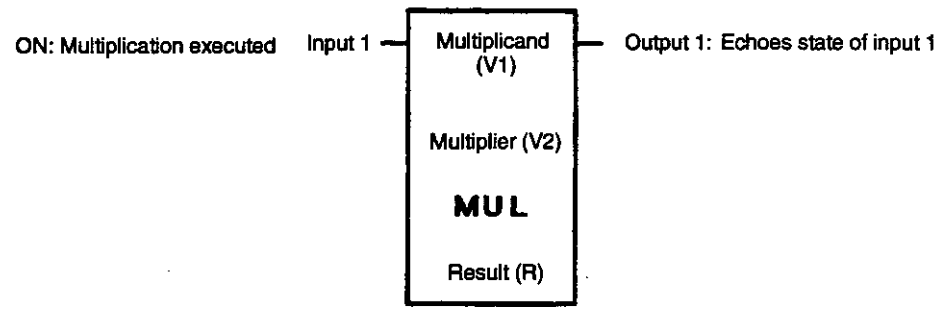
For the above SUB, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000103 will turn ON. When input relay 100001 turns OFF, coil 000103 will turn OFF, but the result will remain in holding register 400003.

2.3.4 UNSIGNED SINGLE PRECISION DECIMAL MULTIPLICATION (MUL)

1. Function

Unsigned multiplication is performed between two 4-digit decimal numbers, V1 and V2.

2. Structure



1) MUL is the symbol for UNSIGNED SINGLE PRECISION DECIMAL MULTIPLICATION.

2) MUL requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.8* lists the register reference numbers and constants that can be specified.

Example

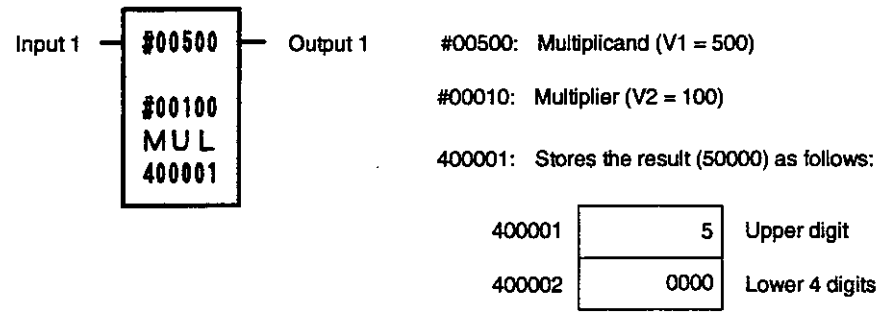


Table 2.8 Structural Elements of MUL

Element	Meaning	Possible Settings
Top (V1)	Either the value of the constant or the contents of the register is used as the multiplicand, V1. V1 must be between 0 and 9,999.	Constant: #00000 to #09999 Input register: 300001 to 300512 (Z00001 to Z00512) Holding register: 400001 to 409999 (W00001 to W09999) Constant register: 700001 to 704096 (K00001 to K04096) Link register: R10001 to R11024 R20001 to R21024
Middle (V2)	Either the value of the constant or the contents of the register is used as the multiplier, V2. V2 must be between 0 and 9,999.	
Bottom (R)	The result is stored in registers as shown below. The result must be between 0 and 99,980,001. R: Upper 4 digits of result R+1: Lower 4 digits of result	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 R20001 to R21023

3. Operation

1) MUL will multiply V1 by V2 when input 1 is ON and process the result as follows:

   a) The upper 4 digits of the result are stored in R and the lower 4 digits are stored in R+1.

   b) Output 1 turns ON.

2) The result remains in R and R+1 even if input 1 turns OFF.

3) The operation of MUL is summarized in the following table.

Table 2.9 Operation of MUL

Input 1	Condition	Operation	Output 1
ON	None	Result of V1 x V2 is stored as shown at right. R: Upper 4 digits of V1 x V2 R+1: Lower 4 digits of V1 x V2	ON
OFF		Nothing is done.	OFF

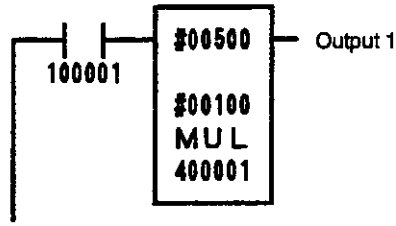
Note Both V1 and V2 must be between 0 and 9,999. MUL will not operate properly if V1 or V2 is not within this range.

◀EXAMPLE▶

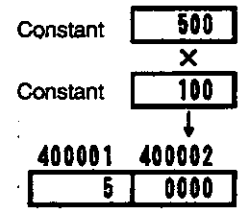
4. Application Examples

Example 1

1) Ladder Programming



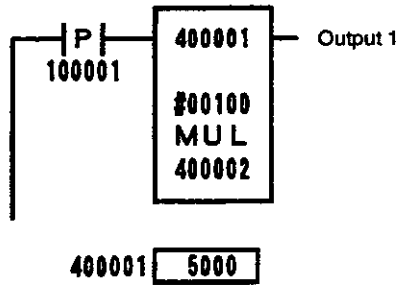
2) Operation



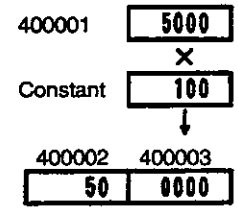
For the above MUL, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. When input relay 100001 turns OFF, output 1 will turn OFF, but the result will remain in holding registers 400001 and 400002.

Example 2

1) Ladder Programming



2) Operation



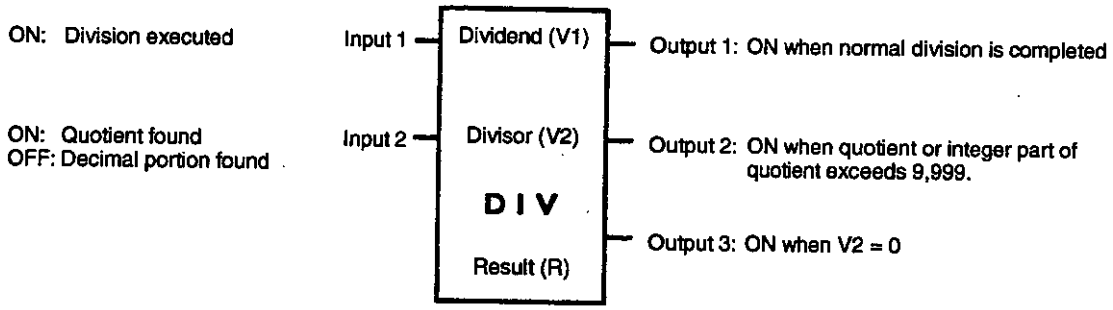
For the above MUL, the operation shown at the right will be performed once during the scan when input relay 100001 goes from OFF to ON, and output 1 will turn ON. When input relay 100001 turns OFF, output 1 will turn OFF, but the result will remain in holding registers 400002 and 400003.

2.3.5 UNSIGNED SINGLE PRECISION DECIMAL DIVISION (DIV)

1. Function

Unsigned division is performed between a 4-digit or 8-digit decimal number V1 and a 4-digit decimal number V2 (V1 ÷ V2).

2. Structure



1) DIV is the symbol for UNSIGNED SINGLE PRECISION DECIMAL DIVISION.

2) DIV requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.10* lists the register reference numbers and constants that can be specified.

Example

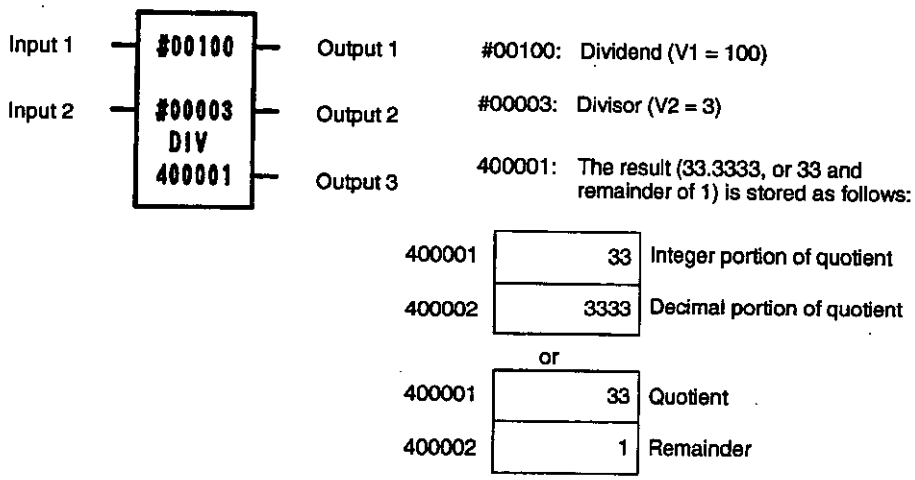


Table 2.10 Structural Elements of DIV

Element	Meaning	Possible Settings
Top (V1)	1) If a constant is specified, its value is used as the dividend, V1. In this case, the value of V1 must be between 0 and 9,999. 2) If a reference number is specified, the contents of two consecutive registers is used as the divisor, V1, as shown in the following example. The value of V1 must be between 0 and 99,989,999. In the example, "400001" was specified for the top element. 400001 Upper 4 digits 400002 Lower 4 digits	Constant: #00000 to #09999 Input register: 300001 to 300511 (Z00001 to Z00511) Holding register: 400001 to 409998 (W00001 to W09998) Constant register: 700001 to 704095 (K00001 to K04095) Link register: R10001 to R11023 R20001 to R21023
Middle (V2)	1) If a constant is specified, its value is used as the dividend, V2. In this case, the value of V2 must be between 0 and 9,999. 2) If a reference number is specified, the contents of the register is used as the divisor, V2. In this case, the value of V2 must also be between 0 and 9,999.	Constant: #00000 to #09999 Input register: 300001 to 300512 (Z00001 to Z00512) Holding register: 400001 to 409999 (W00001 to W09999) Constant register: 700001 to 704096 (K00001 to K04096) Link register: R10001 to R11024 R20001 to R21024
Bottom (R)	The result is stored in registers as shown below. The result must be between 0 and 9,999. 1) **Input 2 OFF** R Quotient R+1 Remainder 2) **Input 2 ON** R Integer portion of quotient R+1 Decimal portion of quotient	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 R20001 to R21023

3. Operation

1) DIV will multiply V1 by V2 when input 1 is ON and process the result as follows:

a) If input 2 is OFF:

(1) The quotient of V1 ÷ V2 is stored in R and remainder is stored in R+1.

(2) Output 1 turns ON.

b) If input 2 is ON:

(1) The integer portion of the quotient of V1 ÷ V2 is stored in R and the decimal portion is stored in R+1.

(2) Output 1 turns ON.

c) Division will not be executed in the following cases and zero (0) is stored in R and R+1.

(1) V2= 0.
In this case, output 3 turns ON.

(2) If the quotient or integer portion of the quotient will not fit in R.
In this case, output 2 turns ON.
Example: If V1 = 500,000 and V2 = 10, the quotient is 50,000, which cannot be stored in R.
Here, 0 is stored in R and R+1.

2) The result remains in R and R+1 even if input 1 turns OFF.

3) The operation of DIV is summarized in the following two tables.

Table 2.11 Operation of DIV with Constant for Top Element

Inputs		Condition	Operation	Outputs		
1	2			1	2	3
ON	OFF	V2 ≠ 0	Result of V1 ÷ V2 stored as follows: R: Quotient R+1: Remainder	**ON**	OFF	OFF
		V2 = 0	Execution not possible. Zero (0) stored in R and R+1.	OFF	OFF	**ON**
ON	**ON**	V2 ≠ 0	Result of V1 ÷ V2 stored as follows: R: Integer portion R+1: Decimal portion Truncated after 4th decimal place	**ON**	OFF	OFF
		V2 = 0	Execution not possible. Zero (0) stored in R and R+1.	OFF	OFF	**ON**
OFF	Any	None	Nothing is done.	OFF	OFF	OFF

Note Both V1 and V2 must be between 0 and 9,999. DIV will not operate properly if V1 or V2 is not within this range.

Table 2.12 Operation of DIV with Reference Number for Top Element

Inputs		Condition	Operation	Outputs		
1	2			1	2	3
ON	OFF	V2 ≠ 0 and $V1_H < V2$*1	Result of V1 ÷ V2 stored as follows: R: Quotient R+1: Remainder	**ON**	OFF	OFF
		V2 ≠ 0 and $V1_H \geq V2$*1	1) Correct division is not possible. 2) Zero (0) is stored in R and R+1.	OFF	**ON**	OFF
		V2 = 0	1) Division is not possible. 2) Zero (0) is stored in R and R+1.	OFF	OFF	**ON**
ON	**ON**	V2 ≠ 0 and $V1_H < V2$*1	Result of V1 ÷ V2 stored as follows (decimal portion is truncated after 4th decimal place): R: Integer portion R+1: Decimal portion	**ON**	OFF	OFF
		V2 ≠ 0 and $V1_H \geq V2$*1	1) Correct division is not possible. 2) Zero (0) is stored in R and R+1.	OFF	**ON**	OFF
		V2 = 0	1) Division is not possible. 2) Zero (0) is stored in R and R+1.	OFF	OFF	**ON**
OFF	Any	None	Nothing is done.	OFF	OFF	OFF

*1: $V1_H$ is the upper 4 digits of V1.

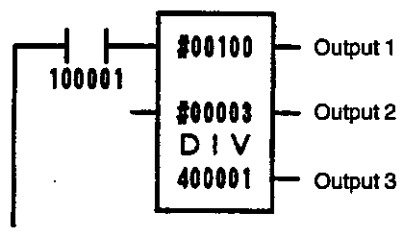
Note Both V1 and V2 must be between 0 and 9,999. DIV will not operate properly if V1 or V2 is not within this range.

◀EXAMPLE▶

4. Application Examples

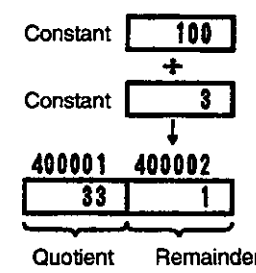
Example 1

1) Ladder Programming



100 ÷ 3 = 33 with remainder of 1

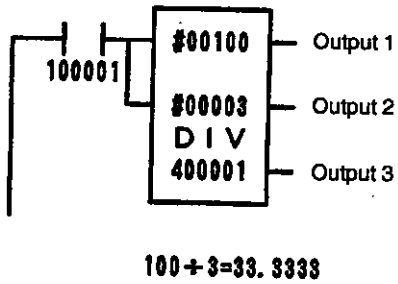
2) Operation



For the above DIV, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. When input relay 100001 turns OFF, output 1 will turn OFF, but the result will remain in holding registers 400001 and 400002.

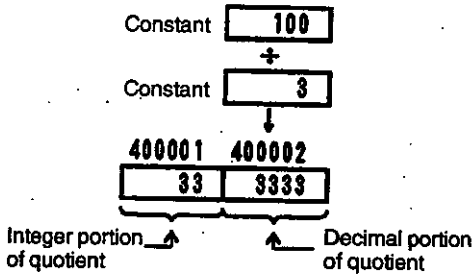
Example 2

1) Ladder Programming



100÷3=33. 3333

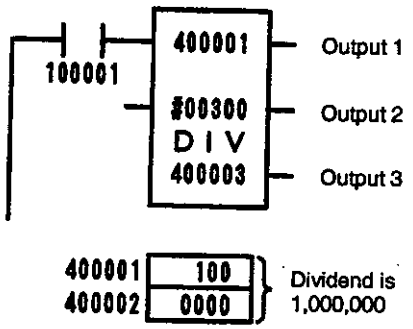
2) Operation



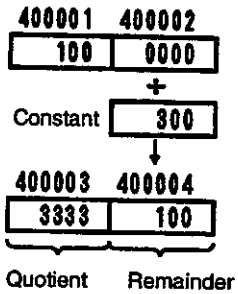
For the above DIV, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. When input relay 100001 turns OFF, output 1 will turn OFF, but the result will remain in holding registers 400001 and 400002.

Example 3

1) Ladder Programming



2) Operation

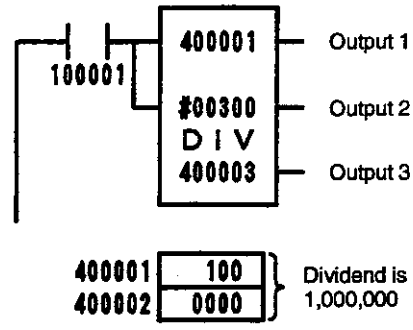


1,000,000 ÷ 300 = 3,333 with remainder of 100

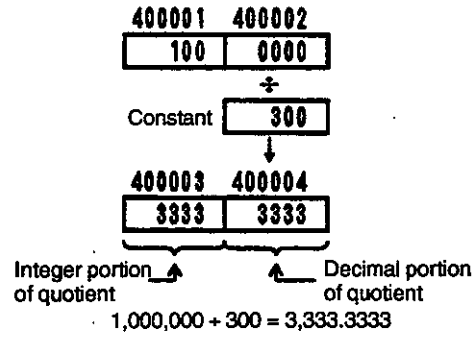
For the above DIV, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. When input relay 100001 turns OFF, output 1 will turn OFF, but the result will remain in holding registers 400003 and 400004.

Example 4

1) Ladder Programming



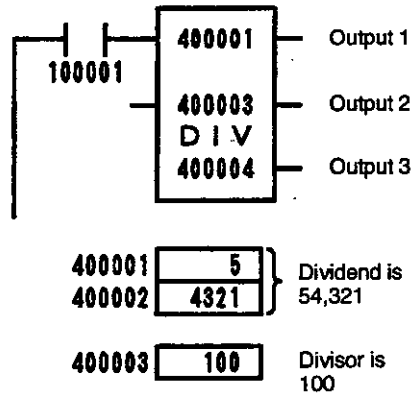
2) Operation



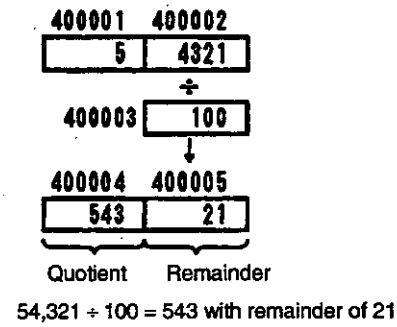
For the above DIV, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. When input relay 100001 turns OFF, output 1 will turn OFF, but the result will remain in holding registers 400003 and 400004.

Example 5

1) Ladder Programming



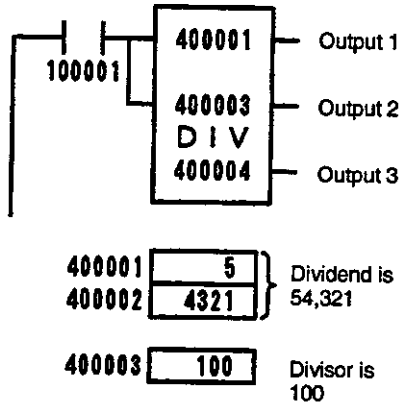
2) Operation



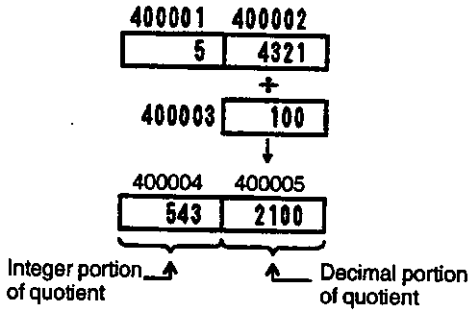
For the above DIV, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. When input relay 100001 turns OFF, output 1 will turn OFF, but the result will remain in holding registers 400004 and 400005.

Example 6

1) Ladder Programming



2) Operation



54,321 ÷ 100 = 543.2100

For the above DIV, the operation shown at the right will be performed when input relay 100002 is ON, and output 1 will turn ON. When input relay 100001 turns OFF, output 1 will turn OFF, but the result will remain in holding registers 400004 and 400005.

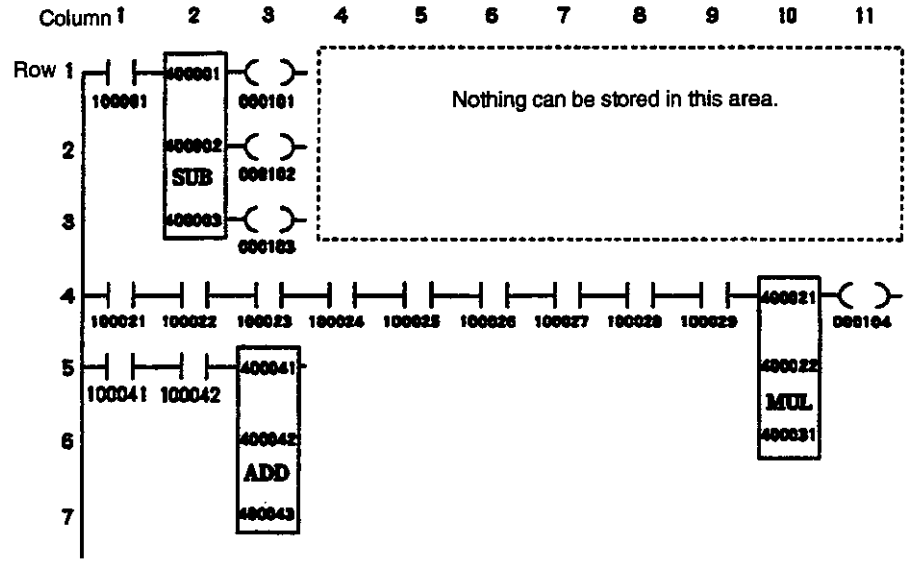
2.3.6 Building Programs

1. Storage Locations on Networks

All unsigned, 4-digit, decimal arithmetic instructions require three vertical elements on a network, one top element, one middle element, and one bottom element. They can thus be stored anywhere on a 5-row by 10-column matrix (rows 1 through 5 and columns 1 through 10) on the network.

Note Unsigned, 4-digit, decimal arithmetic instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

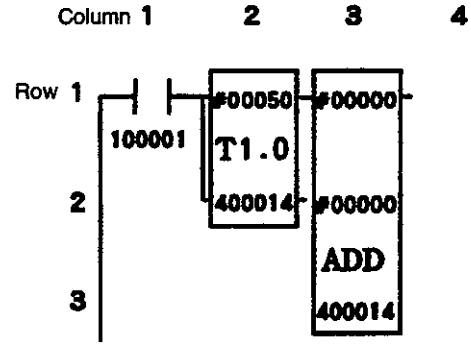
Example



2. Inputs

Inputs to unsigned, 4-digit, decimal math instruction can be connected to relay elements (except coils) and/or outputs from timers, counters, math instruction, data transfer instructions, other instructions, etc.

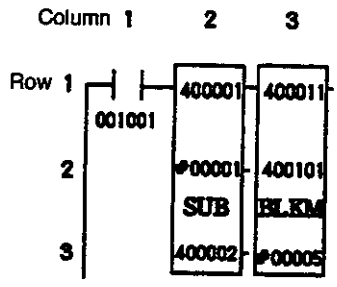
Example



3. Outputs

Outputs from unsigned, 4-digit, decimal math instruction can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data transfer instructions, etc.

Example



2.4 Unsigned, Eight-digit, Decimal Arithmetic Instructions

This section describes the functions, structures, and operation of the unsigned, 8-digit, decimal arithmetic instructions and provides simple examples of their application.

2.4.1 Instructions

Unsigned, 8-digit, decimal arithmetic instructions perform unsigned addition, subtraction, multiplication, and division on two 8-digit, decimal values, V1 and V2. The instructions that are available are shown in *Table 2.13.*

Table 2.13 Unsigned, 8-digit, Decimal Arithmetic Instructions

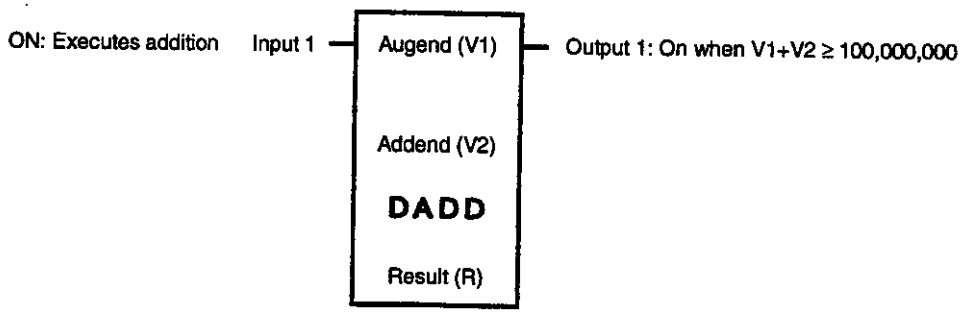
Name	Symbol	Operation	V1	V2	Result
UNSIGNED DOUBLE PRECISION DECIMAL ADDITION	DADD	V1 + V2	0 to 99,999,999		
UNSIGNED DOUBLE PRECISION DECIMAL SUBTRACTION	DSUB	V1 – V2 Comparison	0 to 99,999,999		
UNSIGNED DOUBLE PRECISION DECIMAL MULTIPLICATION	DMUL	V1 × V2	0 to 99,999,999		0 to 9,999,999,800,000,001
UNSIGNED DOUBLE PRECISION DECIMAL DIVISION	DDIV	V1 ÷ V2	0 to 9,999,999,899,999,999	0 to 99,999,999	

2.4.2 UNSIGNED DOUBLE PRECISION DECIMAL ADDITION (DADD)

1. Function

Unsigned addition is performed between two 8-digit decimal numbers, V1 and V2.

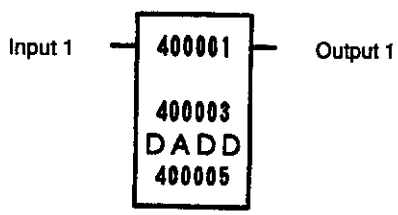
2. Structure



1) DADD is the symbol for UNSIGNED DOUBLE PRECISION DECIMAL ADDITION.

2) DADD requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.14* lists the register reference numbers that can be specified.

Example



400001: The augend, V1, is stored as follows:

400001	Upper 4 digits
400002	Lower 4 digits

400003: The addend, V2, is stored as follows:

400003	Upper 4 digits
400004	Lower 4 digits

400005: The result is stored as follows:

400005	Upper 4 digits of result
400006	Lower 4 digits of result

Table 2.14 Structural Elements of DADD

Element	Meaning	Possible Settings
Top (V1)	The contents of two consecutive registers is used as the augend, V1, as shown in the following example. The value of V1 must be between 0 and 99,999,999. In the example, "400001" was specified for the top element. 400001 Upper 4 digits 400002 Lower 4 digits	Input register: 300001 to 300511 Holding register: 400001 to 409998 (W00001 to W09998) Constant register: 700001 to 704095 (K00001 to K04095) Link register: R10001 to R11023 R20001 to R21023
Middle (V2)	The contents of two consecutive registers is used as the addend, V2 as shown in the following example. The value of V2 must be between 0 and 99,999,999. In the example, "400003" was specified for the middle element. 400003 Upper 4 digits 400004 Lower 4 digits	
Bottom (R)	The result is stored in registers as shown below. The result must be between 0 and 99,999,999. In the example, "400005" was specified for the bottom element. 400005 Upper 4 digits 400006 Lower 4 digits	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 R20001 to R21023

3. Operation

1) DADD will add V2 to V1 when input 1 is ON and process the result as follows:

   a) If $0 \leq V1+V2 \leq 99,999,999$:

      (1) The upper 4 digits of the result of V1+V2 are stored in R and the lower 4 digits are stored in R+1.

      (2) Output 1 remains OFF.

   b) If $100,000,000 \leq V1+V2 \leq 199,999,998$:

      (1) The upper 4 digits of the result of V1+V2−100,000,000 are stored in R and the lower 4 digits are stored in R+1.

      (2) Output 1 turns ON.

2) The result R and R+1 remains even if input 1 turns OFF.

3) The operation of DADD is summarized in the following table.

Table 2.15 Operation of DADD

Input 1	Condition	Operation	Output 1
ON	0 ≤ V1 + V2 ≤ 99,999,999	The result of V1+V2 is stored as follows: R: Upper 4 digits R+1: Lower 4 digits	OFF
	100,000,000 ≤ V1 + V2 ≤ 199,999,998	The result of V1+V2−100,000,000 is stored as follows: R: Upper 4 digits R+1: Lower 4 digits	ON
OFF	None	Nothing is done.	OFF

Note Both V1 and V2 must be between 0 and 99,999,999. DADD will not operate properly if V1 or V2 is not within this range.

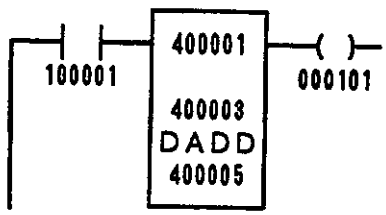
2

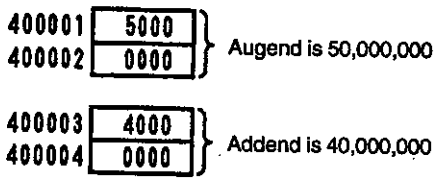
◀EXAMPLE▶

4. Application Examples

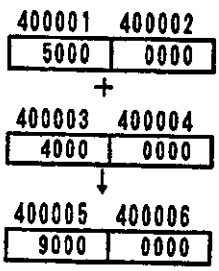
Example 1

1) Ladder Programming





2) Operation

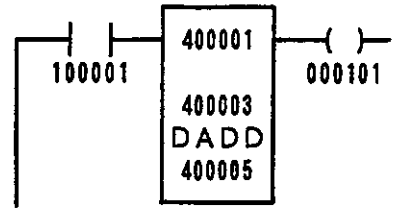


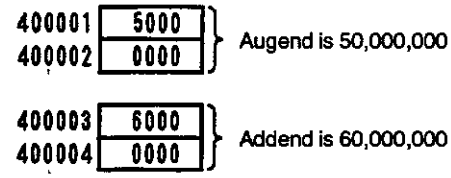
(50,000,000 + 40,000,000) = 90,000,000

For the above DADD, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will remain OFF. The result will remain in holding registers 400005 and 400006 even after input relay 100001 turns OFF.

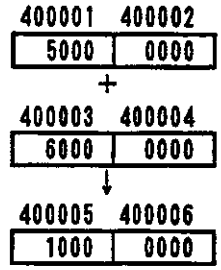
Example 2

1) Ladder Programming





2) Operation



(50, 000, 000+60, 000, 000-100, 000, 000)
=10, 000, 000

For the above DADD, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON. When input relay 100001 turns OFF, coil 000101 will turn OFF, and the result will remain in holding registers 400005 and 400006.

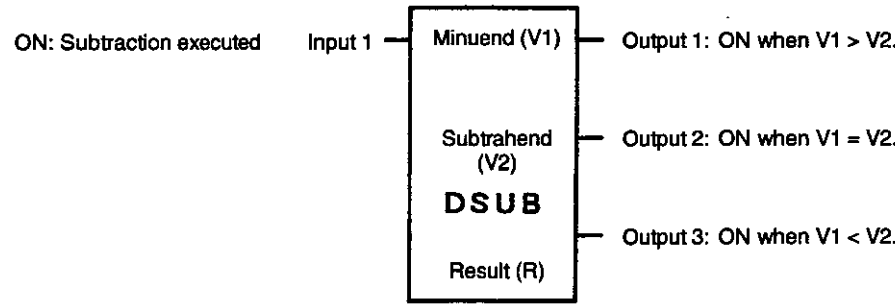
2.4.3 UNSIGNED DOUBLE PRECISION DECIMAL SUBTRACTION (DSUB)

1. Function

1) Unsigned subtraction is performed between two 8-digit decimal numbers, V1 and V2.

2) The sizes of V1 and V2 are compared.

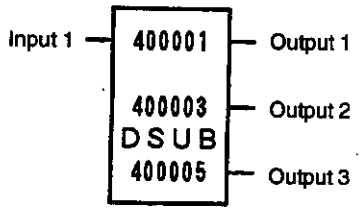
2. Structure



1) DSUB is the symbol for UNSIGNED DOUBLE PRECISION DECIMAL SUBTRACTION.

2) DSUB requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.16* lists the register reference numbers that can be specified.

Example



400001: The minuend, V1, is stored as follows:

400001	Upper 4 digits
400002	Lower 4 digits

400003: The subtrahend, V2, is stored as follows:

400003	Upper 4 digits
400004	Lower 4 digits

400005: The result is stored as follows:

400005	Upper 4 digits of result
400006	Lower 4 digits of result

Table 2.16 Structural Elements of DSUB

Element	Meaning	Possible Settings
Top (V1)	The contents of two consecutive registers is used as the minuend, V1, as shown in the following example. The value of V1 must be between 0 and 99,999,999. In the example, "400001" was specified for the top element. 400001 Upper 4 digits 400002 Lower 4 digits	Input register: 300001 to 300511 (Z00001 to Z00511) Holding register: 400001 to 409998 (W00001 to W09998) Constant register: 700001 to 704095 (K00001 to K04095) Link register: R10001 to R11023, R20001 to R21023
Middle (V2)	The contents of two consecutive registers is used as the subtrahend, V2, as shown in the following example. The value of V2 must be between 0 and 99,999,999. In the example, "400003" was specified for the middle element. 400003 Upper 4 digits 400004 Lower 4 digits	
Bottom (R)	The result is stored in registers as shown below. The result must be between 0 and 99,999,999. In the example, "400005" was specified for the bottom element. 400005 Upper 4 digits 400006 Lower 4 digits	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023, R20001 to R21023

3. Operation

1) DSUB will subtract V2 from V1 when input 1 is ON and process the result as follows:

   a) If V1 > V2:

      (1) The upper 4 digits of the result of V1–V2 are stored in R and the lower 4 digits are stored in R+1.

      (2) Output 1 turns ON.

   b) If V1 = V2:

      (1) Zero (0) is stored in R and R+1.

      (2) Output 2 turns ON.

   c) If V1 < V2:

      (1) The upper 4 digits of the result of V2–V1 are stored in R and the lower 4 digits are stored in R+1.

      (2) Output 3 turns ON.

2) The result remains in R and R+1 even if input 1 turns OFF.

3) The operation of DSUB is summarized in the following table.

Table 2.17 Operation of DSUB

Input 1	Condition	Operation	Outputs		
			1	2	3
ON	V1 > V2	The result of V1–V2 is stored as follows: R: Upper 4 digits R+1: Lower 4 digits	**ON**	OFF	OFF
	V1 = V2	Zero (0) is stored in R and R+1.	OFF	**ON**	OFF
	V1 < V2	The result of V2–V1 is stored as follows: R: Upper 4 digits R+1: Lower 4 digits	OFF	OFF	**ON**
OFF	None	Nothing is done.	OFF	OFF	OFF

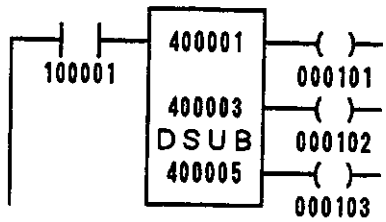
Note Both V1 and V2 must be between 0 and 99,999,999. DSUB will not operate properly if V1 or V2 is not within this range.

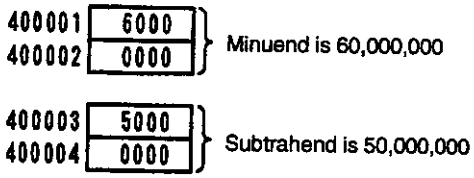
◀EXAMPLE▶

4. Application Examples

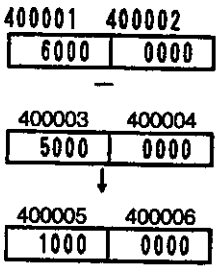
Example 1

1) Ladder Programming





2) Operation

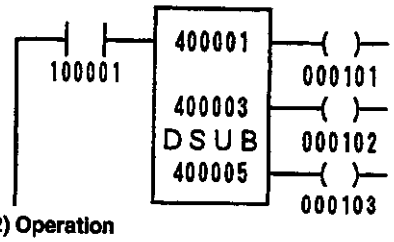


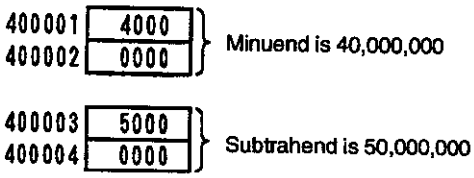
(60, 000, 000-50, 000, 000) = 10, 000, 000

For the above DSUB, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON. When input relay 100001 turns OFF, coil 000101 will turn OFF and the result will remain in holding registers 400005 and 400006.

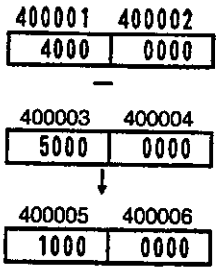
Example 2

1) Ladder Programming





2) Operation



(50, 000, 000-40, 000, 000) = 10, 000, 000

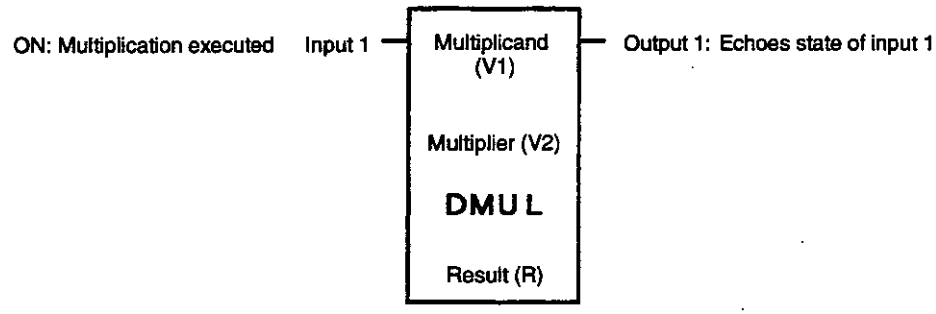
For the above DSUB, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000103 will turn ON. When input relay 100001 turns OFF, coil 000103 will turn OFF and the result will remain in holding registers 400005 and 400006.

2.4.4 UNSIGNED DOUBLE PRECISION DECIMAL MULTIPLICATION (DMUL)

1. Function

Unsigned multiplication is performed between two 8-digit decimal numbers, V1 and V2.

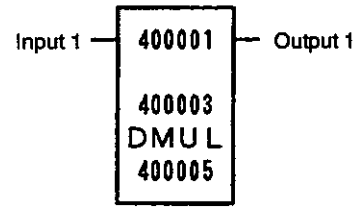
2. Structure



1) DMUL is the symbol for UNSIGNED DOUBLE PRECISION DECIMAL MULTIPLICATION.

2) DMUL requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.18* lists the register reference numbers that can be specified.

Example



400001: The multiplicand, V1, is stored as follows:

400001	Upper 4 digits
400002	Lower 4 digits

400003: The multiplier, V2, is stored as follows:

400003	Upper 4 digits
400004	Lower 4 digits

400005: The result is stored as follows:

400005	Most upper 4 digits
400006	Upper 4 digits
400007	Lower 4 digits
400008	Most lower 4 digits

Table 2.18 Structural Elements of DMUL

Element	Meaning	Possible Settings
Top (V1)	The contents of two consecutive registers is used as the multiplicand, V1, as shown in the following example. The value of V1 must be between 0 and 99,999,999. In the example, "400001" was specified for the top element. 400001 Upper 4 digits 400002 Lower 4 digits	Input register: 300001 to 300511 (Z00001 to Z00511) Holding register: 400001 to 409998 (W00001 to W09998) Constant register: 700001 to 704095 (K00001 to K04095) Link register: R10001 to R11023 R20001 to R21023
Middle (V2)	The contents of two consecutive registers is used as the multiplier, V2, as shown in the following example. The value of V2 must be between 0 and 99,999,999. In the example, "400003" was specified for the middle element. 400003 Upper 4 digits 400004 Lower 4 digits	
Bottom (R)	The result is stored in registers as shown below. The result must be between 0 and 9,999,999,800,000,001. In the example, "400005" was specified for the bottom element. 400005 Most upper 4 digits 400006 Upper 4 digits 400007 Lower 4 digits 400008 Most lower 4 digits	Holding register: 400001 to 409996 (W00001 to W09996) Link register: R10001 to R11021 R20001 to R21021

3. Operation

1) DMUL will multiply V1 by V2 when input 1 is ON and process the result as follows:

   a) The most upper 4 digits of the result are stored in R; the upper 4 digits, in R+1; the lower 4 digits, in R+2, and the most lower 4 digits, in R+3.

   b) Output 1 turns ON.

2) The result remains in R though R+3 even if input 1 turns OFF.

3) The operation of DMUL is summarized in the following table.

Table 2.19 Operation of DMUL

Input 1	Condition	Operation		Output 1
ON	None	Result of V1xV2 is stored as shown at right. The result will be a positive 16-digit decimal integer or 0.	R: Most upper 4 digits R+1: Upper 4 digits R+2: Lower 4 digits R+3: Most lower 4 digits	ON
OFF		Nothing is done.		OFF

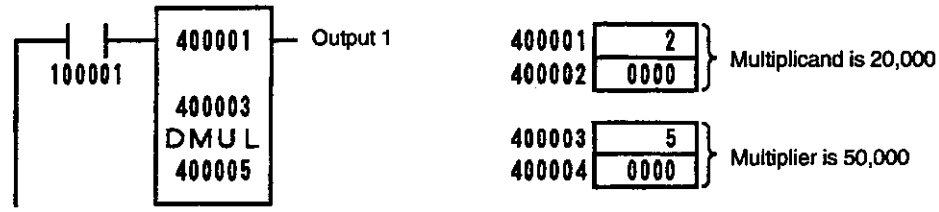
Note Both V1 and V2 must be between 0 and 99,999,999. DMUL will not operate properly if V1 or V2 is not within this range.

◀EXAMPLE▶

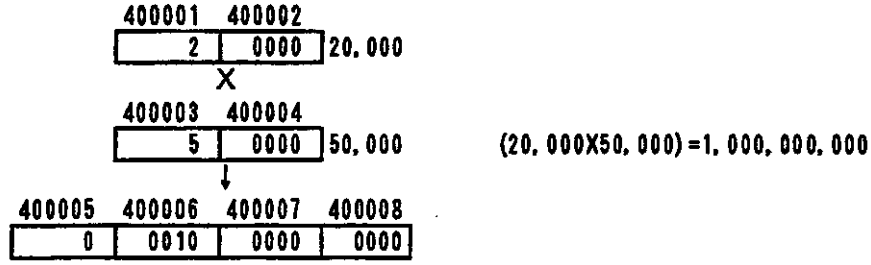
4. Application Example

Example

1) Ladder Programming



2) Operation



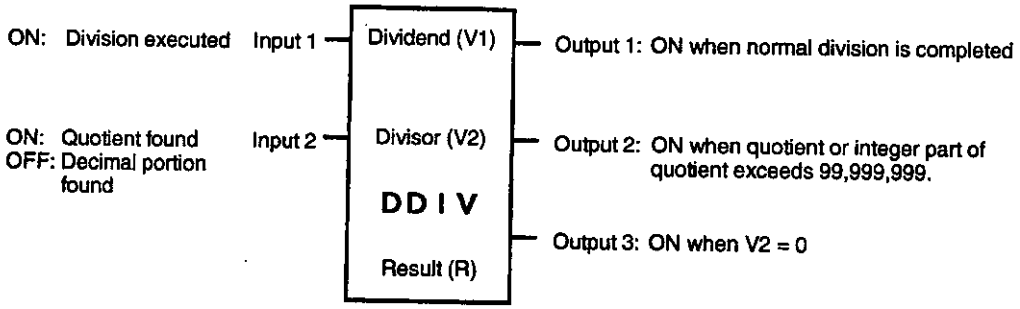
For the above DMUL, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. When input relay 100001 turns OFF, output 1 will turn OFF and the result will remain in holding registers 400005 through 400008.

2.4.5 UNSIGNED DOUBLE PRECISION DECIMAL DIVISION (DDIV)

1. Function

Unsigned division is performed between a 16-digit decimal number V1 and a 8-digit decimal number V2 (V1 ÷ V2).

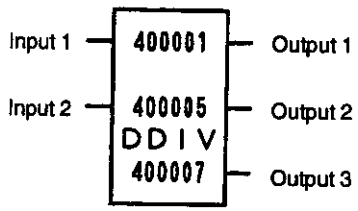
2. Structure



1) DDIV is the symbol for UNSIGNED DOUBLE PRECISION DECIMAL DIVISION.

2) DDIV requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.20* lists the register reference numbers that can be specified.

Example



400001: Dividend, V1, is stored as follows:

400001	Most upper 4 digits
400002	Upper 4 digits
400003	Lower 4 digits
400004	Most lower 4 digits

400005: Divisor, V2, is stored as follows:

400005	Upper 4 digits
400006	Lower 4 digits

400007: The result is stored as follows:

400007	Upper 4 digits of quotient
400008	Lower 4 digits or quotient
400009	Upper 4 digits of remainder
400010	Lower 4 digits of remainder

or

400007	Upper 4 digits of integer portion
400008	Lower 4 digits of integer portion
400009	Upper 4 digits of decimal portion
400010	Lower 4 digits of decimal portion

Table 2.20 Structural Elements of DDIV

Element	Meaning	Possible Settings
Top (V1)	The contents of four consecutive registers is used as the dividend, V1, as shown in the following example. The value of V1 must be between 0 and 9,999,999,899,999,999. In the example, "400001" was specified for the top element. 400001 Most upper 4 digits 400002 Upper 4 digits 400003 Lower 4 digits 400004 Most lower 4 digits	Input register: 300001 to 300509 (Z00001 to Z00509) Holding register: 400001 to 409996 (W00001 to W09996) Constant register: 700001 to 704093 (K00001 to K04093) Link register: R10001 to R11021 R20001 to R21021
Middle (V2)	The contents of two consecutive registers is used as the divisor, V2, as shown in the following example. The value of V2 must be between 0 and 99,999,999. In the example, "400005" was specified for the middle element. 400005 Upper 4 digits 400006 Lower 4 digits	Input register: 300001 to 300511 (Z00001 to Z00511) Holding register: 400001 to 409998 (W00001 to W09998) Constant register: 700001 to 704095 (K00001 to K04095) Link register: R10001 to R11023 R20001 to R21023
Bottom (R)	The result is stored in registers as shown below. The result must be between 0 and 99,999,999. In the example, "400007" was specified for the bottom element. **Input 2 OFF** 400007 Upper 4 digits of quotient 400008 Lower 4 digits of quotient 400009 Upper 4 digits of remainder 400010 Lower 4 digits of remainder **Input 2 ON** 400007 Upper 4 digits of integer portion 400008 Lower 4 digits of integer portion 400009 Upper 4 digits of decimal portion 400010 Lower 4 digits of decimal portion	Holding register: 400001 to 409996 (W00001 to W09996) Link register: R10001 to R11021 R20001 to R21021

3. Operation

1) DDIV will divide V1 by V2 when input 1 is ON and process the result as follows:

   a) If input 2 is OFF:

      (1) The upper 4 digits of the quotient are stored in R and the lower 4 digits are stored in R+1.

      (2) The upper 4 digits of the remainder are stored in R+2 and the lower 4 digits are stored in R+3.

(3) Output 1 turns ON.

b) If input 2 is ON:

(1) The upper 4 digits of the integer portion of the quotient of V1 ÷ V2 are stored in R and lower 4 digits are stored in R+1.

(2) The upper 4 digits of the decimal portion (truncated after the 8th decimal place) of the quotient of V1 ÷ V2 are stored in R+2 and lower 4 digits are stored in R+3.

(3) Output 1 turns ON.

c) Division will not be executed in the following cases and zero (0) is stored in R through R+3.

(1) V2 = 0. In this case, output 3 turns ON.

(2) If the quotient or integer portion of the quotient will not fit in R and R+1. In this case, output 2 turns ON.
Example: If V1 = 5,000,000,000 and V2 = 10, the quotient is 500,000,000, which cannot be stored in R and R+1. Here, 0 is stored in R through R+3.

2) The result remains in R through R+3 even if input 1 turns OFF.

3) The operation of DDIV is summarized in the following table.

Table 2.21 Operation of DDIV

Inputs		Condition	Operation	Outputs		
1	2			1	2	3
ON	OFF	$V2 \neq 0$ and $V1_H < V2$	The following calculation is performed, where the upper 8 digits of V1 are $V1_H$ and the lower 8 digits are $V1_L$: $(V1_H \times 10^8 + V1_L) \div V2$ R: Upper 4 digits of quotient R+1: Lower 4 digits of quotient R+2: Upper 4 digits of remainder R+3: Lower 4 digits of remainder	**ON**	OFF	OFF
		$V2 \neq 0$ and $V1_H \geq V2$	1) Correct division is not possible. 2) Zero (0) is stored in R through R+3.	OFF	**ON**	OFF
		V2 = 0	1) Division is not possible. 2) Zero (0) is stored in R through R+3.	OFF	OFF	**ON**
ON	**ON**	$V2 \neq 0$ and $V1_H < V2$	The following calculation is performed, where the upper 8 digits of V1 are $V1_H$ and the lower 8 digits are $V1_L$: $(V1_H \times 10^8 + V1_L) \div V2$ The decimal portion is truncated after 8th decimal place. R: Upper 4 digits of integer portion of quotient R+1: Lower 4 digits of integer portion of quotient R+2: Upper 4 digits of decimal portion of quotient R+3: Lower 4 digits of decimal portion of quotient	**ON**	OFF	OFF
		$V2 \neq 0$ and $V1_H \geq V2$	1) Correct division is not possible. 2) Zero (0) is stored in R through R+3.	OFF	**ON**	OFF
		V2 = 0	1) Division is not possible. 2) Zero (0) is stored in R through R+3.	OFF	OFF	**ON**
OFF	Any	None	Nothing is done.	OFF	OFF	OFF

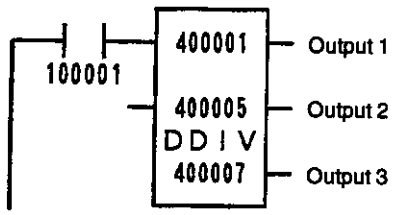
Note Both V1 must be between 0 and 9,999,999,899,999,999 and V2 must be between 0 and 99,999,999. DDIV will not operate properly if V1 or V2 is not within its specified range.

◀EXAMPLE▶

4. Application Examples

Example 1

1) Ladder Programming



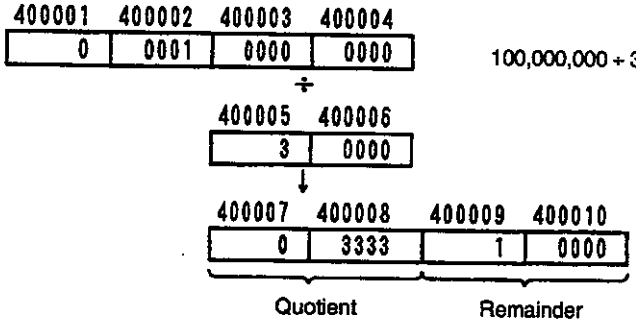
400001	0
400002	1
400003	0000
400004	0000

Dividend is 100,000,000

400005	3
400006	0000

Divisor is 30,000

2) Operation

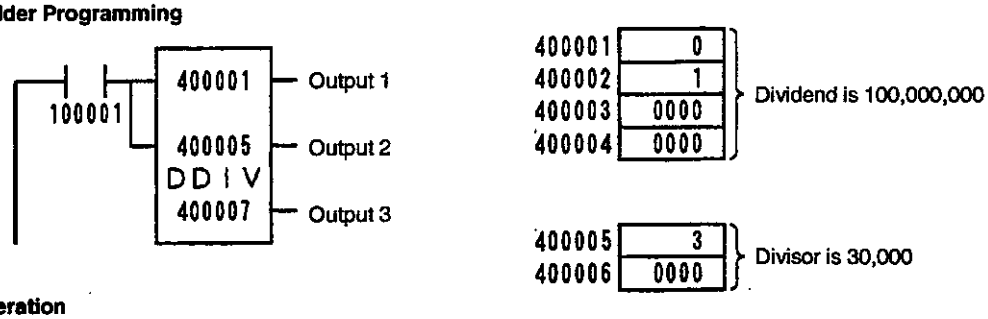


100,000,000 ÷ 30,000 = 3,333 with remainder of 10,000

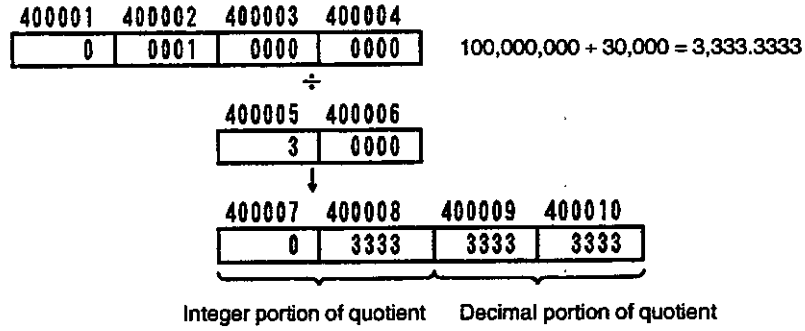
For the above DDIV, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. When input relay 100001 turns OFF, output 1 will turn OFF and the result will remain in holding registers 400007 through 400010.

Example 2

1) Ladder Programming



2) Operation



For the above DDIV, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. When input relay 100001 turns OFF, output 1 will turn OFF and the result will remain in holding registers 400007 through 400010.

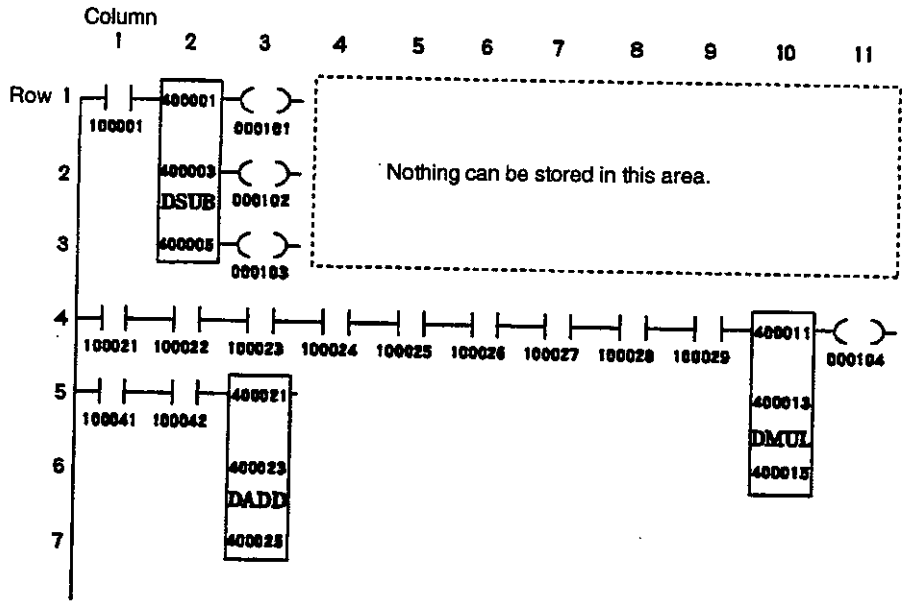
2.4.6 Building Programs

1. Storage Locations on Networks

All unsigned, 8-digit, decimal arithmetic instructions require three vertical elements on a network, one top element, one middle element, and one bottom element. They can thus be stored anywhere on a 5-row by 10-column matrix (rows 1 through 5 and columns 1 through 10) on the network.

Note Unsigned, 4-digit, decimal arithmetic instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

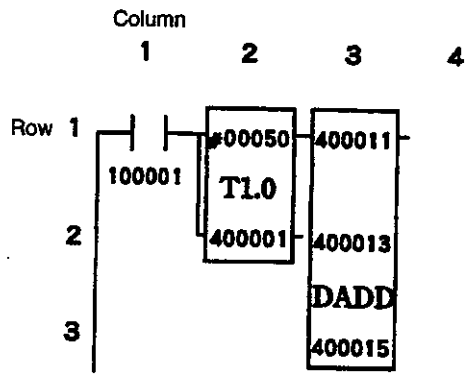
Example



2. Inputs

Inputs to unsigned, 8-digit, decimal math instruction can be connected to relay elements (except coils) and/or outputs from timers, counters, math instructions, data transfer instructions, other instructions, etc.

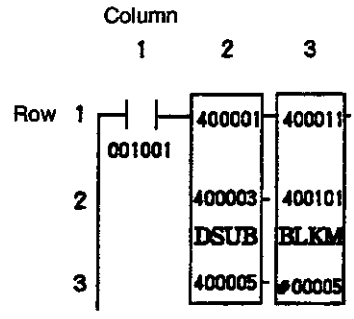
Example



3. Outputs

Outputs from unsigned, 8-digit, decimal math instruction can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data transfer instructions, etc.

Example



2.5 Signed, Four-digit, Decimal Arithmetic Instructions

This section describes the functions, structures, and operation of the signed, 4-digit, decimal arithmetic instructions and provides simple examples of their application.

2.5.1 Instructions

Signed, 4-digit, decimal arithmetic instructions perform signed addition, subtraction, multiplication, and division on two 4-digit, decimal values, V1 and V2. The instructions that are available are shown in *Table 2.22.*

Table 2.22 Signed, 4-digit, Decimal Arithmetic Instructions

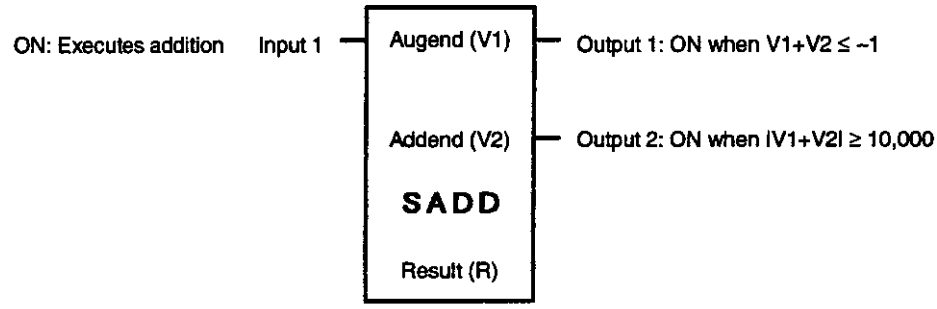
Name	Symbol	Operands	V1	V2	Result
SIGNED SINGLE PRECISION DECIMAL ADDITION	SADD	V1 + V2	–9,999 to 9,999		
SIGNED SINGLE PRECISION DECIMAL SUBTRACTION	SSUB	V1 – V2			
SIGNED SINGLE PRECISION DECIMAL MULTIPLICATION	SMUL	V1 × V2	–9,999 to 9,999		–99,980,001 to 99,980,001
SIGNED SINGLE PRECISION DECIMAL DIVISION	SDIV	V1 ÷ V2	–99,989,999 to 99,989,999	–9,999 to 9,999	

2.5.2 SIGNED SINGLE PRECISION DECIMAL ADDITION (SADD)

1. Function

Signed addition is performed between two 4-digit decimal numbers, V1 and V2.

2. Structure



1) SADD is the symbol for SIGNED SINGLE PRECISION DECIMAL ADDITION.

2) SADD requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.23* lists the register reference numbers that can be specified.

Example

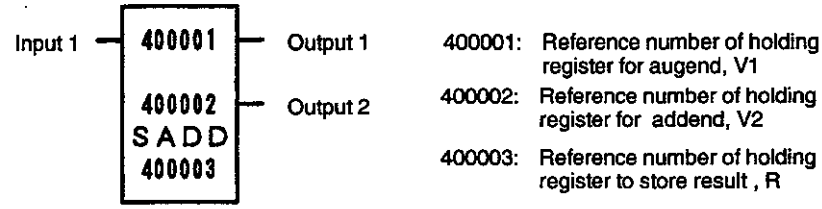


Table 2.23 Structural Elements of SADD

Element	Meaning	Possible Settings
Top (V1)	1) The contents of the specified register is used as the augend, V1. 2) V1 must be between –9,999 and 9,999. 3) Set the MSB of the register to 1 if V1 < 0.	Input register: 300001 to 300512 (Z00001 to Z00512) Holding register: 400001 to 409999 (W00001 to W09999) Constant register: 700001 to 704096 (K00001 to K04096) Link register: R10001 to R11024 R20001 to R21024
Middle (V2)	1) The contents of the specified register is used as the addend, V2. 2) V2 must be between –9,999 and 9,999. 3) Set the MSB of the register to 1 if V2 < 0.	
Bottom (R)	1) The result is stored in the specified register. 2) The result must be between –9,999 and 9,999. 3) The MSB of the register will be set to 1 if the results < 0.	Holding register: 400001 to 409999 (W00001 to W09999) Link register: R10001 to R11024 R20001 to R21024

3. Operation

1) SADD adds V2 to V1 when input 1 is ON and process the result as follows:

a) If $0 \leq V1 + V2 \leq 9,999$:

(1) The result of V1 + V2 is stored in R.

(2) The MSB (most significant bit) of R is set to 0 to indicate a positive result.

(3) Outputs 1 and 2 remain OFF.

b) If $-9,999 \leq V1+V2 \leq -1$:

(1) The result of V1+V2 is stored in R.

(2) The MSB of R is set to 1 to indicate a negative result.

(3) Output 1 turns ON and output 2 remains OFF.

c) If $10,000 \leq V1+V2 \leq 19,998$:

(1) The result of V1+V2 – 10,000 is stored in R.

(2) The MSB of R is set to 0 to indicate a positive result.

(3) Output 1 remains OFF and output 2 turns ON.

d) If $-19,998 \leq V1+V2 \leq -10,000$:

(1) The result of V1+V2 + 10,000 is stored in R.

(2) The MSB of R is set to 1 to indicate a negative result.

(3) Outputs 1 and 2 turn ON .

2) The result remains in R even if input 1 turns OFF.

3) The operation of SADD is summarized in *Table 2.24*.

Table 2.24 Operation of SADD

Input 1	Condition	Operation	Outputs	
			1	2
ON	$0 \leq V1+V2 \leq 9,999$	1) Result of V1+V2 stored in R. 2) MSB of R set to 0.	OFF	OFF
	$-9,999 \leq V1+V2 \leq -1$	1) Result of V1+V2 stored in R. 2) MSB of R set to 1.	**ON**	OFF
	$10,000 \leq V1+V2 \leq 19,998$	1) Result of V1+V2 – 10,000 stored in R. 2) MSB of R set to 0.	OFF	**ON**
	$-19,998 \leq V1+V2 \leq -10,000$	1) Result of V1+V2 + 10,000 stored in R. 2) MSB of R set to 1.	**ON**	**ON**
OFF	None	Nothing is done.	OFF	OFF

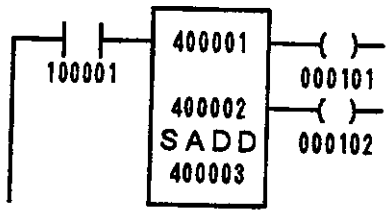
Note Both V1 and V2 must be between –9,999 and 9,999. SADD will not operate properly if V1 or V2 is not within this range.

◀EXAMPLE▶

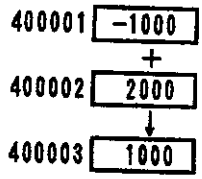
4. Application Examples

Example 1

1) Ladder Programming



2) Operation

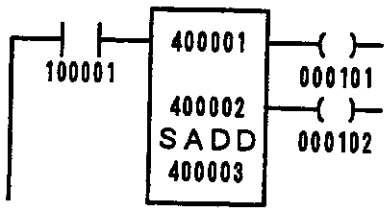


400001 -1000 400002 2000

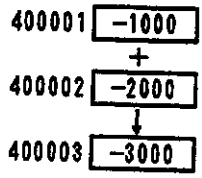
For the above SADD, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 and 000102 will remain OFF. The result will remain in holding register 400003 even after input relay 100001 turns OFF.

Example 2

1) Ladder Programming



2) Operation

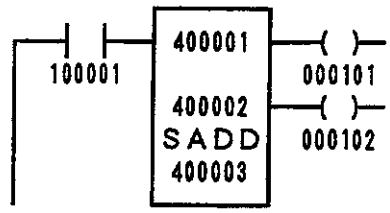


400001 -1000 400002 -2000

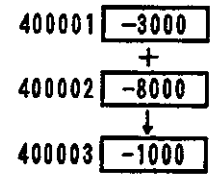
For the above SADD, the operation shown at the right will be performed when input relay 100001 is ON, coil 000101 will turn ON and coil 000102 will remain OFF. When input relay 100001 turns OFF, coil 000101 will turn OFF and the result will remain in holding register 400003.

Example 3

1) Ladder Programming



2) Operation



(-3,000-8,000)+10,000=-1,000

400001 -3000 400002 -8000

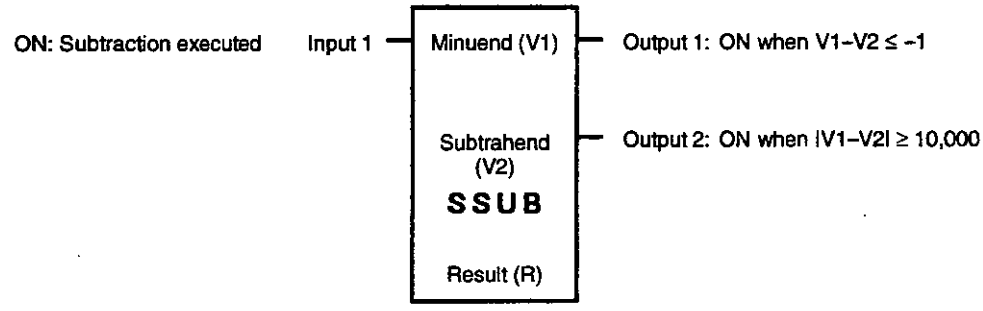
For the above SADD, the operation shown at the right will be performed when input relay 100001 is ON, and coils 000101 and 000102 will turn ON. When input relay 100001 turns OFF, coils 000101 and 000102 will turn OFF, but the result will remain in holding register 400003.

2.5.3 SIGNED SINGLE PRECISION DECIMAL SUBTRACTION (SSUB)

1. Function

Signed subtraction is performed between two 4-digit decimal numbers, V1 and V2.

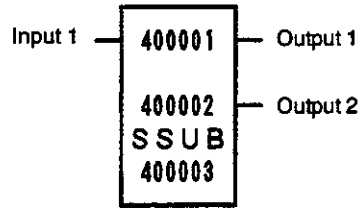
2. Structure



1) SSUB is the symbol for SIGNED SINGLE PRECISION DECIMAL SUBTRACTION.

2) SSUB requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.25* lists the register reference numbers that can be specified.

Example



400001: Reference number of holding register for minuend, V1
400002: Reference number of holding register for subtrahend, V2
400003: Reference number of holding register to store result, R

Table 2.25 Structural Elements of SSUB

Element	Meaning	Possible Settings	
Top (V1)	1) The contents of the register is used as the minuend, V1. 2) V1 must be between –9,999 and 9,999. 3) Set the MSB of the register to 1 if V1 < 0.	Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 R20001 to R21024
Middle (V2)	1) The contents of the register is used as the subtrahend, V2. 2) V2 must be between –9,999 and 9,999. 3) Set the MSB of the register to 1 if V2 < 0.		
Bottom (R)	1) The result is stored in the register. 2) The result must be between –9,999 and 9,999. 3) The MSB of the register will be set to 1 if the results < 0.	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 R20001 to R21024

3. Operation

1) SSUB will subtract V2 from V1 when input 1 is ON and process the result as follows:

a) If $0 \leq V1-V2 \leq 9,999$:

(1) The result of V1–V2 is stored in R.

(2) The MSB of R is set to 0 to indicate a positive result.

(3) Outputs 1 and 2 remain OFF.

b) If $-9,999 \leq V1-V2 \leq -1$:

(1) The result of V1–V2 is stored in R.

(2) The MSB of R is set to 1 to indicate a negative result.

(3) Output 1 turns ON and output 2 remains OFF.

c) If $10,000 \leq V1-V2 \leq 19,998$:

(1) The result of V1–V2 – 10,000 is stored in R.

(2) The MSB of R is set to 0 to indicate a positive result.

(3) Output 1 remains OFF and output 2 turns ON.

d) If $-19,998 \leq V1-V2 \leq -10,000$:

(1) The result of V1–V2 + 10,000 is stored in R.

(2) The MSB of R is set to 1 to indicate a negative result.

(3) Outputs 1 and 2 turn ON .

2) The result remains in R even if input 1 turns OFF.

3) The operation of SSUB is summarized in *Table 2.26.*

Table 2.26 Operation of SSUB

Input 1	Condition	Operation	Outputs	
			1	2
ON	0≤V1–V2≤9,999	1) Result of V1–V2 stored in R. 2) MSB of R set to 0.	OFF	OFF
	–9,999≤V1–V2≤–1	1) Result of V1–V2 stored in R. 2) MSB of R set to 1.	**ON**	OFF
	10,000≤V1–V2≤19,998	1) Result of V1–V2 – 10,000 stored in R. 2) MSB of R set to 0.	OFF	**ON**
	–19,998≤V1–V2≤–10,000	1) Result of V1–V2 + 10,000 stored in R. 2) MSB of R set to 1.	**ON**	**ON**
OFF	None	Nothing is done.	OFF	OFF

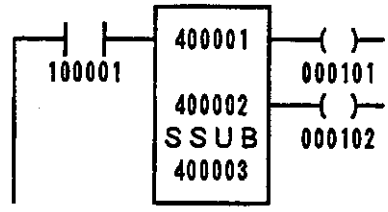
Note Both V1 and V2 must be between –9,999 and 9,999. SSUB will not operate properly if V1 or V2 is not within this range.

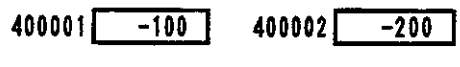
◀EXAMPLE▶

4. Application Examples

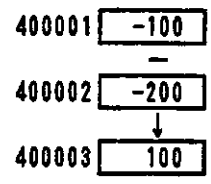
Example 1

1) Ladder Programming





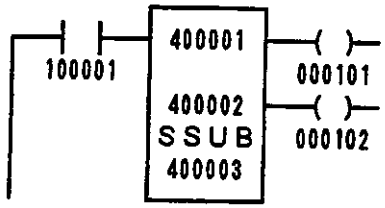
2) Operation



For the above SSUB, the operation shown at the right will be performed when input relay 100001 is ON. Coils 000101 and 000102 will remain OFF. The result will remain in holding register 400003 even after input relay 100001 turns OFF.

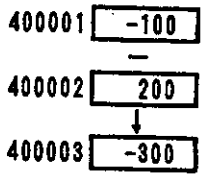
Example 2

1) Ladder Programming



400001 [-100] 400002 [200]

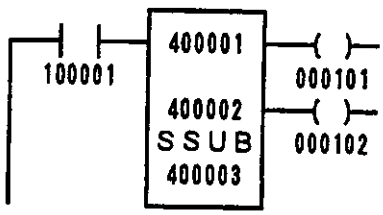
2) Operation



For the above SSUB, the operation shown at the right will be performed when input relay 100001 is ON, coil 000101 will turn ON and coil 000102 will remain OFF. When input relay 100001 turns OFF, coil 000101 will turn OFF and the result will remain in holding register 400003.

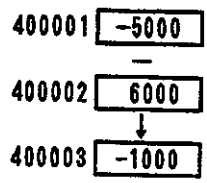
Example 3

1) Ladder Programming



400001 [-5000] 400002 [6000]

2) Operation



(-5000-6000) +10000=-1000

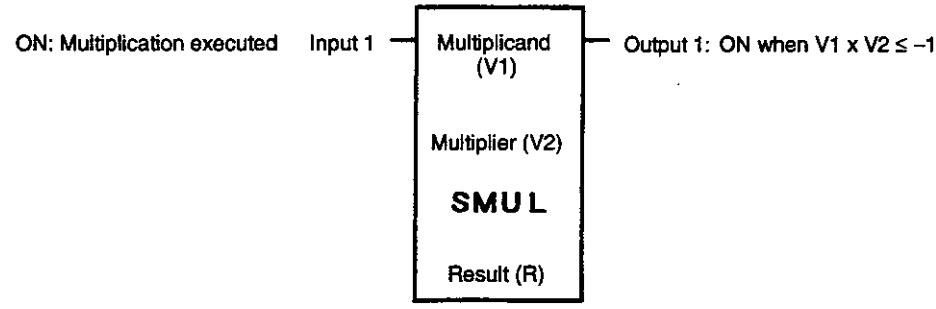
For the above SSUB, the operation shown at the right will be performed when input relay 100001 is ON, and coils 000101 and 000102 will turn ON. When input relay 100001 turns OFF, coils 000101 and 000102 will turn OFF, but the result will remain in holding register 400003.

2.5.4 SIGNED SINGLE PRECISION DECIMAL MULTIPLICATION (SMUL)

1. Function

Signed multiplication is performed between two 4-digit decimal numbers, V1 and V2.

2. Structure



1) SMUL is the symbol for SIGNED SINGLE PRECISION DECIMAL MULTIPLICATION.

2) SMUL requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.27* lists the register reference numbers that can be specified.

Example

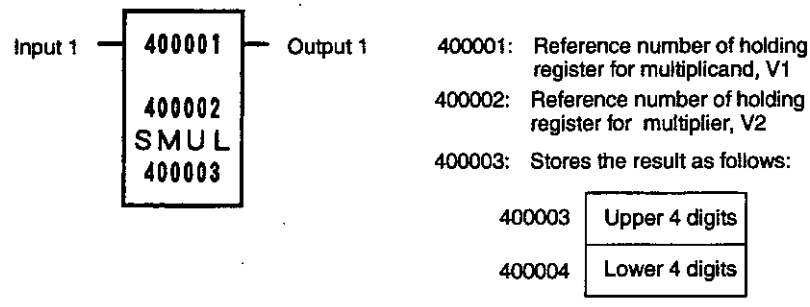


Table 2.27 Structural Elements of SMUL

Element	Meaning	Possible Settings
Top (V1)	1) The contents of the register is used as the multiplicand, V1. 2) V1 must be between –9,999 and 9,999. 3) Set the MSB of the register to 1 if V1 < 0.	Input register: 300001 to 300512 (Z00001 to Z00512) Holding register: 400001 to 409999 (W00001 to W09999) Constant register: 700001 to 704096 (K00001 to K04096) Link register: R10001 to R11024 R20001 to R21024
Middle (V2)	1) The contents of the register is used as the multiplier, V2. 2) V2 must be between –9,999 and 9,999. 3) Set the MSB of the register to 1 if V2 < 0.	
Bottom (R)	1) The result is stored in the specified register and the next register as shown below. 2) The result must be between –99,980,001 and 99,980,001. 3) The MSB of the register will be set to 1 if the result < 0. 4) In the example, "400003" was specified for the bottom element. The MSB of 400003 will be set to 1 if the result is negative. 400003 Upper 4 digits of result 400004 Lower 4 digits of result	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 R20001 to R21023

3. Operation

1) SMUL will multiply V1 by V2 when input 1 is ON and process the result as follows:

a) If $0 \leq V1 \times V2 \leq 99,980,001$:

(1) The upper 4 digits of the result are stored in R and the lower 4 digits are stored in R+1.

(2) The MSB of R is set to 0 to indicate a positive result.

(3) Output 1 remains OFF.

b) If $-99,980,001 \leq V1 \times V2 \leq -1$:

(1) The upper 4 digits of the result are stored in R and the lower 4 digits are stored in R+1.

(2) The MSB of R is set to 1 to indicate a negative result.

(3) Output 1 turns ON.

2) The result remains in R and R+1 even if input 1 turns OFF.

3) The operation of SMUL is summarized in *Table 2.28.*

Table 2.28 Operation of SMUL

Input 1	Condition	Operation	Outputs	
			1	2
ON	0≤V1 × V2≤99,980,001	1) The upper 4 digits of the result are stored in R and the lower 4 digits are stored in R+1. 2) The MSB of R is set to 0 to indicate a positive result.	OFF	OFF
	−99,980,001≤V1 × V2≤−1	1) The upper 4 digits of the result are stored in R and the lower 4 digits are stored in R+1. 2) The MSB of R is set to 1 to indicate a negative result.	**ON**	OFF
OFF	None	Nothing is done.	OFF	OFF

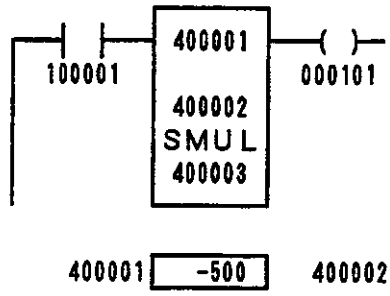
Note Both V1 and V2 must be between −9,999 and 9,999. SMUL will not operate properly if V1 or V2 is not within this range.

◀EXAMPLE▶

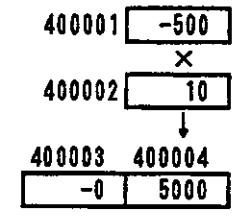
4. Application Example

Example

1) Ladder Programming



2) Operation



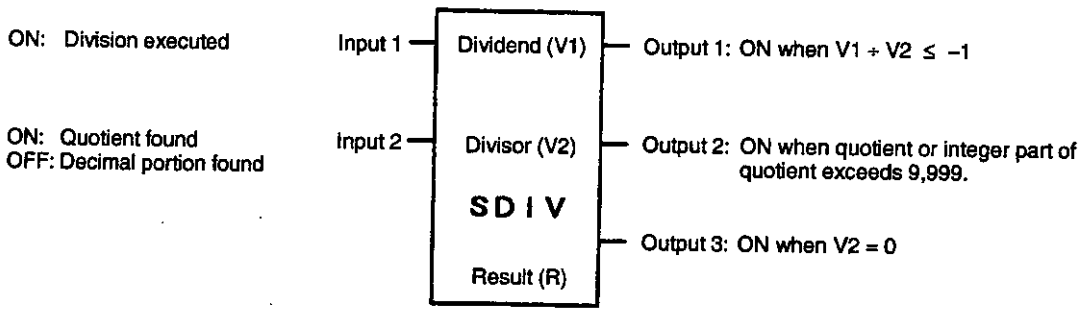
For the above SMUL, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101will turn ON. When input relay 100001 turns OFF, coil 000101 will turn OFF, but the result will remain in holding registers 400003 and 400004.

2.5.5 SIGNED SINGLE PRECISION DECIMAL DIVISION (SDIV)

1. Function

Signed division is performed between a 8-digit decimal number V1 and a 4-digit decimal number V2 (V1 ÷ V2).

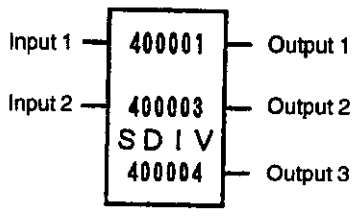
2. Structure



1) SDIV is the symbol for SIGNED SINGLE PRECISION DECIMAL DIVISION.

2) SDIV requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.29* lists the register reference numbers that can be specified.

Example



400001: Stores the dividend as follows:

400001	Upper 4 digits
400002	Lower 4 digits

400003: Reference number of holding register for divisor

400004: Stores the result as follows:

400004	Quotient
400005	Remainder

or

400004	Integer portion of quotient
400005	Decimal portion of quotient

Table 2.29 Structural Elements of SDIV

Element	Meaning	Possible Settings
Top (V1)	1) The contents of two consecutive registers is used as the dividend, V1, as shown in the following example. 2) V1 must be between –99,989,999 and 99,989,999. 3) Set the MSB of the registers to 1 if V1 < 0. 4) In the example, "400001" was specified for the top element. 400001 Upper 4 digits 400002 Lower 4 digits Negative: Set MSB of 400001 to 1 .	Input register: 300001 to 300511 (Z00001 to Z00511) Holding register: 400001 to 409998 (W00001 to W09998) Constant register: 700001 to 704095 (K00001 to K04095) Link register: R10001 to R11023 R20001 to R21023
Middle (V2)	1) The contents of the register is used as the divisor, V2. 2) V2 must be between –9,999 and 9,999. 3) Set the MSB of the register to 1 if V2 < 0.	Input register: 300001 to 300512 (Z00001 to Z00512) Holding register: 400001 to 409999 (W00001 to W09999) Constant register: 700001 to 704096 (K00001 to K04096) Link register: R10001 to R11024 R20001 to R21024
Bottom (R)	1) The result is stored in the specified register and the next register as shown below. 2) The quotient, remainder, and integer portion of the quotient must be between –9,999 and 9,999. The decimal portion of the quotient must be between 0 and 9,999. R Quotient R+1 Remainder or R Integer portion of quotient R+1 Decimal portion of quotient 3) The MSB of R will be set to 1 if the results < 0. 4) In the example, "400004" was specified for the bottom element. 400004 Quotient 400005 Remainder Negative: MSB of 400004 and 400005 set to 1 or 400004 Integer portion of quotient 400005 Decimal portion of quotient Negative: MSB of 400004 set to 1	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 R20001 to R21023

3. Operation

1) SDIV will divide V1 by V2 when input 1 is ON and process the result as follows:

   a) If input 2 is OFF:

      (1) The quotient of V1 ÷ V2 is stored in R and remainder is stored in R+1.

      (2) If the quotient and remainder are positive, the MSB of R will be set to 0 and all outputs will turn OFF.

      (3) If the quotient and remainder are negative, the MSB of R will be set to 1 and output 1 will turn ON.

   b) If input 2 is ON:

      (1) The integer portion of the quotient of V1 ÷ V2 is stored in R and the decimal portion is stored in R+1.

      (2) If the integer portion is positive, the MSB of R will be set to 0 and all outputs will turn OFF.

      (3) If the integer portion is negative, the MSB of R will be set to 1 and output 1 will turn ON.

   c) Division will not be executed in the following cases and zero (0) is stored in R and R+1.

      (1) V2 = 0. In this case, output 3 turns ON.

      (2) If the quotient or integer portion of the quotient will not fit in R. In this case, output 2 turns ON.
      Example: If V1 = 500,000 and V2 = 10, the quotient is –50,000, which cannot be stored in R. Here, 0 is stored in R and R+1.

2) The result remains in R and R+1 even if input 1 turns OFF.

3) The operation of SDIV is summarized in *Table 2.30.*

Table 2.30 Operation of SDIV

Inputs		Condition	Operation	Outputs		
1	2			1	2	3
ON	OFF	V2 ≠ 0 and $\|V1_H\| < \|V2\|$ *1	1) Result of V1 ÷ V2 stored as follows: R: Quotient R+1: Remainder 2) If result < 0, MSB of R and R+1 are set to 1.	**ON** (if the result is negative)	OFF	OFF
		V2 ≠ 0 and $\|V1_H\| \geq \|V2\|$ *1	Correct execution not possible. Zero (0) stored in R and R+1.	OFF	**ON**	OFF
		V2 = 0	Execution not possible. Zero (0) stored in R and R+1.	OFF	OFF	**ON**
ON	**ON**	V2 ≠ 0 and $\|V1_H\| < \|V2\|$ *1	1) Result of V1 ÷ V2 stored as follows: R: Integer portion R+1: Decimal portion Truncated after 4th decimal place 2) If result < 0, MSB of R is set to 1.	**ON** (if the result is negative)	OFF	OFF
		V2 ≠ 0 and $\|V1_H\| \geq \|V2\|$ *1	Correct execution not possible. Zero (0) stored in R and R+1.	OFF	**ON**	OFF
		V2 = 0	Execution not possible. Zero (0) stored in R and R+1.	OFF	OFF	**ON**
OFF	Any	None	Nothing is done.	OFF	OFF	OFF

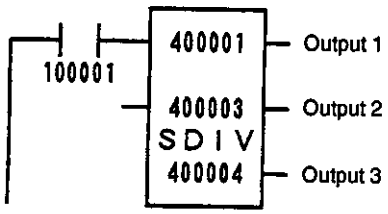
*1: $V1_H$ is the upper 4 digits of V1.

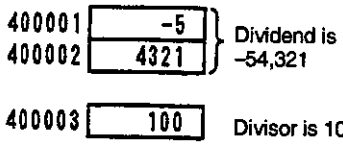
Note V1 must be between -99,989,999 and 99,989,999 and V2 must be between -9,999 and 9,999. SDIV will not operate properly if V1 or V2 is not within this range.

4. Application Examples

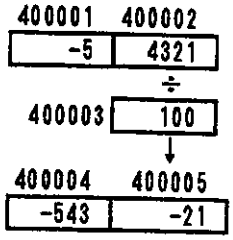
◀EXAMPLE▶ **Example 1**

1) Ladder Programming





2) Operation

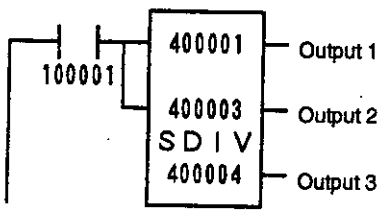


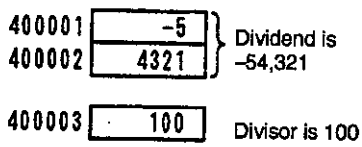
-54,321 ÷ 100 = -543 with remainder of -21

For the above SDIV, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. When input relay 100001 turns OFF, output 1 will turn OFF, but the result will remain in holding registers 400004 and 400005.

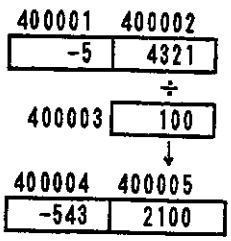
Example 2

1) Ladder Programming





2) Operation



(-54, 321 ÷ 100) = -543. 2100

For the above SDIV, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. When input relay 100001 turns OFF, output 1 will turn OFF, but the result will remain in holding registers 400004 and 400005.

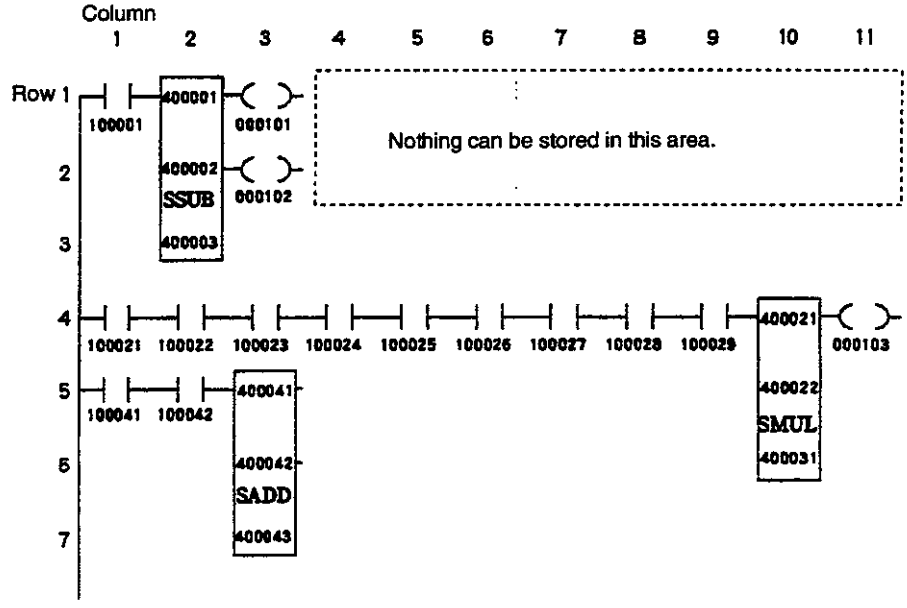
2.5.6 Building Programs

1. Storage Locations on Networks

All signed, 4-digit, decimal arithmetic instructions require three vertical elements on a network, one top element, one middle element, and one bottom element. They can thus be stored anywhere on a 5-row by 10-column matrix (rows 1 through 5 and columns 1 through 10) on the network.

Note Signed, 4-digit, decimal arithmetic instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

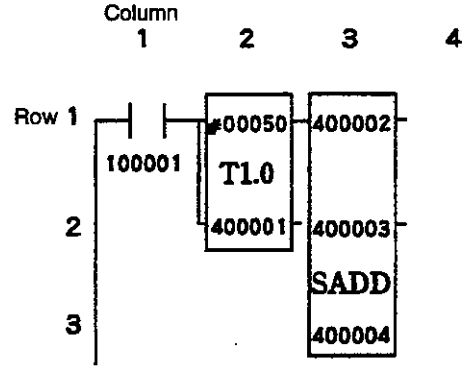
Example



2. Inputs

Inputs to signed, 4-digit, decimal math instruction can be connected to relay elements (except coils) and/or outputs from timers, counters, math instructions, data transfer instructions, other instructions, etc.

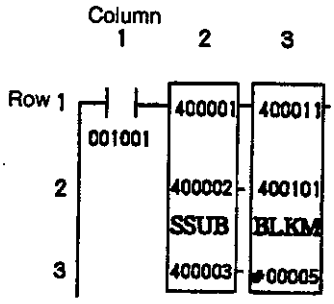
Example



3. Outputs

Outputs from signed, 4-digit, decimal math instruction can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data transfer instructions, etc.

Example



2.6 Signed, Eight-digit, Decimal Arithmetic Instructions

This section describes the functions, structures, and operation of the signed, 8-digit, decimal arithmetic instructions and provides simple examples of their application.

2.6.1 Instructions

Signed, 8-digit, decimal arithmetic instructions perform signed addition and subtraction on two 8-digit, decimal values, V1 and V2. The instructions that are available are shown in 2.31.

Table 2.31 Signed, 8-digit, Decimal Arithmetic Instructions

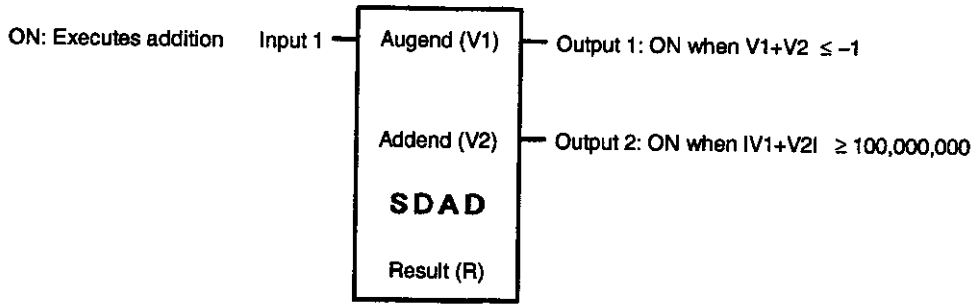
Name	Symbol	Operands	V1	V2	Result
SIGNED DOUBLE PRECISION DECIMAL ADDITION	SDAD	V1 + V2	-99,999,999 to 99,999,999		
SIGNED DOUBLE PRECISION DECIMAL SUBTRACTION	SDSB	V1 – V2			

2.6.2 SIGNED DOUBLE PRECISION DECIMAL ADDITION (SDAD)

1. Function

Signed addition is performed between two 8-digit decimal numbers, V1 and V2.

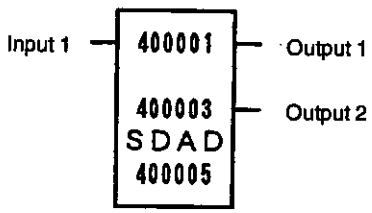
2. Structure



1) SDAD is the symbol for SIGNED DOUBLE PRECISION DECIMAL ADDITION.

2) SDAD requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.32* lists the register reference numbers that can be specified.

Example



400001: The augend, V1, is stored as follows:

400001	Upper 4 digits
400002	Lower 4 digits

400003: The addend, V2, is stored as follows:

400003	Upper 4 digits
400004	Lower 4 digits

400005: The result is stored as follows:

400005	Upper 4 digits of result
400006	Lower 4 digits of result

Table 2.32 Structural Elements of SDAD

Element	Meaning	Possible Settings
Top (V1)	1) The contents of two consecutive registers is used as the augend, V1, as shown in the following example. 2) The value of V1 must be between –99,999,999 and 99,999,999. 3) If V1 < 0, set the MSB to 1. 4) In the example, "400001" was specified for the top element. 400001 Upper 4 digits 400002 Lower 4 digits Negative: Set MSB of 400001 to 1.	Input register: 300001 to 300511 (Z00001 to Z00511) Holding register: 400001 to 409998 (W00001 to W09998) Constant register: 700001 to 704095 (K00001 to K04095) Link register: R10001 to R11023 R20001 to R21023
Middle (V2)	1) The contents of two consecutive registers is used as the addend, V2, as shown in the following example. 2) The value of V2 must be between –99,999,999 and 99,999,999. 3) If V2 < 0, set the MSB to 1. 4) In the example, "400003" was specified for the top element. 400003 Upper 4 digits 400004 Lower 4 digits Negative: Set MSB of 400003 to 1.	
Bottom (R)	1) The result is stored in two consecutive registers, as shown in the following example. 2) The value of the result must be between –99,999,999 and 99,999,999. 3) If result < 0, the MSB of R is set to 1. 4) In the example, "400005" was specified for the top element. 400005 Upper 4 digits 400006 Lower 4 digits Negative: MSB of 400005 set to 1.	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 R20001 to R21023

3. Operation

1) SDAD will add V2 to V1 when input 1 is ON and process the result as follows:

a) If $0 \leq V1 + V2 \leq 99,999,999$:

(1) The upper 4 digits of the result of V1 + V2 are stored in R and the lower 4 digits are stored in R+1.

(2) The MSB (most significant bit) of R is set to 0 to indicate a positive result.

(3) Outputs 1 and 2 remain OFF.

b) If $-99,999,999 \leq V1 + V2 \leq -1$:

(1) The upper 4 digits of the result of V1 + V2 are stored in R and the lower 4 digits are stored in R+1.

(2) The MSB of R is set to 1 to indicate a negative result.

(3) Output 1 turns ON and output 2 remains OFF.

c) If $100,000,000 \leq V1 + V2 \leq 199,999,998$:

(1) The upper 4 digits of the result of V1 + V2 – 100,000,000 are stored in R and the lower 4 bits are stored in R+1.

(2) The MSB of R is set to 0 to indicate a positive result.

(3) Output 1 remains OFF and output 2 turns ON.

d) If $-199,999,998 \leq V1 + V2 \leq -100,000,000$:

(1) The upper 4 digits of the result of V1 + V2 + 100,000,000 are stored in R and the lower 4 bits are stored in R+1.

(2) The MSB of R is set to 1 to indicate a negative result.

(3) Outputs 1 and 2 turn ON.

2) The result R and R+1 remains even if input 1 turns OFF.

3) The operation of SDAD is summarized in *Table 2.33*.

Table 2.33 Operation of SDAD

Input 1	Condition	Operation	Outputs	
			1	2
ON	0 ≤ V1 + V2 ≤ 99,999,999	The result of V1 + V2 is stored as follows: R: Upper 4 digits R+1: Lower 4 digits	OFF	OFF
	–99,999,999 ≤ V1 + V2 ≤ –1	The result of V1 + V2 is stored as follows and MSB of R is set to 1: R: Upper 4 digits R+1: Lower 4 digits	**ON**	OFF
	100,000,000 ≤ V1 + V2 ≤ 199,999,998	The result of V1 + V2 – 100,000,000 is stored as follows: R: Upper 4 digits R+1: Lower 4 digits	OFF	**ON**
	–199,999,998 ≤ V1 + V2 ≤ –100,000,000	The result of V1 + V2 + 100,000,000 is stored as follows and MSB of R is set to 1: R: Upper 4 digits R+1: Lower 4 digits	**ON**	**ON**
OFF	None	Nothing is done.	OFF	OFF

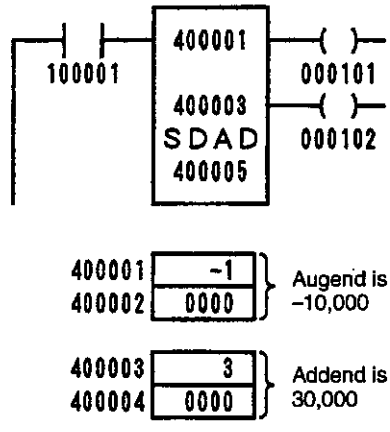
Note Both V1 and V2 must be between –99,999,999 and 99,999,999. SDAD will not operate properly if V1 or V2 is not within this range.

◀EXAMPLE▶

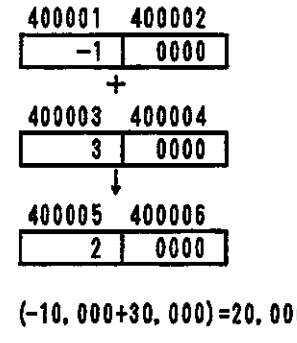
4. Application Examples

Example 1

1) Ladder Programming



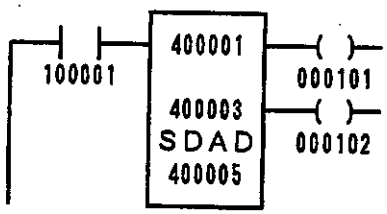
2) Operation

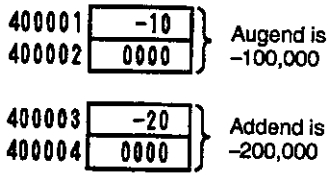


For the above SDAD, the operation shown at the right will be performed when input relay 100001 is ON. Coils 000101 and 000102 will remain OFF. The result will remain in holding registers 400005 and 400006 even after input relay 100001 turns OFF.

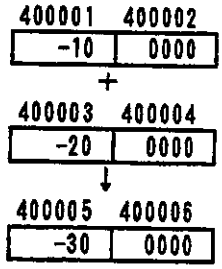
Example 2

1) Ladder Programming





2) Operation

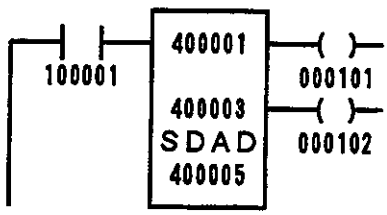


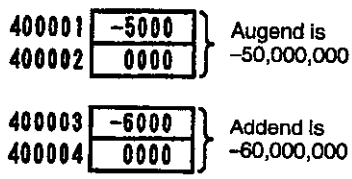
(-100, 000-200, 000)=-300, 000

For the above SDAD, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON. Coil 000102 will remain OFF. When input relay 100001 turns OFF, coil 000101 will turn OFF, and the result will remain in holding registers 400005 and 400006.

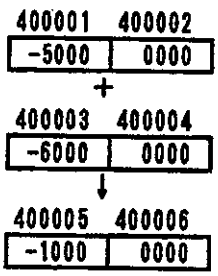
Example 3

1) Ladder Programming





2) Operation



(-50, 000, 000-60, 000, 000+
100, 000, 000)=-10, 000, 000

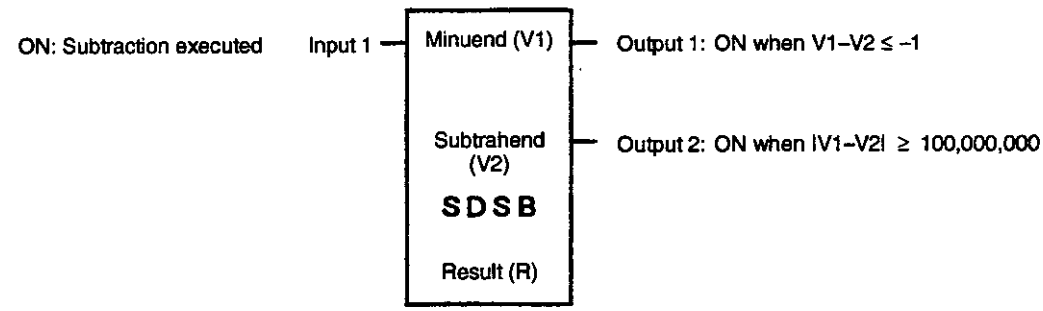
For the above SDAD, the operation shown at the right will be performed when input relay 100001 is ON, and coils 000101 and 000102 will turn ON. When input relay 100001 turns OFF, coils 000101 and 000102 will turn OFF, and the result will remain in holding registers 400005 and 400006.

2.6.3 SIGNED DOUBLE PRECISION DECIMAL SUBTRACTION (SDSB)

1. Function

Signed subtraction is performed between two 8-digit decimal numbers, V1 and V2.

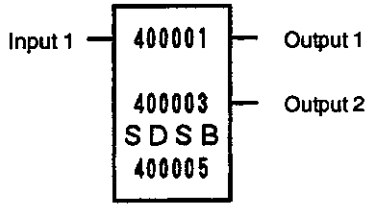
2. Structure



1) SDSB is the symbol for SIGNED DOUBLE PRECISION DECIMAL SUBTRACTION.

2) SDSB requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.34* lists the register reference numbers that can be specified.

Example



400001: The minuend, V1, is stored as follows:

400001	Upper 4 digits
400002	Lower 4 digits

400003: The subtrahend, V2, is stored as follows:

400003	Upper 4 digits
400004	Lower 4 digits

400005: The result is stored as follows:

400005	Upper 4 digits of result
400006	Lower 4 digits of result

Table 2.34 Structural Elements of SDSB

Element	Meaning	Possible Settings
Top (V1)	1) The contents of two consecutive registers is used as the minuend, V1, as shown in the following example. 2) The value of V1 must be between –99,999,999 and 99,999,999. 3) If V1 < 0, set the MSB to 1. 4) In the example, "400001" was specified for the top element. 400001 Upper 4 digits 400002 Lower 4 digits Negative: Set MSB of 400001 to 1.	Input register: 300001 to 300511 (Z00001 to Z00511) Holding register: 400001 to 409998 (W00001 to W09998) Constant register: 700001 to 704095 (K00001 to K04095) Link register: R10001 to R11023 R20001 to R21023
Middle (V2)	1) The contents of two consecutive registers is used as the subtrahend, V2, as shown in the following example. 2) The value of V2 must be between –99,999,999 and 99,999,999. 3) If V2 < 0, set the MSB to 1. 4) In the example, "400003" was specified for the top element. 400003 Upper 4 digits 400004 Lower 4 digits Negative: Set MSB of 400003 to 1.	
Bottom (R)	1) The result is stored in two consecutive registers, as shown in the following example. 2) The value of the result must be between –99,999,999 and 99,999,999. 3) If result < 0, the MSB of R is set to 1. 4) In the example, "400005" was specified for the top element. 400005 Upper 4 digits 400006 Lower 4 digits Negative: MSB of 400005 set to 1.	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 R20001 to R21023

3. Operation

1) SDSB will subtract V2 from V1 when input 1 is ON and process the result as follows:

   a) If $0 \leq V1 - V2 \leq 99,999,999$:

      (1) The upper 4 digits of the result of V1 – V2 are stored in R and the lower 4 digits are stored in R+1.

(2) The MSB of R is set to 0 to indicate a positive result.

(3) Outputs 1 and 2 remain OFF.

b) If $-99,999,999 \leq V1 - V2 \leq -1$:

(1) The upper 4 digits of the result of V1 – V2 are stored in R and the lower 4 digits are stored in R+1.

(2) The MSB of R is set to 1 to indicate a negative result.

(3) Output 1 turns ON and output 2 remains OFF.

c) If $100,000,000 \leq V1 - V2 \leq 199,999,998$:

(1) The upper 4 digits of the result of V1 – V2 – 100,000,000 are stored in R and the lower 4 digits are stored in R+1.

(2) The MSB of R is set to 0 to indicate a positive result.

(3) Output 1 remains OFF and output 2 turns ON.

d) If $-199,999,998 \leq V1 - V2 \leq -100,000,000$:

(1) The upper 4 digits of the result of V1 – V2 + 100,000,000 are stored in R and the lower 4 digits are stored in R+1.

(2) The MSB of R is set to 1 to indicate a negative result.

(3) Outputs 1 and 2 turn ON.

2) The result remains in R and R+1 even if input 1 turns OFF.

3) The operation of SDSB is summarized in *Table 2.35*.

Table 2.35 Operation of SDSB

Input 1	Condition	Operation	Outputs 1	Outputs 2
ON	0 ≤ V1–V2 ≤ 99,999,999	The result of V1 – V2 is stored as follows and MSB of R is set to 0: R: Upper 4 digits R+1: Lower 4 digits	OFF	OFF
	–99,999,999,999 ≤ V1–V2 ≤ –1	The result of V1 – V2 is stored as follows and MSB of R is set to 1: R: Upper 4 digits R+1: Lower 4 digits	**ON**	OFF
	100,000,000 ≤ V1–V2 ≤ 199,999,998	The result of V1 – V2 – 100,000,000 is stored as follows and MSB of R is set to 0: R: Upper 4 digits R+1: Lower 4 digits	OFF	**ON**
	–199,999,998 ≤ V1–V2 ≤ –100,000,000	The result of V1 – V2 + 100,000,000 is stored as follows and MSB of R is set to 1: R: Upper 4 digits R+1: Lower 4 digits	**ON**	**ON**
OFF	None	Nothing is done.	OFF	OFF

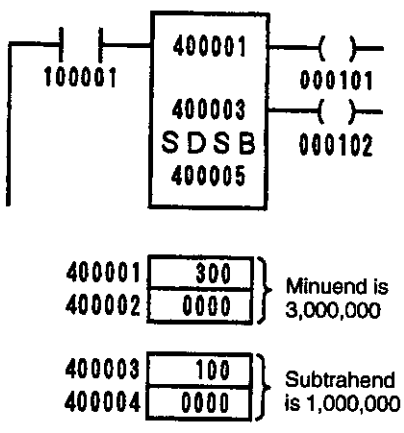
Note Both V1 and V2 must be between –99,999,999 and 99,999,999. SDSB will not operate properly if V1 or V2 is not within this range.

◀EXAMPLE▶

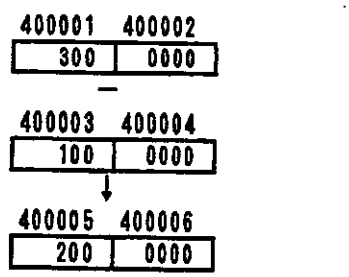
4. Application Examples

Example 1

1) Ladder Programming



2) Operation

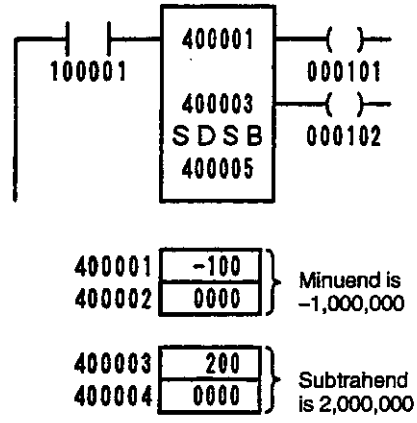


(3,000,000–1,000,000)=2,000,000

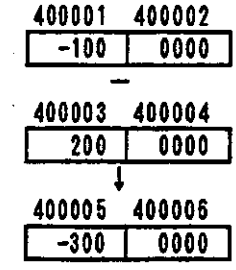
For the above SDSB, the operation shown at the right will be performed when input relay 100001 is ON. Coils 000101 and 000102 will remain OFF. The result will remain in holding registers 400005 and 400006 even after input relay 100001 turns OFF.

Example 2

1) Ladder Programming



2) Operation

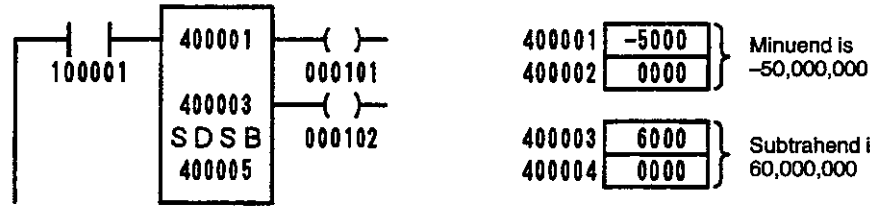


(-1, 000, 000-2, 000, 000) =-3, 000, 000

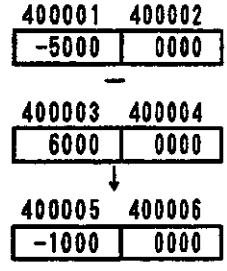
For the above SDSB, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON. Coil 000102 will remain OFF. When input relay 100001 turns OFF, coil 000101 will turn OFF, and the result will remain in holding registers 400005 and 400006.

Example 3

1) Ladder Programming



2) Operation



(-50, 000, 000-60, 000, 000+100, 000, 000) =-10, 000, 000

For the above SDSB, the operation shown at the right will be performed when input relay 100001 is ON, and coils 000101 and 000102 will turn ON. When input relay 100001 turns OFF, coils 000101 and 000102 will turn OFF, and the result will remain in holding registers 400005 and 400006.

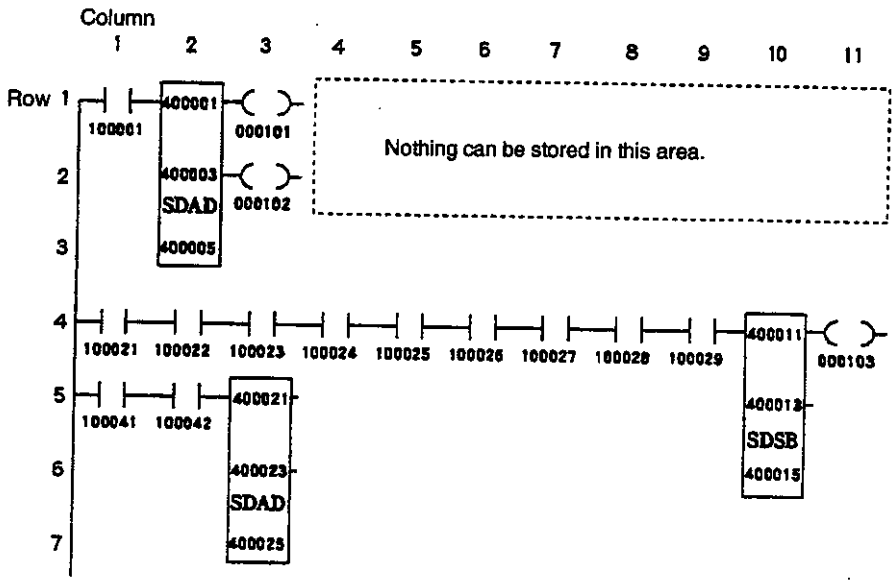
2.6.4 Building Programs

1. Storage Locations on Networks

All signed, 8-digit, decimal arithmetic instructions require three vertical elements on a network, one top element, one middle element, and one bottom element. They can thus be stored anywhere on a 5-row by 10-column matrix (rows 1 through 5 and columns 1 through 10) on the network.

Note Signed, 4-digit, decimal arithmetic instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

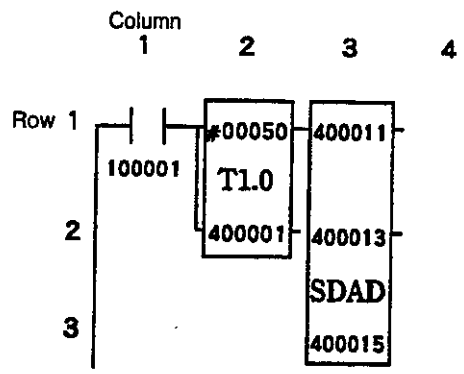
Example



2. Inputs

Inputs to signed, 8-digit, decimal math instruction can be connected to relay elements (except coils) and/or outputs from timers, counters, math instructions, data transfer instructions, etc.

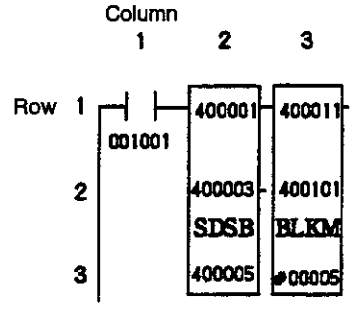
Example



3. Outputs

Outputs from signed, 8-digit, decimal math instruction can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data transfer instructions, etc.

Example



2.7 Decimal Square Root Instructions

This section describes the functions, structures, and operation of the decimal square root instructions and provides simple examples of their application.

2.7.1 Instructions

Decimal square root instructions find the square root of a 4-digit or 8-digit, decimal value, V. The instructions that are available are shown in *Table 2.36.*

Table 2.36 Decimal Square Root Instructions

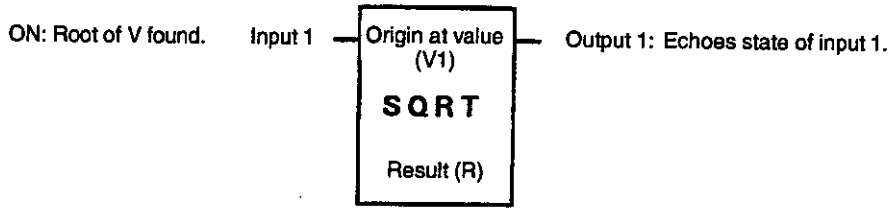
Name	Symbol	Operation	V	Result
SINGLE PRECISION DECIMAL SQUARE ROOT	SQRT	$\sqrt{V}$	0 to 9,999	0 to 99.9949
DOUBLE PRECISION DECIMAL SQUARE ROOT	DSQR		0 to 99,999,999	0 to 9999.9999

2.7.2 SINGLE PRECISION DECIMAL SQUARE ROOT (SQRT)

1. Function

The square root of a 4-digit decimal value, V, is found.

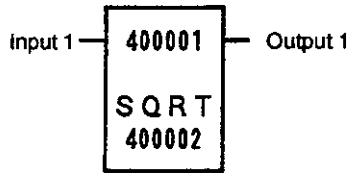
2. Structure



1) SQRT is the symbol for SINGLE PRECISION DECIMAL SQUARE ROOT.

2) SQRT requires two elements, one top element and one bottom element, located vertically on the network. *Table 2.37* lists the register reference numbers that can be specified.

Example



400001: Stores the value of V.

400002: Stores the square root of V as follows:

400002	Integer portion of result
400003	Decimal portion of result

Table 2.37 Structural Elements of SQRT

Element	Meaning	Possible Settings
Top (V)	The contents of the specified register is used as V. V must be between 0 and 9,999.	Input register: 300001 to 300512 (Z00001 to Z00512) Holding register: 400001 to 409999 (W00001 to W09999) Constant register: 700001 to 704096 (K00001 to K04096) Link register: R10001 to R11024 R20001 to R21024
Bottom (R)	The square root of V is stored in registers as shown below. The decimal portion is truncated after the 4th decimal place. In the example, "400002" was specified for the bottom element. 400002 Integer portion of result 400003 Decimal portion of result	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 R20001 to R21023

3. Operation

1) SQRT will take the square root of V when input 1 is ON and process the result as follows:

   a) The integer portion of the square root is stored in R and the decimal portion is stored in R+1. The decimal portion is truncated after the 4th decimal place.

   b) Output 1 turns OFF.

2) The result in R and R+1 remains even if input 1 turns OFF.

3) The operation of SQRT is summarized in the following table.

Table 2.38 Operation of SQRT

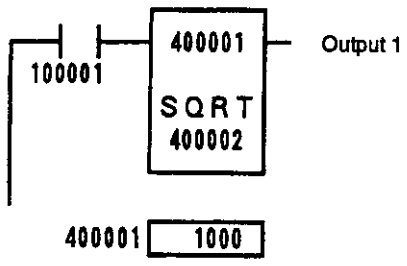
Input 1	Condition	Operation	Output 1
ON	None	The square root of V is stored as shown below. The decimal portion is truncated after the 4th decimal place. R: Integer portion of result R+1: Decimal portion of result	ON
OFF		Nothing is done.	OFF

Note V must be between 0 and 9,999. SQRT will not operate properly if V is not within this range.

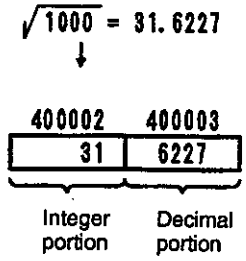
◀EXAMPLE▶

4. Application Example

1) Ladder Programming



2) Operation



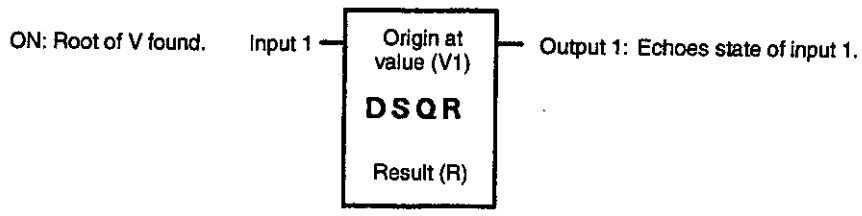
For the above SQRT, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. When input relay 100001 turns OFF, output 1 will turn OFF, but the result will remain in holding registers 400002 and 400003.

2.7.3 DOUBLE PRECISION DECIMAL SQUARE ROOT (DSQR)

1. Function

The square root of an 8-digit decimal value, V, is found.

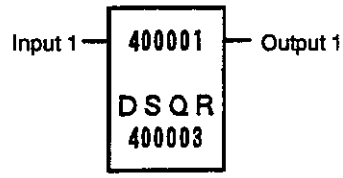
2. Structure



1) DSQR is the symbol for DOUBLE PRECISION DECIMAL SQUARE ROOT.

2) DSQR requires two elements, one top element and one bottom element, located vertically on the network. *Table 2.39* lists the register reference numbers that can be specified.

Example



400001: Stores the value of V as follows:

400001	Upper 4 digits
400002	Lower 4 digits

400003: Stores the square root of V as follows:

400003	Integer portion of result
400004	Decimal portion of result

Table 2.39 Structural Elements of DSQR

Element	Meaning	Possible Settings
Top (V1)	The contents of the specified register and the next register is used as V as shown below. V must be between 0 and 99,999,999. In the example, "400001" was specified for the bottom element. 400001 Upper 4 digits 400002 Lower 4 digits	Input register: 300001 to 300511 (Z00001 to Z00511) Holding register: 400001 to 409998 (W00001 to W09998) Constant register: 700001 to 704095 (K00001 to K04095) Link register: R10001 to R11023 R20001 to R21023
Bottom (R)	The square root of V is stored in registers as shown below. The decimal portion is truncated after the 4th decimal place. In the example, "400003" was specified for the bottom element. 400003 Integer portion of result 400004 Decimal portion of result	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 R20001 to R21023

3. Operation

1) DSQR will take the square root of V when input 1 is ON and process the result as follows:

   a) The integer portion of the square root is stored in R and the decimal portion is stored in R+1. The decimal portion is truncated after the 4th decimal place.

   b) Output 1 turns ON.

2) The result in R and R+1 remains even if input 1 turns OFF.

3) The operation of DSQR is summarized in the following table.

Table 2.40 Operation of DSQR

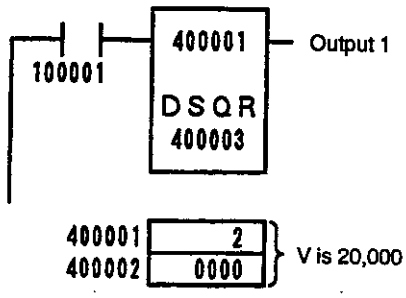
Input 1	Condition	Operation	Output 1
ON	None	The square root of V is stored as shown below. The decimal portion is truncated after the 4th decimal place. R: Integer portion of result R+1: Decimal portion of result	ON
OFF		Nothing is done.	OFF

Note V must be between 0 and 99,999,999. DSQR will not operate properly if V is not within this range.

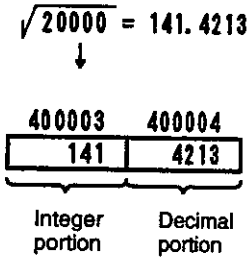
◀EXAMPLE▶

4. Application Example

1) Ladder Programming



2) Operation



For the above DSQR, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. When input relay 100001 turns OFF, output 1 will turn OFF, but the result will remain in holding registers 400003 and 400004.

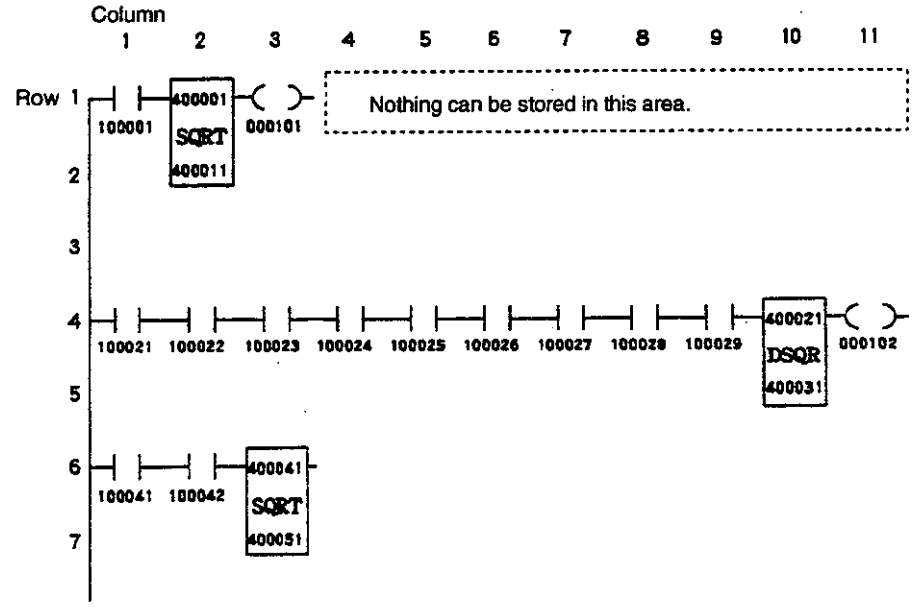
2.7.4 Building Programs

1. Storage Locations on Networks

The decimal square root instructions require two vertical elements on a network, one top element and one bottom element. They can thus be stored anywhere on a 6-row by 10-column matrix (rows 1 through 6 and columns 1 through 10) on the network.

Note The decimal square root instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

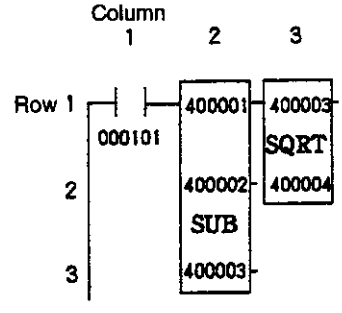
Example



2. Inputs

Inputs to decimal square root instruction can be connected to relay elements (except coils) and/or outputs from timers, counters, math instructions, data transfer instructions, other instructions, etc.

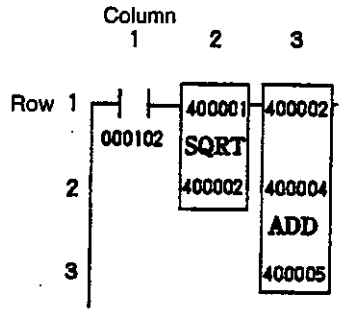
Example



3. Outputs

Outputs from decimal square root instruction can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data transfer instructions, etc.

Example



2.8 Decimal Trigonometric Instruction

This section describes the functions, structures, and operation of the decimal trigonometric instructions and provides simple examples of their application.

2.8.1 Instructions

Decimal trigonometric instructions find the sine and cosine of a specified angle, θ. The instructions that are available are shown in *Table 2.41*.

Table 2.41 Decimal Trigonometric Instructions

Name	Symbol	Operation	θ	Result
DECIMAL SINE	SIN	SINθ	0.0000° to 360.0000°	0.0000 to 1.0000
DECIMAL COSINE	COS	COSθ		

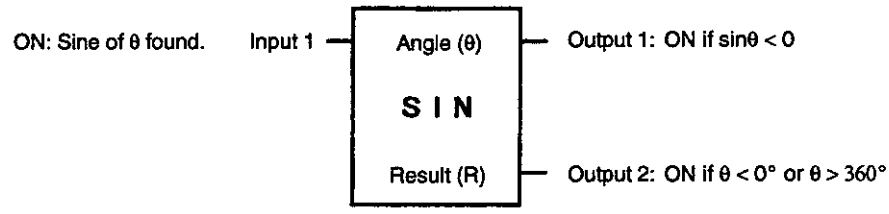
Note The value for which the sine or cosine is to be found (θ) must be an angle. The decimal trigonometric instructions will not operate properly if the value is in radians.

2.8.2 DECIMAL SINE (SIN)

1. Function

The sine of an angle, θ, between 0° and 360° is found.

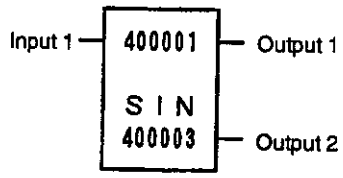
2. Structure



1) SIN is the symbol for DECIMAL SINE.

2) SIN requires two elements, one top element and one bottom element, located vertically on the network. *Table 2.42* lists the register reference numbers that can be specified.

Example



400001: Stores the value of θ as follows:

400001	Integer portion of θ
400002	Decimal portion of θ

400003: Stores the sine of θ as follows:

400003	Integer portion of result
400004	Decimal portion of result

Table 2.42 Structural Elements of SIN

Element	Meaning	Possible Settings
Top (θ)	The contents of the specified register and the next register is used as θ as shown below. θ must be between 0.0000° and 360.0000°. In the example, "400001" was specified for the bottom element. 400001 Integer portion of θ 400002 Decimal portion of θ	Input register: 300001 to 300511 (Z00001 to Z00511) Holding register: 400001 to 409998 (W00001 to W09998) Constant register: 700001 to 704095 (K00001 to K04095) Link register: R10001 to R11023 R20001 to R21023
Bottom (R)	The result is stored in registers as shown below. The decimal portion is truncated after the 4th decimal place. In the example, "400003" was specified for the bottom element. 400003 Integer portion of |sinθ| 400004 Decimal portion of |sinθ|	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 R20001 to R21023

3. Operation

1) SIN will find the sine of θ when input 1 is ON and process the result as follows:

   a) The integer portion of the absolute value of the sine is stored in R and the decimal portion is stored in R+1. The decimal portion is truncated after the 4th decimal place.

   b) If $\sin\theta \geq 0$, outputs 1 and 2 remain OFF.

   c) If $\sin\theta < 0$, output 1 turns ON and output 2 remains OFF.

   d) If $\theta < 0$ or $\theta > 360°$, the operation cannot be performed and output 2 turns ON.

2) The result in R and R+1 remains even if input 1 turns OFF.

3) The operation of SIN is summarized in the following table.

Table 2.43 Operation of SIN

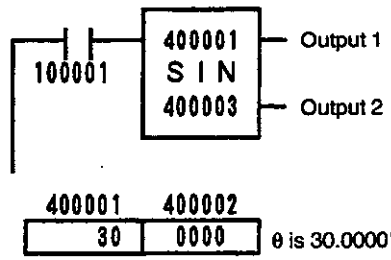
Input 1	Condition	Operation	Outputs	
			1	2
ON	0.0000° ≤ θ ≤ 180.0000°	The absolute value of the sine is stored in R and R+1. The decimal portion is truncated after the 4th decimal place.	OFF	OFF
	180.0001° ≤ θ ≤ 359.9999°	R: Integer portion of |sinθ| / R+1: Decimal portion of |sinθ|	ON	OFF
	θ = 360.0000°		OFF	OFF
	θ < 0 or θ ≥ 360.0001°	Nothing is done.	OFF	ON
OFF	None		OFF	OFF

◀EXAMPLE▶

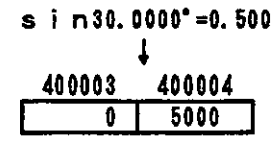
4. Application Examples

Example 1

1) Ladder Programming



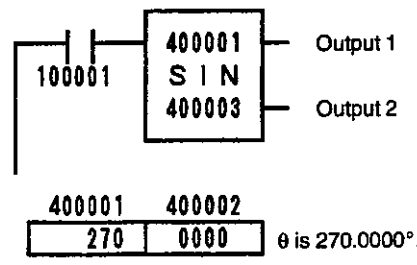
2) Operation



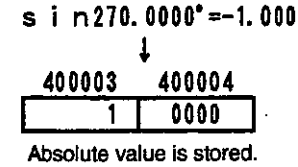
For the above SIN, the operation shown at the right will be performed when input relay 100001 is ON. Outputs 1 and 2 will remain OFF. The result will remain in holding registers 400003 and 400004 even after input relay 100001 turns OFF.

Example 2

1) Ladder Programming



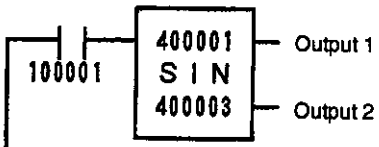
2) Operation

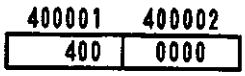


For the above SIN, the operation shown at the right will be performed when input relay 100001 is ON. The result is negative, so output 1 will turn ON and output 2 will remain OFF. When input relay 100001 turns OFF, output 1 will turn OFF, but the result will remain in holding registers 400003 and 400004.

Example 3

1) Ladder Programming





θ is 400.0000°.

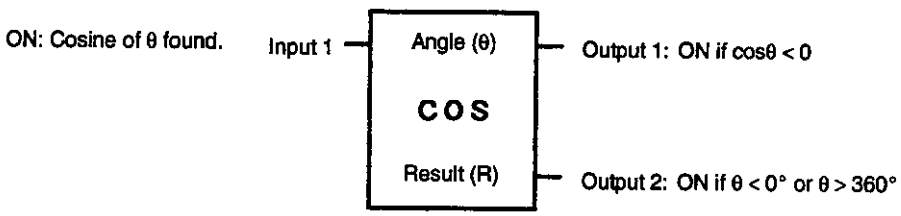
For the above SIN, the value of θ is greater than 360°, so no operation will be performed and output 2 will turn ON.

2.8.3 DECIMAL COSINE (COS)

1. Function

The cosine of an angle, θ, between 0° and 360° is found.

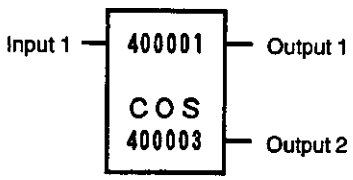
2. Structure



1) COS is the symbol for DECIMAL COSINE.

2) COS requires two elements, one top element and one bottom element, located vertically on the network. *Table 2.44* lists the register reference numbers that can be specified.

Example



400001: Stores the value of θ as follows:

400001	Integer portion of θ
400002	Decimal portion of θ

400003: Stores the cosine of θ as follows:

400003	Integer portion of result
400004	Decimal portion of result

Table 2.44 Structural Elements of COS

Element	Meaning	Possible Settings
Top (θ)	The contents of the specified register and the next register is used as θ as shown below. θ must be between 0.0000° and 360.0000°. In the example, "400001" was specified for the bottom element. 400001 | Integer portion of θ 400002 | Decimal portion of θ	Input register: 300001 to 300511 (Z00001 to Z00511) Holding register: 400001 to 409998 (W00001 to W09998) Constant register: 700001 to 704095 (K00001 to K04095) Link register: R10001 to R11023 R20001 to R21023
Bottom (R)	The result is stored in registers as shown below. The decimal portion is truncated after the 4th decimal place. In the example, "400003" was specified for the bottom element. 400003 | Integer portion of |cosθ| 400004 | Decimal portion of |cosθ|	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 R20001 to R21023

3. Operation

1) COS will find the cosine of θ when input 1 is ON and process the result as follows:

   a) The integer portion of the absolute value of the cosine is stored in R and the decimal portion is stored in R+1. The decimal portion is truncated after the 4th decimal place.

   b) If $\cos\theta \geq 0$, outputs 1 and 2 remain OFF.

   c) If $\cos\theta < 0$, output 1 turns ON and output 2 remains OFF.

   d) If $\theta < 0$ or $\theta \geq 360°$, the operation cannot be performed and output 2 turns ON.

2) The result in R and R+1 remains even if input 1 turns OFF.

3) The operation of COS is summarized in the following table.

Table 2.45 Operation of COS

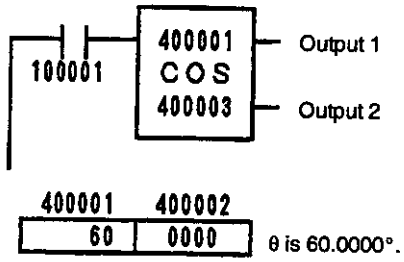
Input 1	Condition	Operation	Outputs 1	Outputs 2
ON	$0.0000° \leq \theta \leq 90.0000°$	The absolute value of the cosine is stored in R and R+1. The decimal portion is truncated after the 4th decimal place. R | Integer portion of result R+1 | Decimal portion of result	OFF	OFF
	$90.0001° \leq \theta \leq 269.9999°$		**ON**	OFF
	$270.0000° \leq \theta \leq 360.0000°$		OFF	OFF
	$\theta < 0°$ or $\theta \geq 360.0001°$	Nothing is done.	OFF	**ON**
OFF	None		OFF	OFF

◀EXAMPLE▶

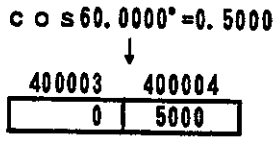
4. Application Examples

Example 1

1) Ladder Programming



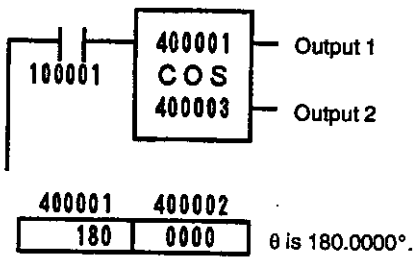
2) Operation



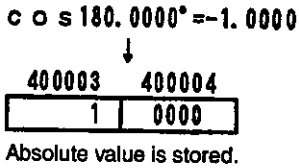
For the above COS, the operation shown at the right will be performed when input relay 100001 is ON. Outputs 1 and 2 will remain OFF. The result will remain in holding registers 400003 and 400004 even after input relay 100001 turns OFF.

Example 2

1) Ladder Programming



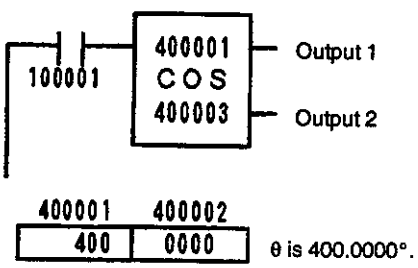
2) Operation



For the above COS, the operation shown at the right will be performed when input relay 100001 is ON. The result is negative, so output 1 will turn ON and output 2 will remain OFF. When input relay 100001 turns OFF, output 1 will turn OFF, but the result will remain in holding registers 400003 and 400004.

Example 3

1) Ladder Programming



For the above COS, the value of θ is greater than 360°, so no operation will be performed and output 2 will turn ON.

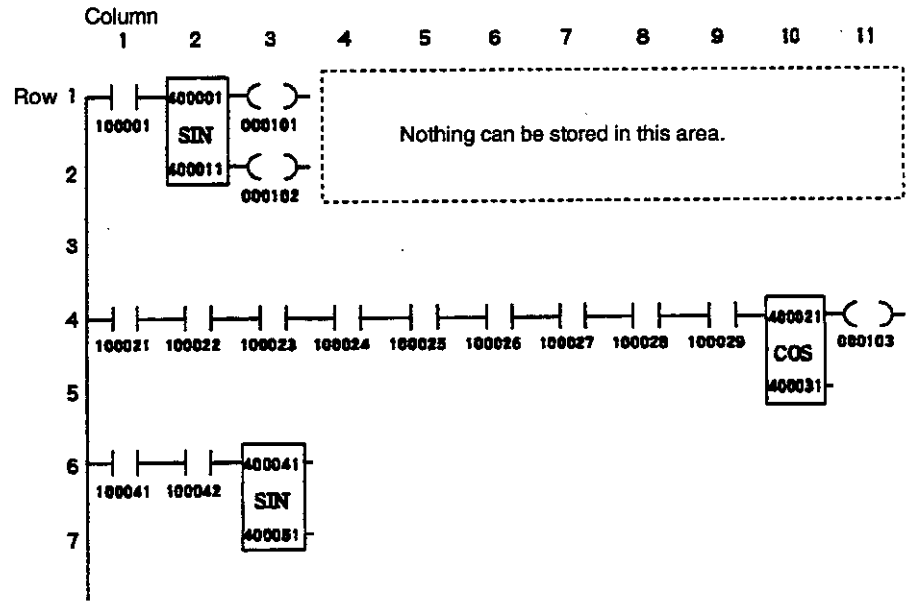
2.8.4 Building Programs

1. Storage Locations on Networks

The decimal trigonometric instructions require two vertical elements on a network, one top element and one bottom element. They can thus be stored anywhere on a 6-row by 10-column matrix (rows 1 through 6 and columns 1 through 10) on the network.

Note The decimal trigonometric instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

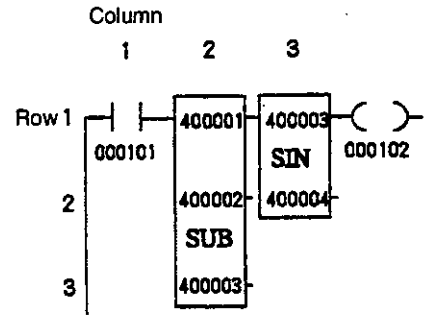
Example



2. Inputs

Inputs to decimal trigonometric instruction can be connected to relay elements (except coils) and/or outputs from timers, counters, math instructions, data transfer instructions, other instructions, etc.

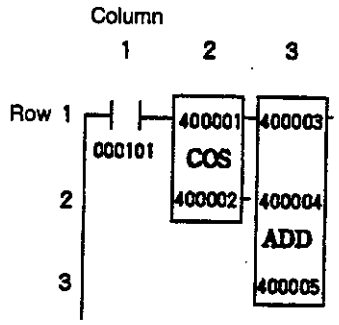
Example



3. Outputs

Outputs from decimal trigonometric instruction can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data transfer instructions, etc.

Example



2.9 Sixteen-bit Arithmetic Instructions

This section describes the functions, structures, and operation of the 16-bit arithmetic instructions and provides simple examples of their application.

2.9.1 Instructions

Sixteen-bit arithmetic instructions perform unsigned or signed addition, subtraction, multiplication, and division on two 16-bit binary numbers, V1 and V2. A negative number is treated as its two's complement. The instructions that are available are shown in *Table 2.46*.

Table 2.46 Sixteen-bit Arithmetic Instructions

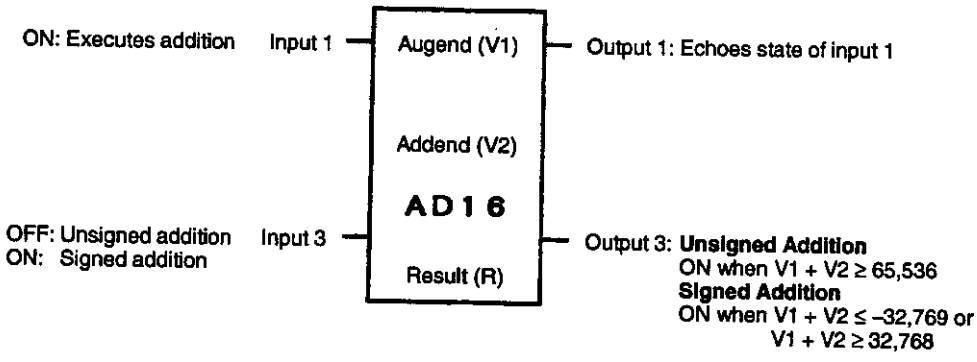
Name	Symbol	Operands	V1	V2	Result
16-BIT ADDITION	AD16	V1 + V2	1) Unsigned: 0 to 65,535		
16-BIT SUBTRACTION	SU16	V1 – V2 Comparison	2) Signed: –32,768 to 32,767		
16-BIT MULTIPLICATION	MU16	V1 x V2	1) Unsigned: 0 to 65,535		1) Unsigned: 0 to 4,294,967,295
			2) Signed: –32,768 to 32,767		2) Signed: –2,147,483,648 to 2,147,483,647
16-BIT DIVISION	DV16	V1 ÷ V2	1) Unsigned: 0 to 4,294,967,295	1) Unsigned: 0 to 65,535	
			2) Signed: –2,147,483,648 to 2,147,483,647	2) Signed: –32,768 to 32,767	

2.9.2 16-BIT ADDITION (AD16)

1. Function

Unsigned or signed addition is performed between two 16-bit binary numbers, V1 and V2. A negative number is treated as its two's complement.

2. Structure



1) AD16 is the symbol for 16-BIT ADDITION.

2) AD16 requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.47* lists the register reference numbers and constants that can be specified.

Example

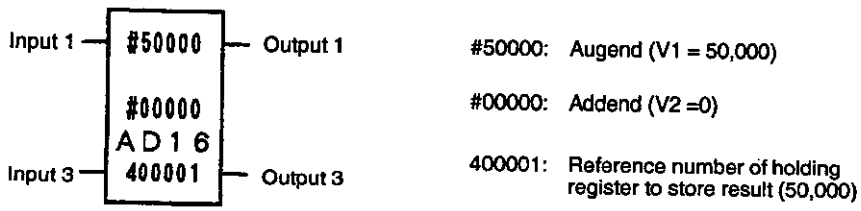


Table 2.47 Structural Elements of AD16

Element	Meaning	Possible Settings
Top (V1)	1) If a constant is specified, its value is used as the augend, V1. If a reference number is specified, the contents of the register is used. 2) V1 must be between the following values: **Unsigned Addition** Between 0 and 65,535 **Signed Addition** Between –32,768 and 32,767	Constant: #00000 to #65535 Input register: 300001 to 300512 (Z00001 to Z00512) Holding register: 400001 to 409999 (W00001 to W09999) Constant register: 700001 to 704096 (K00001 to K04096) Link register: R10001 to R11024 R20001 to R21024
Middle (V2)	1) If a constant is specified, its value is used as the addend, V2. If a reference number is specified, the contents of the register is used. 2) V2 must be within the same ranges as V1.	
Bottom (R)	1) The result is stored in the register. 2) The result must be within the same ranges as V1.	Holding register: 400001 to 409999 (W00001 to W09999) Link register: R10001 to R11024 R20001 to R21024

Note If the value of the top or middle element is between 32,768 and 65,535 for signed addition, the numbers will be handled as two's complements, i.e., between –32,768 and –1.

3. Operation

1) AD16 adds the 16-bit binary values in V2 to V1 when input 1 is ON and process the result as follows:

   a) If input 3 is OFF, 16-bit unsigned addition is performed as follows:

      (1) If $0 \leq V1 + V2 \leq 65,535$:

      - The result of V1 + V2 is stored in R.
      - Output 1 turns ON and output 3 remains OFF.

      (2) If $65,536 \leq V1 + V2$:

      - The result of V1 + V2– 65,536 is stored in R.
      - Outputs 1 and 3 turn ON.

   b) If input 3 is ON, 16-bit signed addition is performed as follows:

      (1) If $-32,768 \leq V1 + V2 \leq 32,767$:

      - The result of V1 + V2 is stored in R.
      - Output 1 turns ON and output 3 remains OFF.

(2) If 32,768 ≤ V1 + V2:

- The result of V1 + V2– 65,536 is stored in R.
- Outputs 1 and 3 turn ON.

(3) If V1 + V2 ≤ –32,769:

- The result of V1 + V2 + 65,536 is stored in R.
- Outputs 1 and 3 turn ON.

2) The result remains in R even if input 1 turns OFF.

3) The operation of AD16 is summarized in the following table.

Table 2.48 Operation of AD16

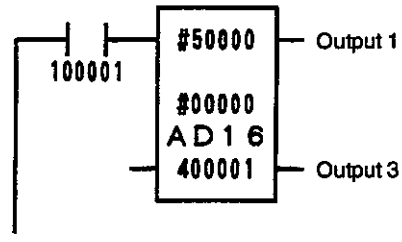
Inputs		Condition	Operation	Outputs	
1	3			1	3
ON	OFF	0 ≤ V1 + V2 ≤ 65,535	V1 + V2 stored in R.	ON	OFF
		65,536 ≤ V1 + V2	V1 + V2 – 65,536 stored in R.		ON
	ON	–32,768 ≤ V1 + V2 ≤ 32,767	V1 + V2 stored in R.		OFF
		32,768 ≤ V1 + V2	V1 + V2 – 65,536 stored in R.		ON
		V1 + V2 ≤ –32,769	V1 + V2 + 65,536 stored in R.		
OFF	Any	None	Nothing is done.	OFF	OFF

◀EXAMPLE▶

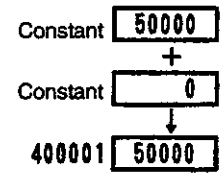
4. Application Examples

Example 1: Unsigned 16-bit Addition

1) Ladder Programming



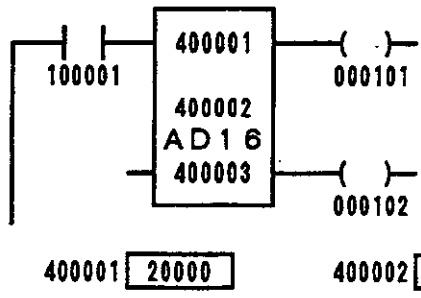
2) Operation



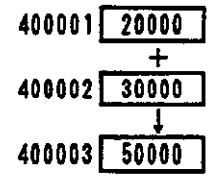
For the above AD16, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. Output 3 will remain OFF, and the result will remain in holding register 400001 even after input relay 100001 turns OFF.

Example 2: Signed 16-bit Addition

1) Ladder Programming



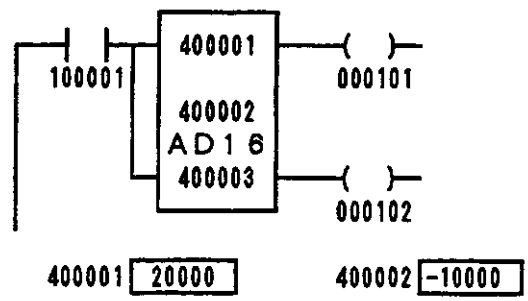
2) Operation



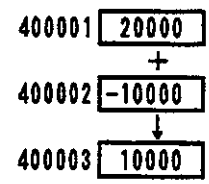
For the above AD16, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON. Coil 000102 will remain OFF, and the result will remain in holding register 400003 even after input relay 100001 turns OFF.

Example 3: Signed 16-bit Addition

1) Ladder Programming



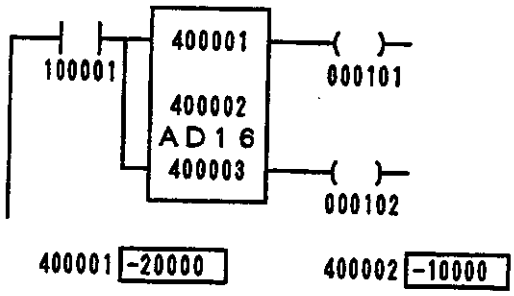
2) Operation



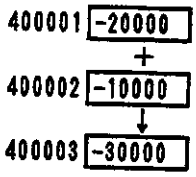
For the above AD16, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 turns ON. Coil 000102 will remain OFF and the result will remain in holding register 400003 even after input relay 100001 turns OFF.

Example 4: Signed 16-bit Addition

1) Ladder Programming



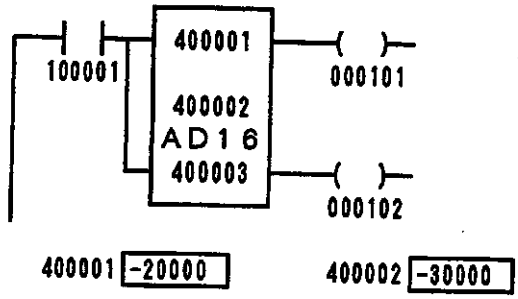
2) Operation



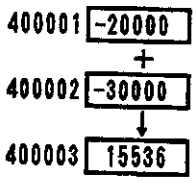
For the above AD16, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON. Coil 000102 will remain OFF and the result will remain in holding register 400003 even after input relay 100001 turns OFF.

Example 5: Signed 16-bit Addition with Overflow in Result

1) Ladder Programming



2) Operation



(-20000) + (-30000) +65536
=15536

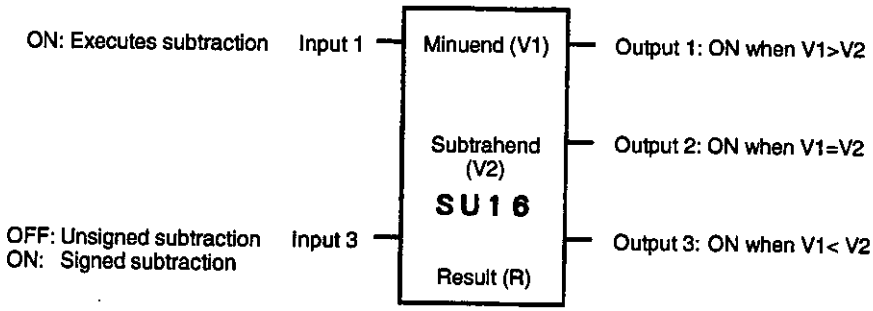
For the above AD16, the operation shown at the right will be performed when input relay 100001 is ON, and coils 000101 and 000102 will turn ON. The result will remain in holding register 400003 even after input relay 100001 turns OFF.

2.9.3 16-BIT SUBTRACTION (SU16)

1. Function

Unsigned or signed subtraction is performed between two 16-bit binary numbers, V1 and V2. A negative number is treated as its two's complement.

2. Structure



1) SU16 is the symbol for 16-BIT SUBTRACTION.

2) SU16 requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.49* lists the register reference numbers and constants that can be specified.

Example

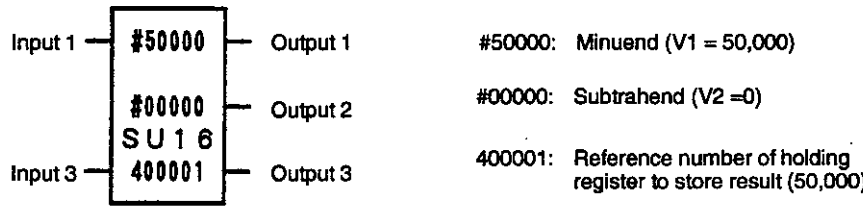


Table 2.49 Structural Elements of SU16

Element	Meaning	Possible Settings	
Top (V1)	1) If a constant is specified, its value is used as the minuend, V1. If a reference number is specified, the contents of the register is used. 2) V1 must be between the following values: **Unsigned Subtraction** Between 0 and 65,535 **Signed Subtraction** Between –32,768 and 32,767	Constant: Input register: Holding register: Constant register: Link register:	#00000 to #65535 300001 to 300512 (Z00001 to Z00512) 400001 to 409999 (W00001 to W09999) 700001 to 704096 (K00001 to K04096) R10001 to R11024 R20001 to R21024
Middle (V2)	1) If a constant is specified, its value is used as the subtrahend, V2. If a reference number is specified, the contents of the register is used. 2) V2 must be within the same ranges as V1.		
Bottom (R)	1) The result is stored in the register. 2) The result must be within the same ranges as V1.	Holding register: Link register:	400001 to 409999 (W00001 to W09999) R10001 to R11024 R20001 to R21024

Note If the value of the top or middle element is between 32,768 and 65,535 for signed subtraction, the numbers will be handled as two's complements, i.e., between –32,768 and –1.

3. Operation

1) SU16 subtracts the 16-bit binary value in V2 from the 16-bit binary value V1 when input 1 is ON and process the result as follows:

a) If input 3 is OFF, 16-bit unsigned subtraction is performed as follows:

(1) If $0 \leq V1 - V2 \leq 65,535$, the result of V1 – V2 is stored in R.

(2) If $V1 - V2 \leq -1$, the result of V1 – V2 + 65,536 is stored in R.

b) If input 3 is ON, 16-bit signed subtraction is performed as follows:

(1) If $-32,768 \leq V1 - V2 \leq 32,767$, the result of V1 – V2 is stored in R.

(2) If $32,768 \leq V1 - V2$, the result of V1 – V2 – 65,536 is stored in R.

(3) If $V1 - V2 \leq -32,769$, the result of V1 – V2 + 65,536 is stored in R.

c) The outputs are treated as follows regardless of the status of input 3:

- If V1 > V2, output 1 turns ON.
- If V1 = V2, output 2 turns ON.
- If V1 < V2, output 3 turns ON.

2) The result remains in R even if input 1 turns OFF.

3) The operation of SU16 is summarized in *Table 2.50.*

Table 2.50 Operation of SU16

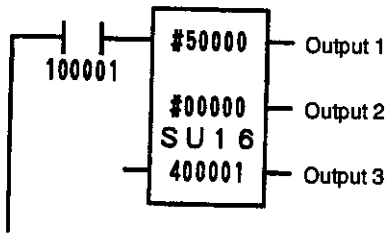
Inputs		Condition	Operation	Outputs 1 to 3
1	3			
ON	OFF	$0 \leq V1 - V2 \leq 65,535$	V1 – V2 stored in R.	• If V1>V2, output 1 turns ON.
		$V1 - V2 \leq -1$	V1 – V2 + 65,536 stored in R.	
	ON	$-32,768 \leq V1 - V2 \leq 32,767$	V1 – V2 stored in R.	• If V1=V2, output 2 turns ON.
		$32,768 \leq V1 - V2$	V1 – V2 – 65,536 stored in R.	
		$V1 - V2 \leq -32,769$	V1 – V2 + 65,536 stored in R.	• If V1<V2, output 3 turns ON.
OFF	Any	None	Nothing is done.	OFF

◀EXAMPLE▶

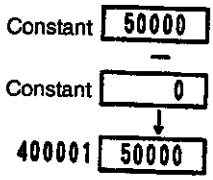
4. Application Examples

Example 1: Unsigned 16-bit Subtraction

1) Ladder Programming



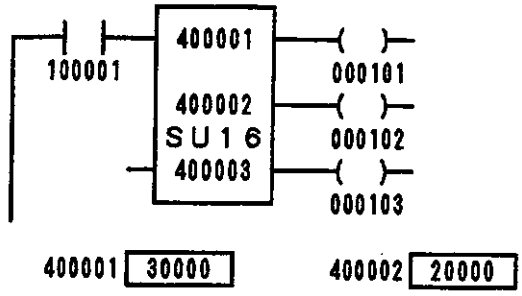
2) Operation



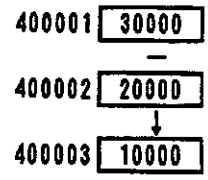
For the above SU16, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. Outputs 2 and 3 will remain OFF, and the result will remain in holding register 400001 even after input relay 100001 turns OFF.

Example 2: Unsigned 16-bit Subtraction

1) Ladder Programming



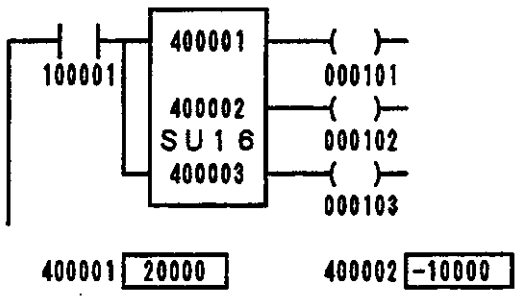
2) Operation



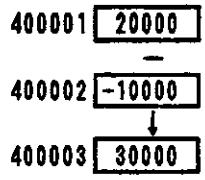
For the above SU16, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON. Coils 000102 and 000103 will remain OFF, and the result will remain in holding register 400003 even after input relay 100001 turns OFF.

Example 3: Signed 16-bit Subtraction

1) Ladder Programming



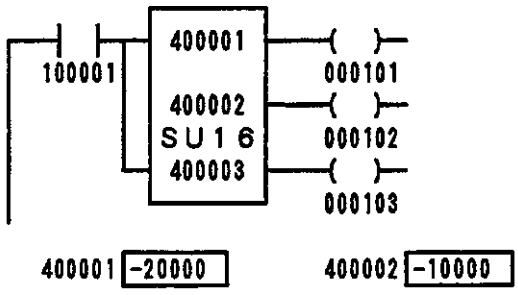
2) Operation



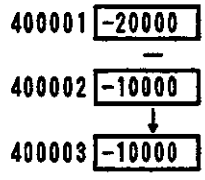
For the above SU16, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 turn ON. Coils 000102 and 000103 will remain OFF, and the result will remain in holding register 400003 even after input relay 100001 turns OFF.

Example 4: Signed 16-bit Subtraction

1) Ladder Programming



2) Operation

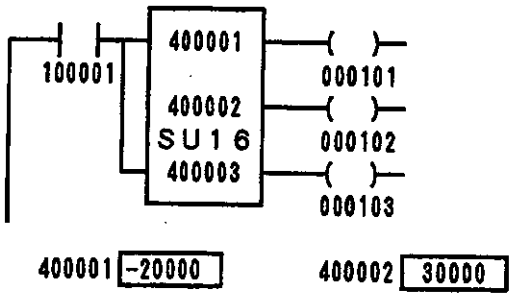


For the above SU16, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000103 will turn ON. Coils 000101 and 000102 will remain OFF,

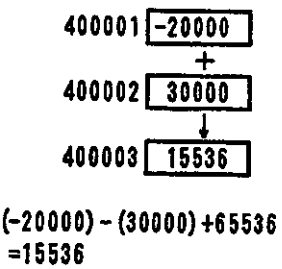
and the result will remain in holding register 400003 even after input relay 100001 turns OFF.

Example 5: Signed 16-bit Subtraction with Overflow in Result

1) Ladder Programming



2) Operation



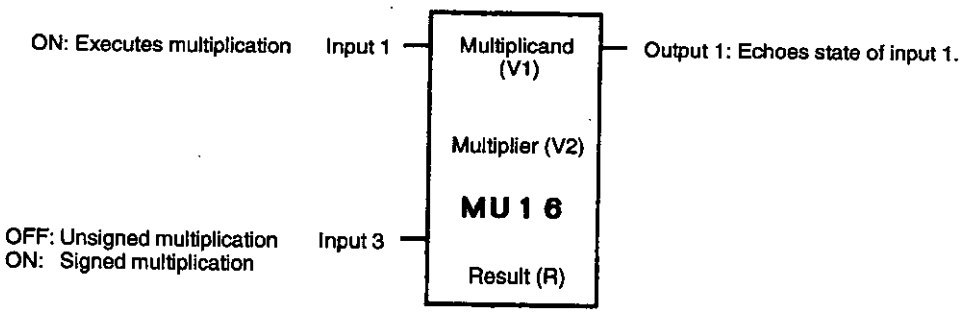
For the above SU16, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000103 will turn ON. Coils 000101 and 000102 will turn OFF, and the result will remain in holding register 400003 even after input relay 100001 turns OFF.

2.9.4 16-BIT MULTIPLICATION (MU16)

1. Function

Unsigned or signed multiplication is performed between two 16-bit binary numbers, V1 and V2. A negative number is treated as its two's complement.

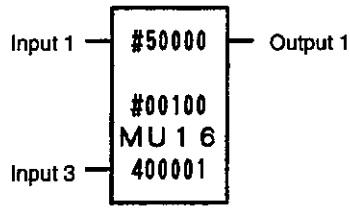
2. Structure



1) MU16 is the symbol for 16-BIT MULTIPLICATION.

2) MU16 requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.51* lists the register reference numbers and constants that can be specified.

Example



#50000: Multiplicand (50,000)

#00100: Multiplier (100)

400001: Stores the result (5,000,000) as follows:

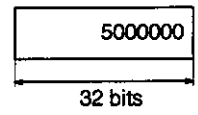


Table 2.51 Structural Elements of MU16

Element	Meaning	Possible Settings
Top (V1)	1) If a constant is specified, its value is used as the multiplicand, V1. If a reference number is specified, the contents of the register is used. 2) V1 must be between the following values: **Unsigned Multiplication** Between 0 and 65,535 **Signed Multiplication** Between –32,768 and 32,767	Constant: #00000 to #65535 Input register: 300001 to 300512 (Z00001 to Z00512) Holding register: 400001 to 409999 (W00001 to W09999) Constant register: 700001 to 704096 (K00001 to K04096) Link register: R10001 to R11024 R20001 to R21024
Middle (V2)	1) If a constant is specified, its value is used as the multiplier, V2. If a reference number is specified, the contents of the register is used. 2) V2 must be within the same ranges as V1.	
Bottom (R)	1) The result is stored in two consecutive registers, as shown in the following example. 2) The range of the result is the range that can be expressed with a 32-bit binary number. The value of the result must thus be between 0 and 4,294,967,295 for unsigned multiplication and between –2,147,483,648 and 2,147,483,647 for signed multiplication. 3) In the example, "400001" was specified for the bottom element. 400002 Upper 16 bits 400001 Lower 16 bits	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 R20001 to R21023

Note If the value of the top or middle element is between 32,768 and 65,535 for signed multiplication, the numbers will be handled as two's complements, i.e., between –32,768 and –1.

3. Operation

1) MU16 multiplies the 16-bit binary values in V2 to V1 when input 1 is ON and process the result as follows:

   a) If input 3 is OFF, 16-bit unsigned multiplication is performed and the upper 16 bits of the result of V1 x V2 are stored in R+1 and the lower 16 bits are stored in R.

   b) If input 3 is ON, 16-bit signed multiplication is performed and the upper 16 bits of the result of V1 x V2 are stored in R+1 and the lower 16 bits are stored in R.

   c) Output 1 is ON while input 1 is ON and is not affected by the status of input 3.

2) The result remains in R and R+1 even if input 1 turns OFF.

3) The operation of MU16 is summarized in *Table 2.52.*

Table 2.52 Operation of MU16

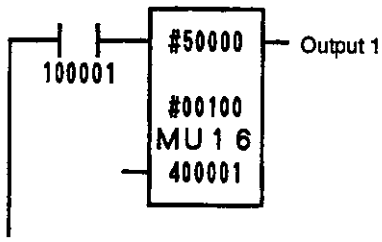
Inputs		Condition	Operation	Output 1	Remarks
1	3				
ON	OFF	None	V1 x V2 stored in R+1 and R.	ON	Unsigned multiplication is performed.
	ON				Signed multiplication is performed.
OFF	Any		Nothing is done.	OFF	---

◀EXAMPLE▶

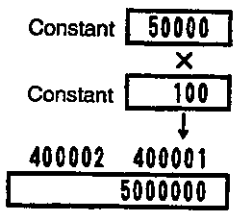
4. Application Examples

Example 1: Unsigned 16-bit Multiplication

1) Ladder Programming



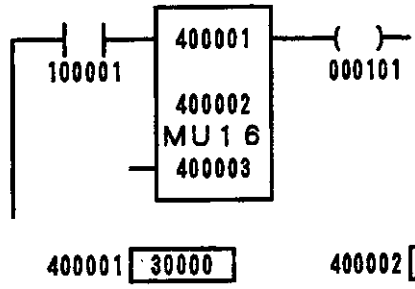
2) Operation



For the above MU16, the operation shown at the right will be performed when input relay 100001 is ON, and output 1 will turn ON. The result will remain in holding registers 400001 and 400002 even after input relay 100001 turns OFF.

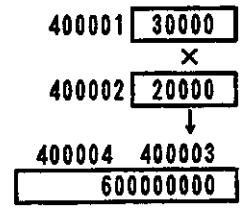
Example 2: Signed 16-bit Multiplication

1) Ladder Programming



400001 30000 400002 20000

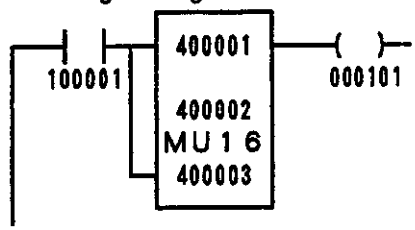
2) Operation



For the above MU16, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON. The result will remain in holding registers 400003 and 400004 even after input relay 100001 turns OFF.

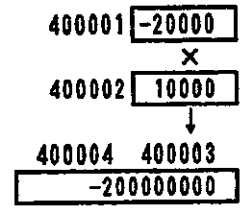
Example 3: Signed 16-bit Multiplication

1) Ladder Programming



400001 -20000 400002 10000

2) Operation



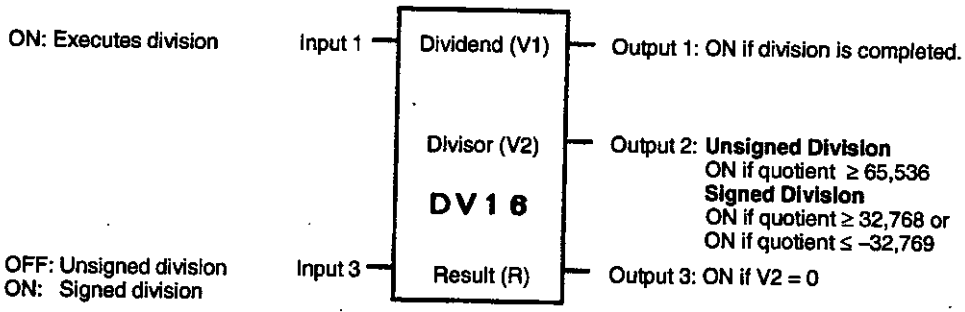
For the above MU16, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 turn ON. The result will remain in holding registers 400003 and 400004 even after input relay 100001 turns OFF.

2.9.5 16-BIT DIVISION (DV16)

1. Function

Unsigned or signed division is performed between two 16-bit binary numbers, V1 and V2. A negative number is treated as its two's complement. The remainder is also found.

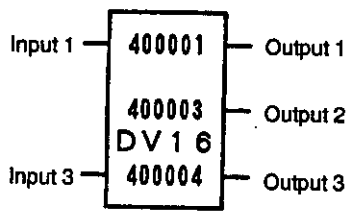
2. Structure



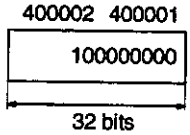
1) DV16 is the symbol for 16-BIT DIVISION.

2) DV16 requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.53* lists the register reference numbers and constants that can be specified.

Example



400001: Stores the dividend, V1, as follows:



400003: Stores the divisor, V2.

400004: Stores the result as follows:

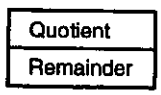


Table 2.53 Structural Elements of DV16

Element	Meaning	Possible Settings
Top (V1)	1) If a constant is specified, its value is used as the dividend, V1. The range of V1 is the range that can be expressed with a 16-bit binary number, i.e., between 0 and 65,535 for unsigned division and between –32,768 and 32,767 for signed division. 2) If a reference number is specified, the contents of the specified register and the next reference is used. The range of the result is the range that can be expressed with a 32-bit binary number, i.e., between 0 and 4,294,967,295 for unsigned division and between –2,147,483,648 and 2,147,483,647 for signed division. 3) In the example, "400001" was specified for the top element, 400002 Upper 16 bits 400001 Lower 16 bits	Constant: #00000 to #65535 Input register: 300001 to 300511 (Z00001 to Z00511) Holding register: 400001 to 409998 (W00001 to W09998) Constant register: 700001 to 704095 (K00001 to K04095) Link register: R10001 to R11023 R20001 to R21023
Middle (V2)	1) If a constant is specified, its value is used as the divisor, V2. 2) If a reference number is specified, the contents of the register is used. 3) The range of the result is the range that can be expressed with a 16-bit binary number, i.e., between 0 and 65,535 for unsigned division and between –32,768 and 32,767 for signed division.	Input register: 300001 to 300512 (Z00001 to Z00512) Holding register: 400001 to 409999 (W00001 to W09999) Constant register: 700001 to 704096 (K00001 to K04096) Link register: R10001 to R11024 R20001 to R21024
Bottom (R)	1) The result is stored in two consecutive registers, as shown in the following example. 2) The range of the result is the same as that of V2. 3) In the example, "400004" was specified for the bottom element. 400005 Quotient 400004 Remainder	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 R20001 to R21023

Note If the value of the top or middle element is between 32,768 and 65,535 for signed division, the numbers will be handled as two's complements, i.e., between –32,768 and –1.

3. Operation

1) DV16 divides the 16-bit binary value in V1 by the 16-bit binary value in V2 when input 1 is ON and process the result as follows:

   a) If input 3 is OFF, 16-bit unsigned division is performed as follows:

      (1) If $0 \leq V1 \div V2 \leq 65,535$:

      - The quotient and remainder of V1 ÷ V2 is stored in R+1 and R.
      - Output 1 turns ON.

      (2) Division will not be executed in the following cases and zero (0) is stored in R and R+1.

      - V2 = 0. In this case, output 3 turns ON.
      - If the quotient is not between 0 and 65,535. In this case, output 2 turns ON.

   b) If input 3 is ON, 16-bit signed division is performed as follows:

      (1) If $-32,768 \leq V1 \div V2 \leq 32,767$:

      - The quotient and remainder of V1 ÷ V2 is stored in R+1 and R.
      - Output 1 turns ON.

      (2) Division will not be executed in the following cases and zero (0) is stored in R and R+1.

      - V2 = 0. In this case, output 3 turns ON.
      - If the quotient is not between –32,768 and 32,767. In this case, output 2 turns ON.

2) The result remains in R+1 and R even if input 1 turns OFF.

3) The operation of DV16 is summarized in *Table 2.54.*

Table 2.54 Operation of DV16

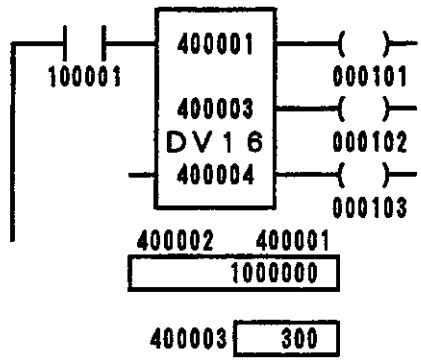
Inputs		Condition	Operation	Outputs		
1	3			1	2	3
ON	OFF	0 ≤ V1 ÷ V2 ≤ 65,535	V1 + V2 stored as follows: R+1 Quotient R Remainder	**ON**	OFF	OFF
		V1 ÷ V2 >65,535 or V1 ÷ V2 < 0	Accurate division not possible. Zero (0) stored in R+1 and R.	OFF	**ON**	OFF
		V2 = 0	Execution not possible. Zero (0) stored in R+1 and R.	OFF	OFF	**ON**
	ON	-32,768 ≤ V1 ÷ V2 ≤ 32,767	V1 + V2 stored as follows: R+1 Quotient R Remainder	**ON**	OFF	OFF
		V1 ÷ V2 < -32,768 or V1 ÷ V2 > 32,767	Accurate division not possible. Zero (0) stored in R+1 and R.	OFF	**ON**	OFF
		V2 = 0	Execution not possible. Zero (0) stored in R+1 and R.	OFF	OFF	**ON**
OFF	Any	None	Nothing is done.	OFF		

◀EXAMPLE▶

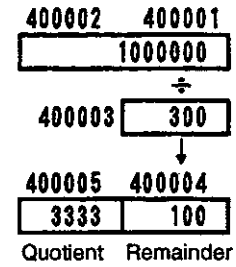
4. Application Examples

Example 1: Unsigned 16-bit Division

1) Ladder Programming



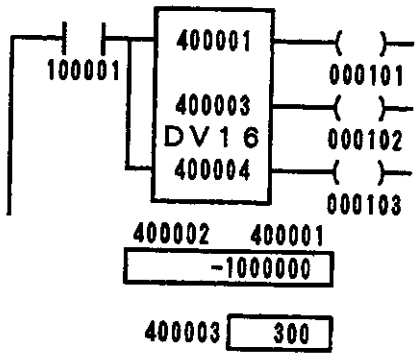
2) Operation



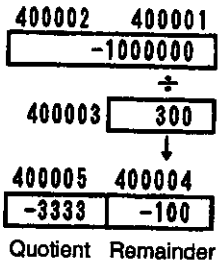
For the above DV16, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON. Outputs 2 and 3 will remain OFF. When input relay 100001 turns OFF, coil 000101 will turn OFF and the result will remain in holding registers 400004 and 400005.

Example 2: Signed 16-bit Division

1) Ladder Programming



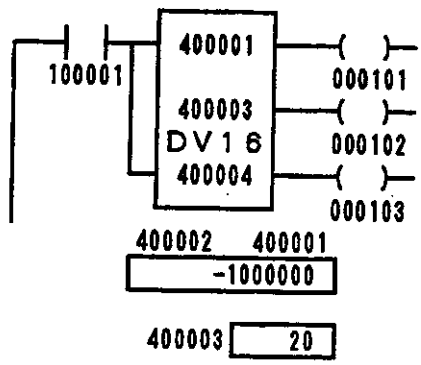
2) Operation



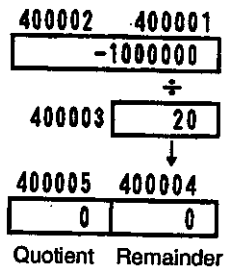
For the above DV16, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON. Outputs 2 and 3 will remain OFF. When input relay 100001 turns OFF, coil 000101 will turn OFF and the result will remain in holding registers 400004 and 400005.

Example 3: Signed 16-bit Division

1) Ladder Programming



2) Operation



-1000000÷20=-50000≥32768

For the above DV16, the operation shown at the right will be performed when input relay 100001 is ON, zeros will be stored in 400004 and 400005 because the quotient exceeds 32,767, and coil 000102 will turn ON. Outputs 1 and 3 will remain OFF.

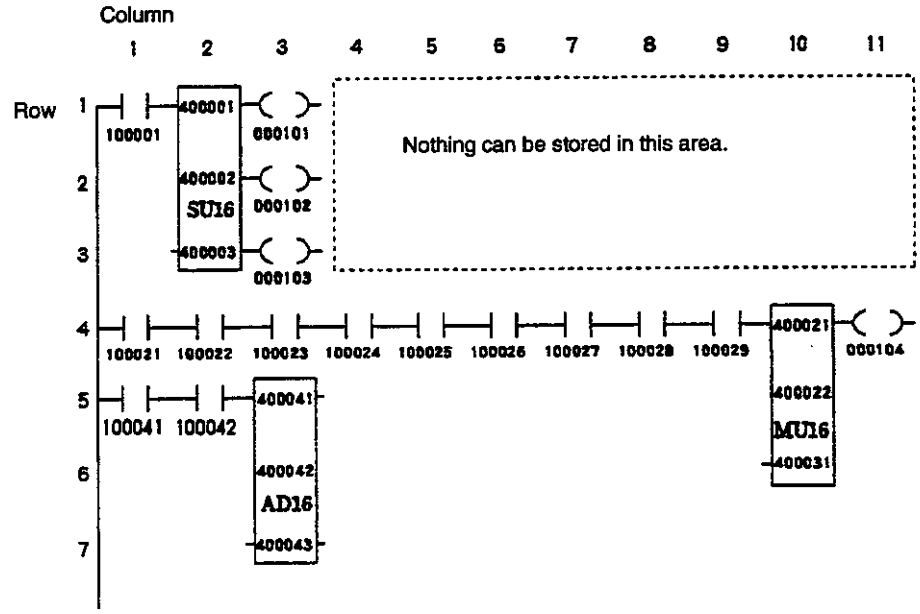
2.9.6 Building Programs

1. Storage Locations on Networks

All 16-bit arithmetic instructions require three vertical elements on a network, one top element, one middle element, and one bottom element. They can thus be stored anywhere on a 5-row by 10-column matrix (rows 1 through 5 and columns 1 through 10) on the network.

Note Sixteen-bit arithmetic instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

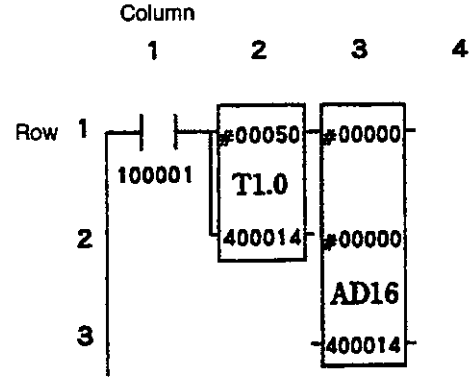
Example



2. Inputs

Inputs to 16-bit math instructions can be connected to relay elements (except coils) and/or outputs from timers, counters, math instructions, data transfer instructions, other instructions, etc.

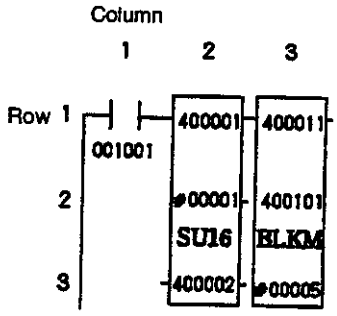
Example



3. Outputs

Outputs from 16-bit math instructions can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data transfer instruction, etc.

Example



2.10 Thirty-two-bit Arithmetic Instructions

This section describes the functions, structures, and operation of the 32-bit arithmetic instructions and provides simple examples of their application.

2.10.1 Instructions

Thirty-two-bit arithmetic instructions perform unsigned or signed addition or subtraction on two 32-bit binary numbers, V1 and V2. A negative number is treated as its two's complement. The instructions that are available are shown in *Table 2.55.*

Table 2.55 Thirty-two-bit Arithmetic Instructions

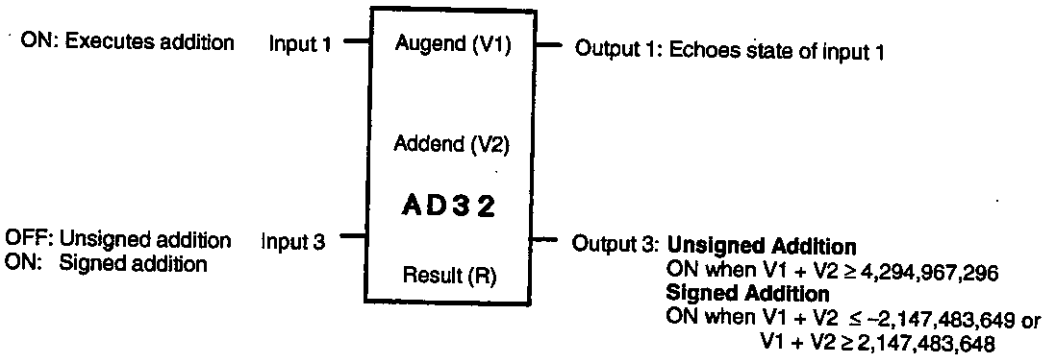
Name	Symbol	Operands	V1	V2	Result
32-BIT ADDITION	AD32	V1 + V2	1) Unsigned: 0 to 4,294,967,295		
32-BIT SUBTRACTION	SU32	V1 – V2 Comparison	2) Signed: –2,147,483,648 to 2,147,483,647		
32-BIT COMPARE	TEST	Comparison	• **16-bit Comparison** 1) Unsigned: 0 to 65,535 2) Signed: –32,768 to 32,767 • **32-bit Comparison** 1) Unsigned: 0 to 4,294,967,295 2) Signed: –2,147,483,648 to 2,147,483,647		

2.10.2 32-BIT ADDITION (AD32)

1. Function

Unsigned or signed addition is performed between two 32-bit binary numbers, V1 and V2. A negative number is treated as its two's complement.

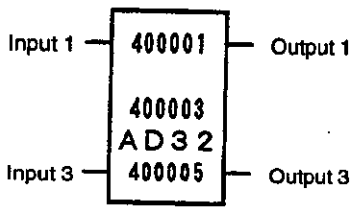
2. Structure



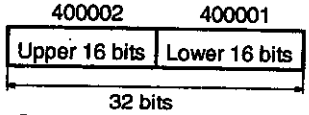
1) AD32 is the symbol for 32-BIT ADDITION.

2) AD32 requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.56* lists the register reference numbers that can be specified.

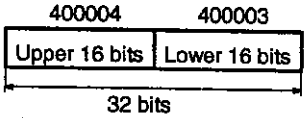
Example



400001: Stores the augend as follows:



400003: Stores the addend as follows:



400005: Stores the result as follows:

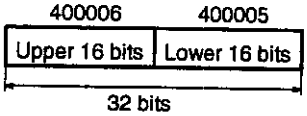


Table 2.56 Structural Elements of AD32

Element	Meaning	Possible Settings
Top (V1)	1) The contents of the specified register and the next register is used as the augend, V1. 2) V1 must be within the range of the result is the range that can be expressed with a 32-bit binary number, i.e., the following ranges: **Unsigned Addition** Between 0 and 4,294,967,295 **Signed Addition** Between –2,147,483,648 and 2,147,483,647 3) In the example, "400001" was specified for the top element. 400002 Upper 16 bits 400001 Lower 16 bits	Input register: 300001 to 300511 (Z00001 to Z00511) Holding register: 400001 to 409998 (W00001 to W09998) Constant register: 700001 to 704095 (K00001 to K04095) Link register: R10001 to R11023 R20001 to R21023
Middle (V2)	1) The contents of the specified register and the next register is used as the addend, V2. 2) V2 must be within the same ranges as V1. 3) In the example, "400003" was specified for the middle element. 400004 Upper 16 bits 400003 Lower 16 bits	
Bottom (R)	1) The result is stored in the specified register and the next register. 2) R must be within the same ranges as V1. 3) In the example, "400005" was specified for the bottom element. 400006 Upper 16 bits 400005 Lower 16 bits	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 R20001 to R21023

3. Operation

1) AD32 adds the 32-bit binary values in V2 to V1 when input 1 is ON and process the result as follows:

a) If input 3 is OFF, 32-bit unsigned addition is performed as follows:

(1) If $0 \leq V1 + V2 \leq 4,294,967,295$:

- The upper 16 bits of result of V1 + V2 are stored in R+1 and the lower 16 bits are stored in R.
- Output 1 turns ON and output 3 remains OFF.

(2) If $4,294,967,296 \leq V1 + V2$:

- The upper 16 bits of result of V1 + V2 – 4,294,967,296 are stored in R+1 and the lower 16 bits are stored in R.
- Outputs 1 and 3 turn ON.

b) If input 3 is ON, 32-bit signed addition is performed as follows:

(1) If $-2,147,483,648 \leq V1 + V2 \leq 2,147,483,647$:

- The upper 16 bits of result of V1 + V2 are stored in R+1 and the lower 16 bits are stored in R.
- Output 1 turns ON and output 3 remains OFF.

(2) If $2,147,483,648 \leq V1 + V2$:

- The upper 16 bits of result of V1 + V2 – 4,294,967,296 are stored in R+1 and the lower 16 bits are stored in R.
- Outputs 1 and 3 turn ON.

(3) If $V1 + V2 \leq -2,147,483,649$:

- The upper 16 bits of result of V1 + V2 + 4,294,967,296 are stored in R+1 and the lower 16 bits are stored in R.
- Outputs 1 and 3 turn ON.

2) The result remains in R and R+1 even if input 1 turns OFF.

3) The operation of AD32 is summarized in *Table 2.57*.

Table 2.57 Operation of AD32

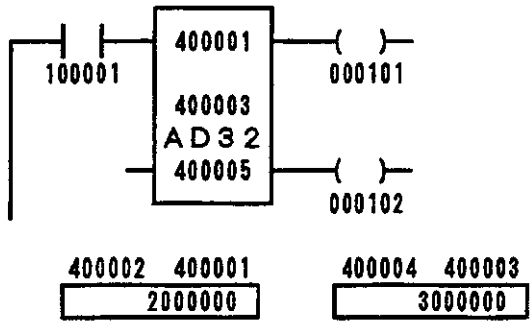
Inputs		Condition	Operation	Outputs	
1	3			1	3
ON	OFF	$0 \leq V1 + V2 \leq 4,294,967,295$	V1 + V2 stored in R+1 and R.	**ON**	OFF
		$4,294,967,296 \leq V1 + V2$ or $V1 + V2 < 0$	V1 + V2 – 4,294,967,296 stored in R+1 and R.		**ON**
	ON	$-2,147,483,648 \leq V1 + V2 \leq 2,147,483,647$	V1 + V2 stored in R+1 and R.		OFF
		$2,147,483,648 \leq V1 + V2$	V1+ V2 – 4,294,967,296 stored in R+1 and R.		**ON**
		$V1 + V2 \leq -2,147,483,649$	V1 + V2 + 4,294,967,296 stored in R+1 and R.		
OFF	Any	None	Nothing is done.	OFF	OFF

◀EXAMPLE▶

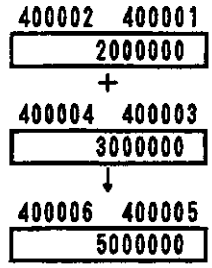
4. Application Examples

Example 1: Unsigned 32-bit Addition

1) Ladder Programming



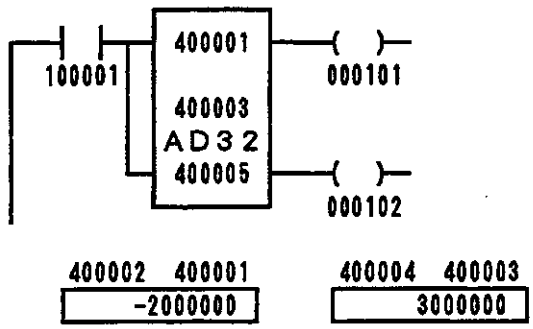
2) Operation



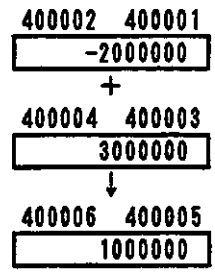
For the above AD32, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON. Coil 000102 will remain OFF. When input relay 100001 turns OFF, coil 000101 will turn OFF and the result will remain in holding registers 400005 and 400006.

Example 2: Signed 32-bit Addition

1) Ladder Programming



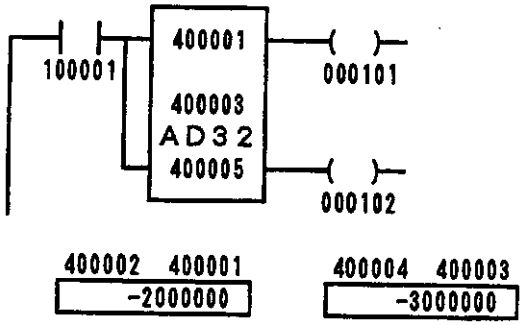
2) Operation



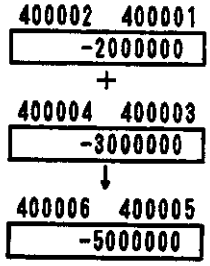
For the above AD32, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON. Coil 000102 will remain OFF. When input relay 100001 turns OFF, coil 000101 will turn OFF and the result will remain in holding registers 400005 and 400006.

Example 3: Signed 32-bit Addition

1) Ladder Programming



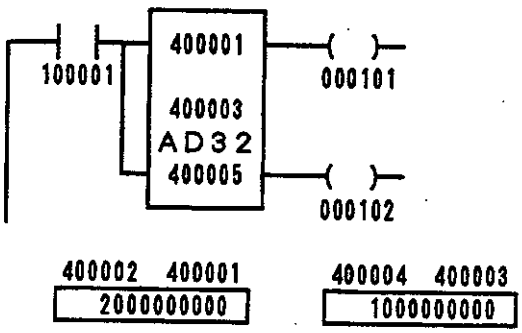
2) Operation



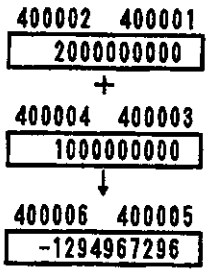
For the above AD32, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON. Coil 000102 will remain OFF. When input relay 100001 turns OFF, coil 000101 will turn OFF and the result will remain in holding registers 400005 and 400006.

Example 4: Signed 32-bit Addition with Overflow in Result

1) Ladder Programming



2) Operation



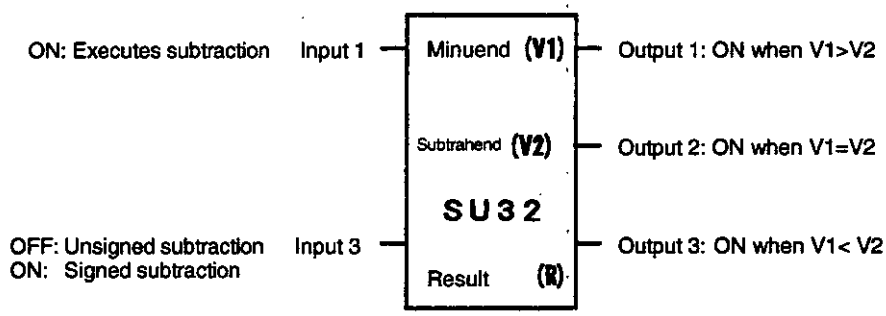
For the above AD32, the operation shown at the right will be performed when input relay 100001 is ON, and coils 000101 and 000102 will turn ON. The result will remain in holding registers 400005 and 400006 even after input relay 100001 turns OFF.

2.10.3 32-BIT SUBTRACTION (SU32)

1. Function

Unsigned or signed subtraction is performed between two 32-bit binary numbers, V1 and V2. A negative number is treated as its two's complement.

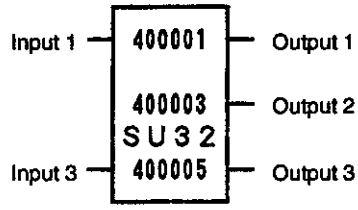
2. Structure



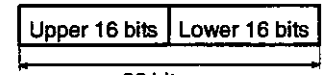
1) SU32 is the symbol for 32-BIT SUBTRACTION.

2) SU32 requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.58* lists the register reference numbers that can be specified.

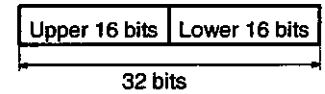
Example



400001: Stores the minuend as follows:



400003: Stores the subtrahend as follows:



400005: Stores the result as follows:

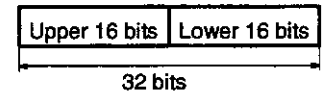


Table 2.58 Structural Elements of SU32

Element	Meaning	Possible Settings
Top (V1)	1) The contents of the specified register and the next register is used as the minuend, V1. 2) V1 must be within the range that can be expressed with a 32-bit binary number, i.e., the following ranges: **Unsigned Subtraction** Between 0 and 4,294,967,295 **Signed Subtraction** Between –2,147,483,648 and 2,147,483,647 3) In the example, "400001" was specified for the top element. 400002 Upper 16 bits 400001 Lower 16 bits	Input register: 300001 to 300511 (Z00001 to Z00511) Holding register: 400001 to 409998 (W00001 to W09998) Constant register: 700001 to 704095 (K00001 to K04095) Link register: R10001 to R11023 R20001 to R21023
Middle (V2)	1) The contents of the specified register and the next register is used as the subtrahend, V2. 2) V2 must be within the same ranges as V1. 3) In the example, "400003" was specified for the middle element. 400004 Upper 16 bits 400003 Lower 16 bits	
Bottom (R)	1) The result is stored in the specified register and the next register. 2) R must be within the same ranges as V1. 3) In the example, "400005" was specified for the bottom element. 400006 Upper 16 bits 400005 Lower 16 bits	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 R20001 to R21023

3. Operation

1) SU32 subtracts the 32-bit binary value in V2 from the 32-bit binary value V1 when input 1 is ON and process the result as follows:

   a) If input 3 is OFF, 32-bit unsigned subtraction is performed as follows:

      (1) If $0 \leq V1 - V2 \leq 4,294,967,295$, the upper 16 bits of result of V1 – V2 are stored in R+1 and the lower 16 bits are stored in R.

      (2) If $V1 - V2 \leq -1$, the upper 16 bits of result of V1 – V2 + 4,294,967,296 are stored in R+1 and the lower 16 bits are stored in R.

   b) If input 3 is ON, 32-bit signed subtraction is performed as follows:

      (1) If $-2,147,483,648 \leq V1 - V2 \leq 2,147,483,647$, the upper 16 bits of result of V1 – V2 are stored in R+1 and the lower 16 bits are stored in R.

      (2) If $2,147,483,648 \leq V1 - V2$, the upper 16 bits of result of V1 – V2 – 4,294,967,296 are stored in R+1 and the lower 16 bits are stored in R.

      (3) If $V1 - V2 \leq -2,147,483,649$, the upper 16 bits of result of V1 – V2 + 4,294,967,296 are stored in R+1 and the lower 16 bits are stored in R.

   c) The outputs are treated as follows regardless of the status of input 3:

      - If V1 > V2, output 1 turns ON.
      - If V1 = V2, output 2 turns ON.
      - If V1 < V2, output 3 turns ON.

2) The result remains in R and R+1 even if input 1 turns OFF.

3) The operation of SU32 is summarized in the following table.

Table 2.59 Operation of SU32

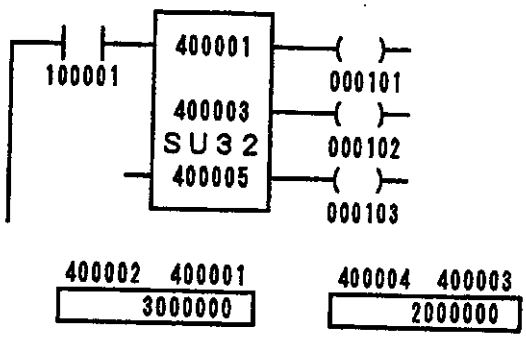
Inputs		Condition	Operation	Outputs 1 to 3
1	3			
ON	OFF	$0 \leq V1 - V2 \leq 4,294,967,295$	V1–V2 stored in R+1 and R.	• If V1>V2, output 1 turns ON. • If V1=V2, output 2 turns ON. • If V1<V2, output 3 turns ON.
		4,294,967,296 $\leq$ V1 – V2 or V1 – V2 < 0	V1–V2+4,294,967,296 stored in R+1 and R.	
	ON	$-2,147,483,648 \leq V1 - V2 \leq 2,147,483,647$	V1–V2 stored in R+1 and R.	
		$2,147,483,648 \leq V1 - V2$	V1–V2–4,294,967,296 stored in R+1 and R.	
		$V1 - V2 \leq -2,147,483,649$	V1–V2+4,294,967,296 stored in R+1 and R.	
OFF	Any	None	Nothing is done.	OFF

◀EXAMPLE▶

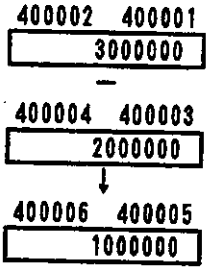
4. Application Examples

Example 1: Unsigned 32-bit Subtraction

1) Ladder Programming



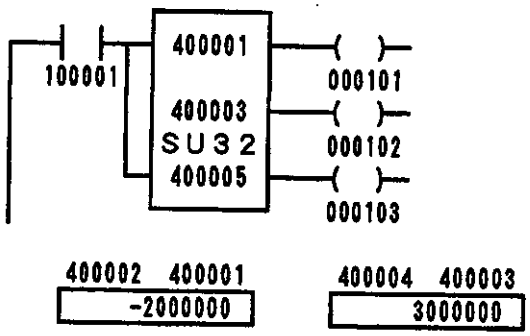
2) Operation



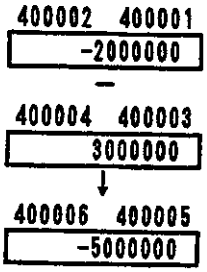
For the above SU32, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON. Coils 000102 and 000103 will remain OFF, and the result will remain in holding registers 400005 and 400006 even after input relay 100001 turns OFF.

Example 2: Signed 32-bit Subtraction

1) Ladder Programming



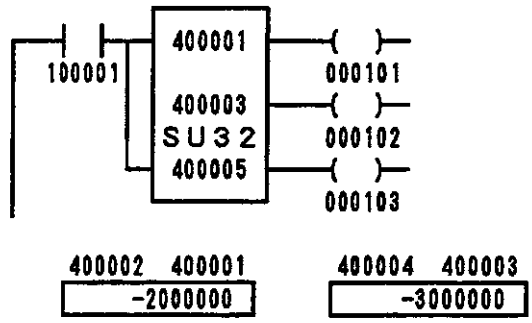
2) Operation



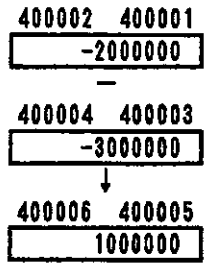
For the above SU32, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000103 will turn ON. Coils 000101 and 000102 will remain OFF, and the result will remain in holding registers 400005 and 400006 even after input relay 100001 turns OFF.

Example 3: Signed 32-bit Subtraction

1) Ladder Programming



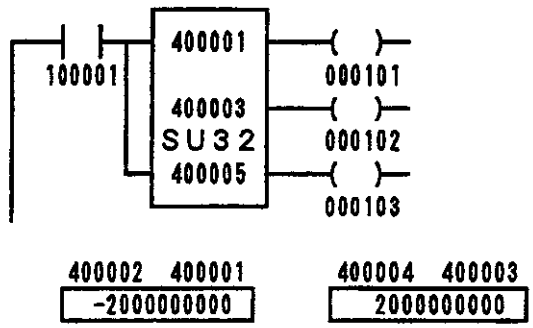
2) Operation



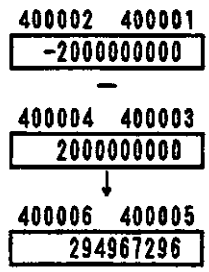
For the above SU32, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000101 turn ON. Coils 000102 and 000103 will remain OFF and the result will remain in holding registers 400005 and 400006 even after input relay 100001 turns OFF.

Example 4: Signed 32-bit Subtraction with Overflow in Result

1) Ladder Programming



2) Operation



For the above SU32, the operation shown at the right will be performed when input relay 100001 is ON, and coil 000103 will turn ON. Coils 000101 and 000102 will turn OFF, and the result will remain in holding registers 400005 and 400006 even after input relay 100001 turns OFF.

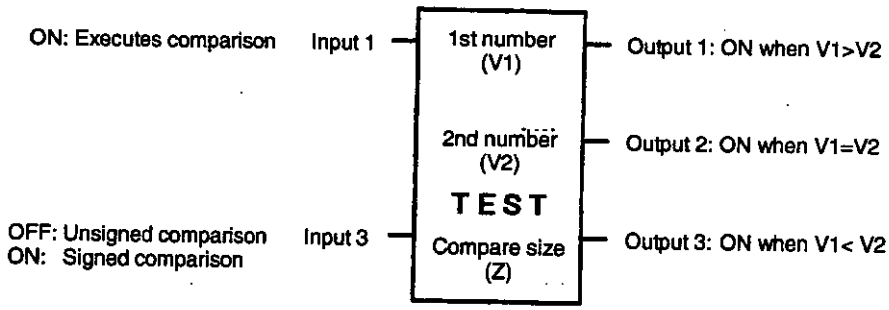
2.10.4 32-BIT COMPARE (TEST)

1. Function

1) **Sixteen-bit Comparison**
Unsigned or signed comparison is performed between two 16-bit binary numbers, V1 and V2. A negative number is treated as its two's complement.

2) **Thirty-two-bit Comparison**
Unsigned or signed comparison is performed between two 32-bit binary numbers, V1 and V2. A negative number is treated as its two's complement.

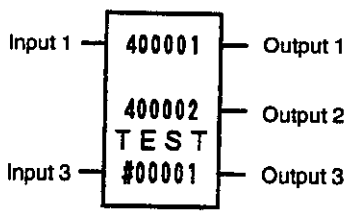
2. Structure



1) TEST is the symbol for 32-BIT COMPARE.

2) TEST requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 2.60* and *table 2.61* list the register reference numbers and constants that can be specified.

Example 1: Sixteen-bit Comparison

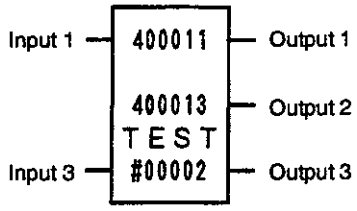


400001: Reference number of the holding register storing V1.

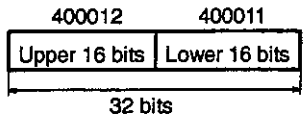
400002: Reference number of the holding register storing V2.

#00001: Set to 1 to indicate 16-bit comparison.

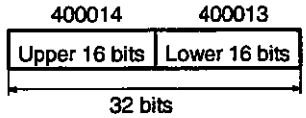
Example 2: Thirty-two-bit Comparison



400001: Stores V1 as follows:



400003: Stores V2 as follows:



#00002: Set to 2 to indicate 32-bit comparison.

Table 2.60 Structural Elements of TEST for 16-bit Comparison

Element	Meaning	Possible Settings	
Top (V1)	1) The contents of the register is used as V1. 2) V1 must be within the range that can be expressed with a 16-bit binary number, i.e., the following ranges: **Unsigned Comparison** Between 0 and 65,535 **Signed Comparison** Between –32,768 and 32,767	Input register: Holding register: Constant register: Link register:	300001 to 300512 (Z00001 to Z00512) 400001 to 409999 (W00001 to W09999) 700001 to 704096 (K00001 to K04096) R10001 to R11024 R20001 to R21024
Middle (V2)	1) The contents of the register is used as V2. 2) V2 must be within the same range as V1.		
Bottom (Z)	Comparison size	Constant:	#00001

Table 2.61 Structural Elements of TEST for 32-bit Comparison

Element	Meaning	Possible Settings
Top (V1)	1) The contents of the specified register and the next register is used as V1. 2) V1 must be within the range that can be expressed with a 32-bit binary number, i.e., the following ranges: **Unsigned Addition** Between 0 and 4,294,967,295 **Signed Addition** Between –2,147,483,648 and 2,147,483,647 3) In the example, "400001" was specified for the top element. 400002 Upper 16 bits 400001 Lower 16 bits	Input register: 300001 to 300511 (Z00001 to Z00511) Holding register: 400001 to 409998 (W00001 to W09998) Constant register: 700001 to 704095 (K00001 to K04095) Link register: R10001 to R11023 R20001 to R21023
Middle (V2)	1) The contents of the specified register and the next register is used as V2. 2) V2 must be within the same ranges as V1. 3) In the example, "400003" was specified for the middle element. 400004 Upper 16 bits 400003 Lower 16 bits	
Bottom (Z)	Comparison size	Constant: #00002

3. Operation

1) If the comparison size is "1," the following operation is performed.

   a) If input 1 is ON, TEST performs 16-bit comparison as follows:

      (1) If input 3 is OFF, the values are compared as unsigned integers expressed as 16-bit binary numbers (0 to 65,535).

      (2) If input 3 is ON, the values are compared as signed integers expressed as 16-bit binary numbers (–32,768 to 32,767).

2) If the comparison size is "2," the following operation is performed.

   a) If input 3 is ON, TEST performs 32-bit signed comparison as follows:

      (1) If input 3 is OFF, the values are compared as unsigned integers expressed as 16-bit binary numbers (0 to 4,294,967,295).

(2) If input 3 is ON, the values are compared as signed integers expressed as 32-bit binary numbers (–2,147,483,648 to 2,147,483,647).

3) The outputs are treated as follows depending on the size of the two operands.

- If V1 > V2, output 1 turns ON.
- If V1 = V2, output 2 turns ON.
- If V1 < V2, output 3 turns ON.

4) The operation of TEST is summarized in the following two tables.

Table 2.62 Operation of TEST for 16-bit Comparison

Inputs		Operation	Outputs 1 to 3
1	3		
ON	OFF	V1 and V2 are compared as unsigned integers expressed as 16-bit binary numbers (0 to 65,535).	• If V1>V2, output 1 turns ON. • If V1=V2, output 2 turns ON. • If V1<V2, output 3 turns ON.
	ON	V1 and V2 are compared as signed integers expressed as 16-bit binary numbers (–32,768 to 32,767).	
OFF	Any	Nothing is done.	OFF

Table 2.63 Operation of TEST for 32-bit Comparison

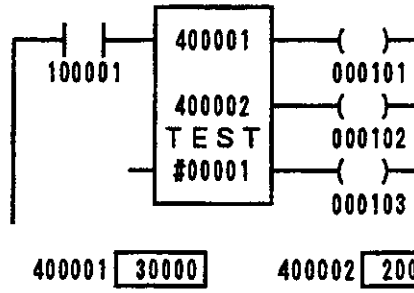
Inputs		Operation	Outputs 1 to 3
1	3		
ON	OFF	V1 and V2 are compared as unsigned integers expressed as 32-bit binary numbers (0 to 4,924,967,295).	• If V1>V2, output 1 turns ON. • If V1=V2, output 2 turns ON. • If V1<V2, output 3 turns ON.
	ON	V1 and V2 are compared as signed integers expressed as 32-bit binary numbers (–2,147,483,648 to 2,147,483,647).	
OFF	Any	Nothing is done.	OFF

◀EXAMPLE▶

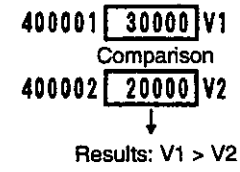
4. Application Examples

Example 1: Unsigned 16-bit Comparison

1) Ladder Programming



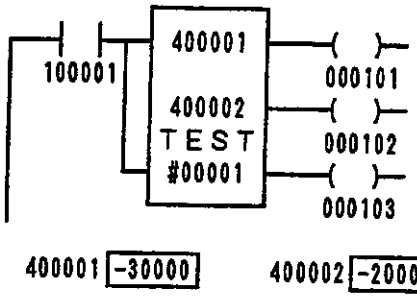
2) Operation



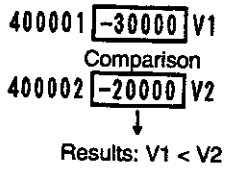
For the above TEST, the comparison shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON.

Example 2: Signed 16-bit Comparison

1) Ladder Programming



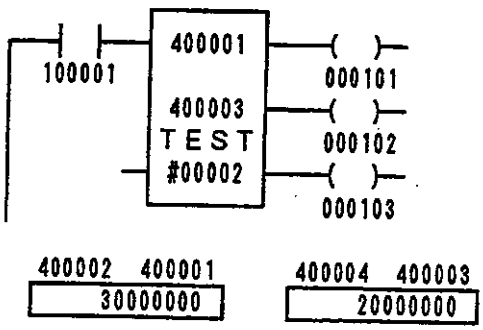
2) Operation



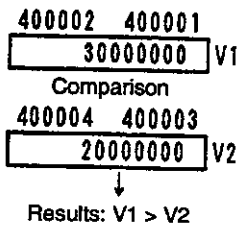
For the above TEST, the comparison shown at the right will be performed when input relay 100001 is ON, and coil 000103 will turn ON.

Example 3: Unsigned 32-bit Comparison

1) Ladder Programming



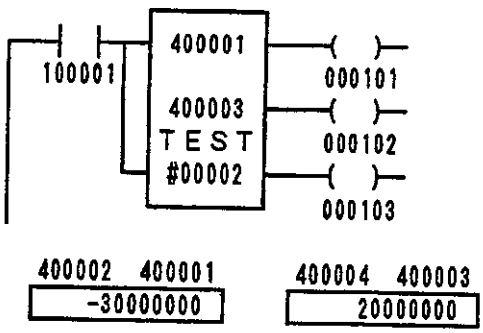
2) Operation



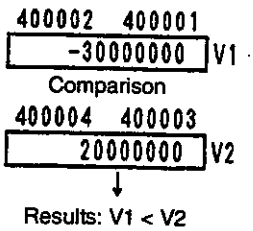
For the above TEST, the comparison shown at the right will be performed when input relay 100001 is ON, and coil 000101 will turn ON.

Example 4: Signed 32-bit Comparison

1) Ladder Programming



2) Operation



For the above TEST, the comparison shown at the right will be performed when input relay 100001 is ON, and coil 000103 will turn ON.

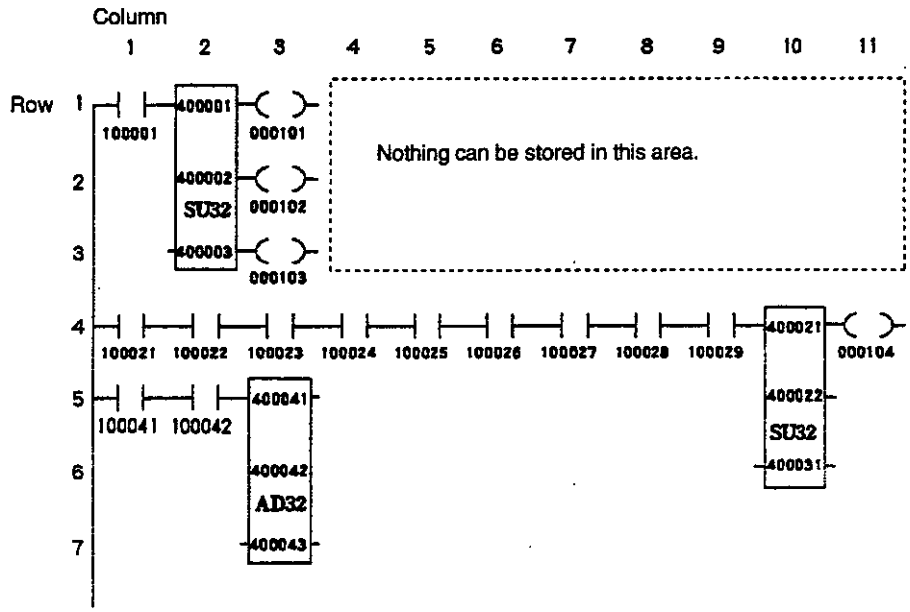
2.10.5 Building Programs

1. Storage Locations on Networks

All 32-bit arithmetic instructions require three vertical elements on a network, one top element, one middle element, and one bottom element. They can thus be stored anywhere on a 5-row by 10-column matrix (rows 1 through 5 and columns 1 through 10) on the network.

Note Sixteen-bit arithmetic instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

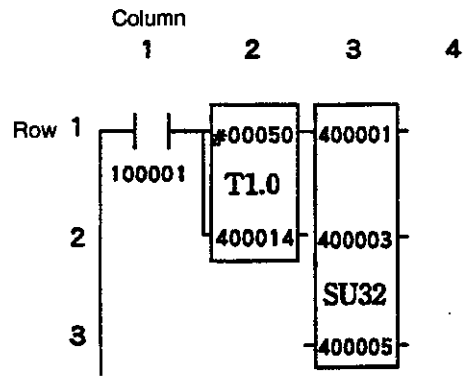
Example



2. Inputs

Inputs to 32-bit math instruction can be connected to relay elements (except coils) and/or outputs from timers, counters, math instructions, data transfer instructions, other instructions, etc.

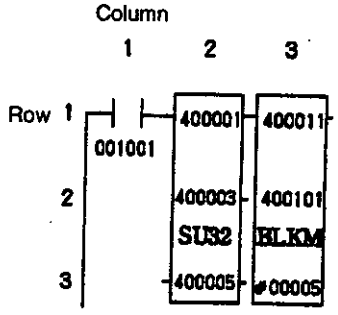
Example



3. Outputs

Outputs from 32-bit math instruction can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data transfer instructions, etc.

Example



Data Transfer Instructions

3

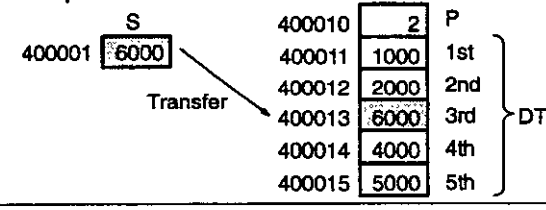
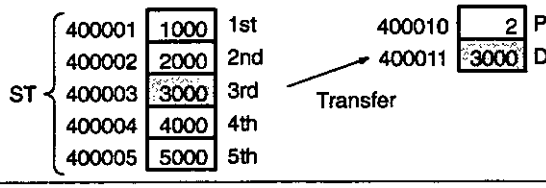
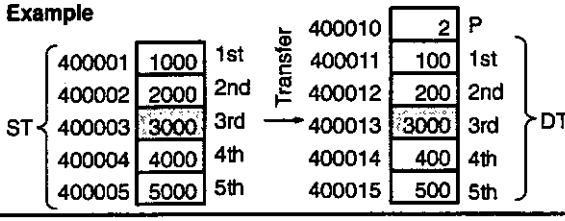
This chapter describes the data transfer instructions.

3.1 Data Transfer Instructions

This section describes the data transfer instructions used to copy data stored in data memory (such as holding registers, input relays, or coils) from one location to another in word units. Difficult and complicated data processing can be performed by combining these data transfer instructions with basic instructions and arithmetic instructions.

- The twelve data transfer instructions are outlined in the following table.

Table 3.1 Data Transfer Instructions

Instruction Name	Symbol	Function	Page
REGISTER-TO-TABLE MOVE	R→T	Copies the content of the source word (S) to the word in the destination table (DT) specified by the pointer value (P). **Example** 	3-11
TABLE-TO-REGISTER MOVE	T→R	Copies the content of the word in the source table (ST) specified by the pointer value (P) to the destination word (D). **Example** 	3-19
TABLE-TO-TABLE MOVE	T→T	Copies the content of the word in the source table (ST) specified by the pointer value (P) to the corresponding word in the destination table (DT). **Example** 	3-27

Abbreviations
S: Source Word
ST: Source Table
D: Destination Word
DT Destination Table
P: Pointer

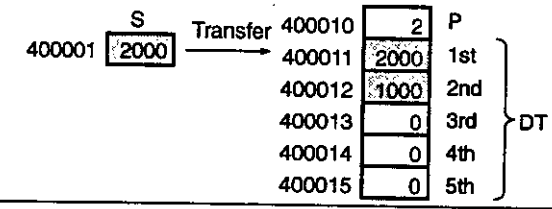
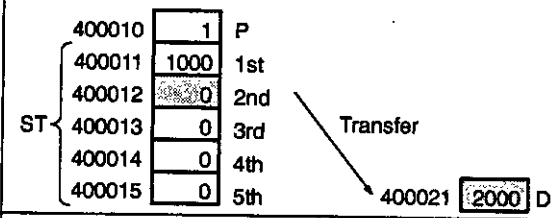
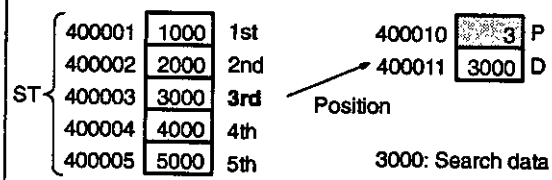
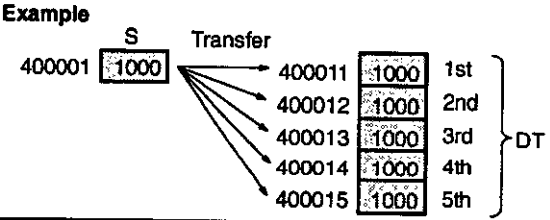
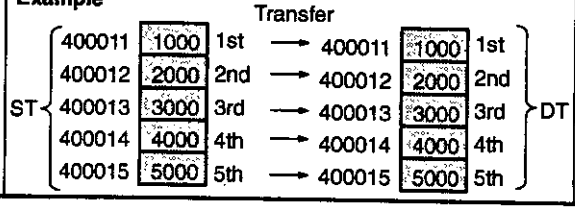
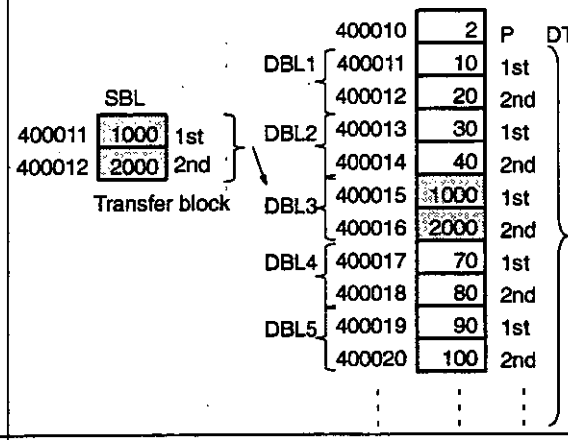
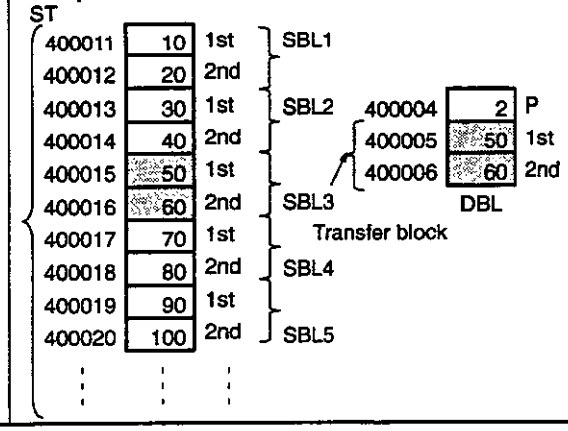
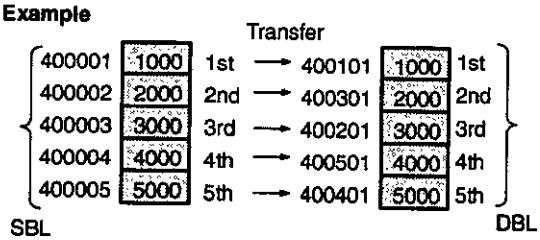
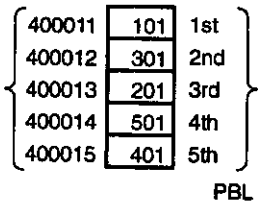
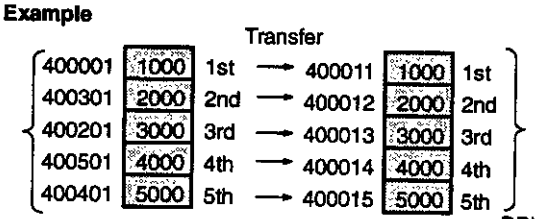
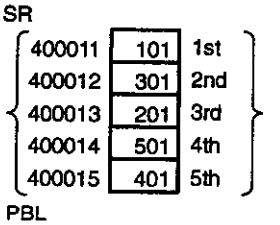
Name	Symbol	Function	Page
FIRST IN	FIN	When there is an "empty register" in the destination table, shifts all of the data down by one word and copies the contents of the source word to the leading word in the destination table. **Example** 	3-34
FIRST OUT	FOUT	Transfers the first word of data inserted into the table (at the end of the source table) to the destination word. **Example** 	3-42
TABLE SEARCH	SRCH	Searches for the search data specified in the destination word and writes that table position in the pointer. **Example** 	3-49
TABLE SET	TSET	Copies the contents of the source word to all of the words in the destination table in a single scan. **Example** 	3-56
BLOCK MOVE	BLKM	Copies the contents of the words in the source table to the corresponding words in the destination table in a single scan. **Example** 	3-58

Table 3.1 Data Transfer Instructions (Cont'd)

Name	Symbol	Function	Page
BLOCK-TO-TABLE MOVE	BLKT	Copies the contents of the source block (SBL) to the destination block (DBL) specified by the pointer value (P). **Example** 	3-65
TABLE-TO-BLOCK MOVE	TBLK	Copies the contents of the source block (SBL) specified by the pointer value (P) to the destination block (DBL). **Example** 	3-72

Abbreviations SBL: Source Block
DBL: Destination Block

Name	Symbol	Function	Page
INDIRECT BLOCK WRITE	IBKW	Copies the contents of each word in the source block (SBL) to the destination holding register (DR) specified by the corresponding word in the pointer block (PBL). **Example**  400000 is added to the content of the words in the pointer block to determine the destination registers for the words in the source block.  For example, 400000 is added to the content of the leading word in the pointer block (101) to determine the leading destination register (400101).	3-79
INDIRECT BLOCK READ	IBKR	Copies the contents of each source holding register (SR) specified by the pointer block (PBL) to the corresponding word in the destination block (DBL). **Example**   400000 is added to the content of the words in the pointer block to determine the source registers. For example, 400000 is added to the content of the leading word in the pointer block (101) to determine the leading source register (400101).	3-86

Abbreviations PBL: Pointer Block
DR: Destination Registers
SR: Source Registers

3.2 Data Transfer Instruction Terminology

This section explains the terms required to understand the operation of the data transfer instructions.

3.2.1 Data Tables

The following tables as known as data tables: register tables, coil tables, and relay tables. These tables are described below.

1. Register Tables

1) A register table is a group of registers with consecutive reference numbers.

2) The following registers can be used to make register tables.

   a) Input registers

   b) Holding registers

   c) Constant registers

   d) Link registers

Example 1
A register table composed of 5 consecutive input registers.

300001	100
300002	200
300003	300
300004	400
300005	500

Example 2
A register table composed of 5 consecutive holding registers.

400001	1000
400002	2000
400003	3000
400004	4000
400005	5000

2. Coil Tables

1) A coil table is a group of coils with consecutive reference numbers.

2) The following coils can be used to make coil tables.

   a) Normal coils (output coils, internal coils)

   b) Link coils

   c) MC coils

   d) MC control coils

3) The number of coils in a coil table must be a multiple of 16.

4) The lower 5 digits of the reference number of the first coil in the coil table must be16n +1 (where n = 0, 1, 2, ...). In the following example, the reference number of the first coil in the coil table (000001) satisfies this condition.

Example
This coil table is composed of 32 consecutive coils.

ON —( )— 000001	ON —( )— 000002	--------	ON —( )— 000016
OFF —( )— 000017	OFF —( )— 000018	--------	OFF —( )— 000032

3. Relay Tables

1) A relay table is a group of relays with consecutive reference numbers.

2) The following relays can be used to make relay tables.

a) Input relays

b) MC relays

c) MC control relays

d) M code relays

3) The number of relays in a relay table must be a multiple of 16.

4) The lower 5 digits of the reference number of the first relay in the relay table must be16n +1 (where n = 0, 1, 2, ...). In the following example, the reference number of the first relay in the relay table (100001) satisfies this condition.

Example
This relay table is composed of 32 consecutive relays.

ON ⊣ ⊢ 100001	ON ⊣ ⊢ 100002	--------	ON ⊣ ⊢ 100016
OFF ⊣ ⊢ 100017	OFF ⊣ ⊢ 100018	--------	OFF ⊣ ⊢ 100032

Note N.O. contacts are used to represent the input relays that make up input relay tables.
Example

⊣ ⊢ 100001 This symbol represents input relay 100001.

3.2.2 Data Table Sizes

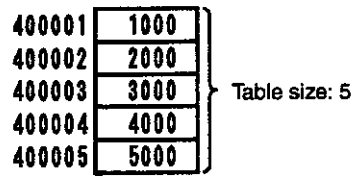
1) The size of a data table is indicated by the "table size." The following rules apply to the table size.

   a) For register tables, the table size is calculated with units of 1 register.

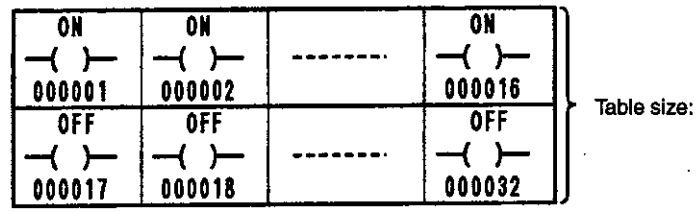
   b) For coil tables, the table size is calculated with units of 16 coils.

   c) For relay tables, the table size is calculated with units of 16 relays.

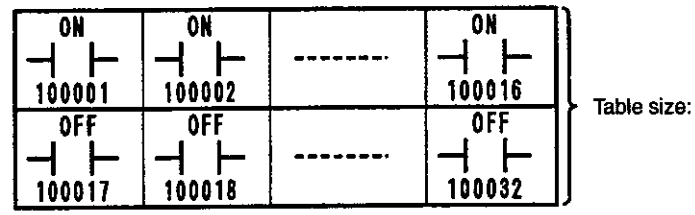
Example 1
The following diagram shows a holding register table with a table size of 5.



Example 2
The following diagram shows a coil table with a table size of 2.



Example 3
The following diagram shows an input relay table with a table size of 2.



2) The maximum table size depends on the data transfer instruction and type of reference being used. Check the maximum table size in the instruction's "component element table" when creating the program.

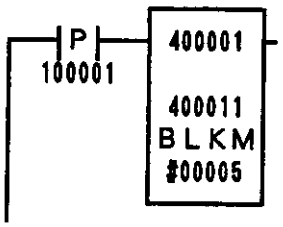
3.2.3 Source and Destination Tables

1) The source location for a data transfer is known as the source word or source table and the destination location for a data transfer is known as the destination word or destination table.

2) When a data transfer instruction is executed, the data is copied from the source to the destination. The source data is left unchanged.

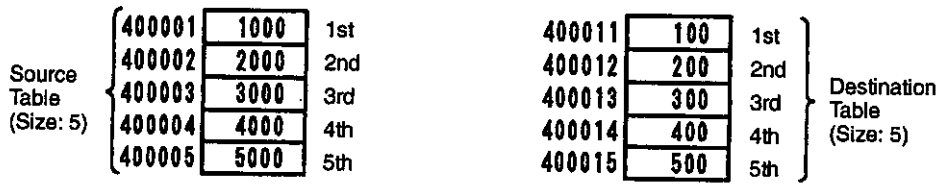
Example
This example shows a data transfer with the BLOCK MOVE (BLKM) instruction.

1) Ladder Programming

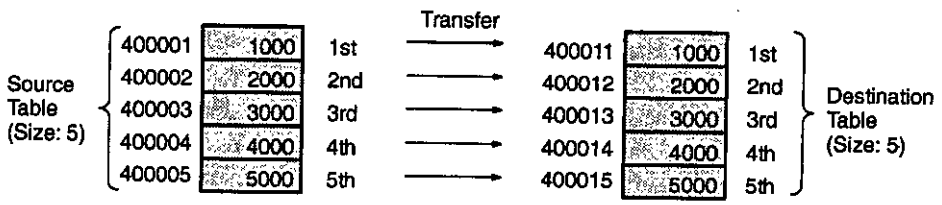


2) Transfer Operation

a) Status Before Execution



b) The following data transfer is executed when input relay 100001 goes from OFF to ON. The transfer is completed in one scan.



(1) The content of the n^{th} word (n=1 to 5) in the source table is copied to the n^{th} word (n=1 to 5) in the destination table.

(2) The contents of the source table aren't changed by executing the instruction.

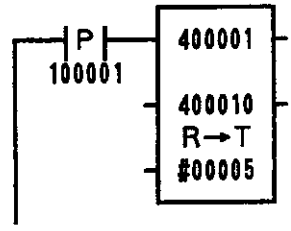
3.2.4 Pointers

1) A pointer is a register that is used to specify a particular location (a register, 16-coil group, or 16-relay group) in a source table or destination table.

2) In data transfer instructions other than the BLOCK MOVE (BLKM) instruction and TABLE SET (TSET) instruction, pointers are used to specify the source and/or destination.

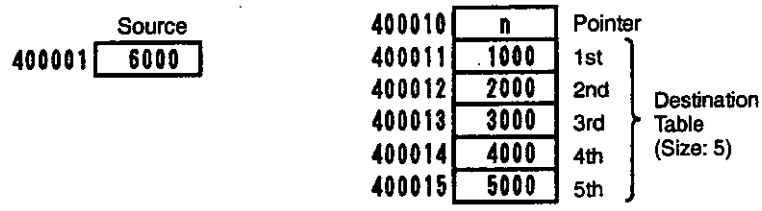
Example
In this example, a data transfer is performed with the REGISTER-TO-TABLE MOVE (R→T) instruction.

1) Ladder Programming



2) Transfer Operation

a) Status Before Execution



b) The following data transfer is executed when the pointer value is 2 and input relay 100001 goes from OFF to ON. The transfer is completed in one scan.

(1) The source data is copied to the third register in the destination table.

(2) The pointer value is 3.

c) The source data can be copied to any word in the table by setting the pointer value from 0 to 4.

3.3 Data Transfer Instruction Details

This section describes the functions, structures, and operation of the data transfer instructions and provides simple examples of their application.

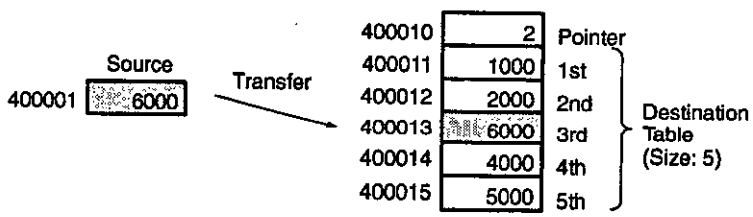
3.3.1 REGISTER-TO-TABLE MOVE (R→T)

1. Function

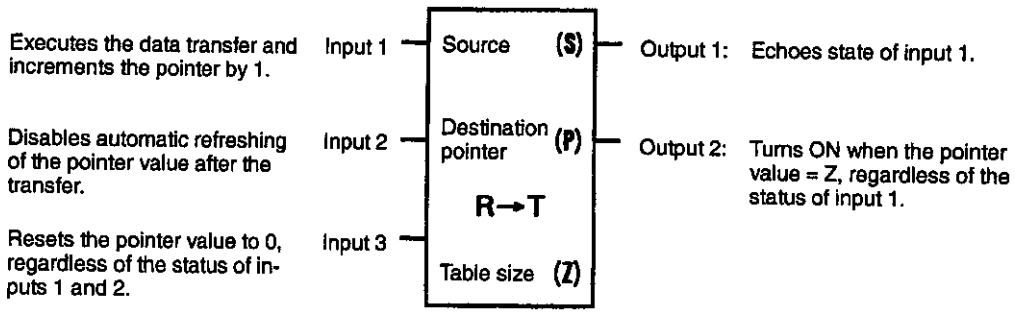
1) Data from a single register is transferred to a data table. The pointer value determines which register in the table is the destination register.

2) The word of data in the source is copied to the register in the destination table specified by the pointer value.

Example
In this example, the instruction is executed with a pointer value of 2, which transfers the source data to the 3rd register in the destination table.



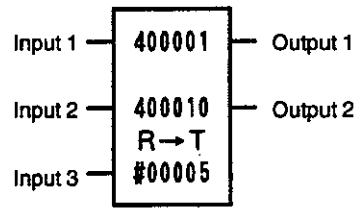
2. Structure



1) R→T is the symbol for REGISTER-TO-TABLE MOVE.

2) R→T requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 3.2* lists the reference numbers and constants that can be specified. The leading register in the destination table is the one just after the pointer.

Example



400001: Reference number of the source
400010: Reference number of the pointer (The next register is the leading register in the destination table.)
#00005: Size of the destination table (5)

Table 3.2 Structural Elements of R→T

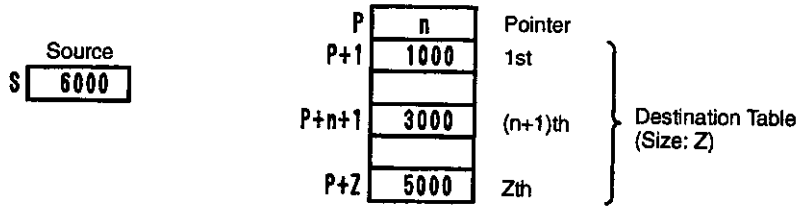
Element	Meaning	Possible Settings	
Top (S)	Reference number of the source	Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (P)	Reference number of the pointer	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 or R20001 to R21023
Bottom (Z)	Size of the destination table	Constant:	#00001 to #00999

Note (1) When a coil or relay is being specified, the last 5 digits of the reference number must be16n +1 (where n = 0, 1, 2, ...).

(2) The destination table starts from the register just after the pointer.

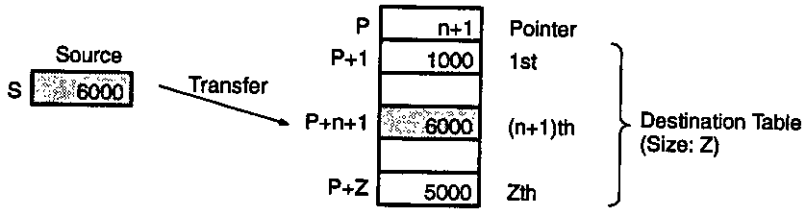
3. Operation

1) Before Execution



2) Pointer Value (n): 0 ≤ n ≤ Z–1

If the pointer value is less than Z, the following data transfer will be executed when input 1 turns ON. The transfer is completed in one scan.



a) The data in the source is copied to the $(n+1)^{th}$ register in the destination table.

b) The pointer value will be incremented by 1 if input 2 is OFF; it is left unchanged if input 2 is ON.

c) The content of the source is left unchanged.

d) The status of the outputs is as follows:

(1) Output 1: Turns ON.

(2) Output 2: Transfer result. Turns ON only when n=Z.

3) Pointer Value (n): n=Z

If the pointer value is equal to Z, the data transfer won't be executed even when input 1 turns ON. Both output 1 and output 2 will be ON.

4) The pointer value must be 0 ≤ n ≤ Z, regardless of the status of inputs 1, 2, and 3. If the pointer value is less than zero, it will be set to zero; if it is greater than Z, it will be set to Z.

5) When input 3 is turned ON, the pointer will be reset to zero, regardless of the status of inputs 1 and 2.

6) The following table shows the operation of the R→T instruction for all possible input combinations. The pointer value is n and the destination table size is Z.

Table 3.3 R→T Operation

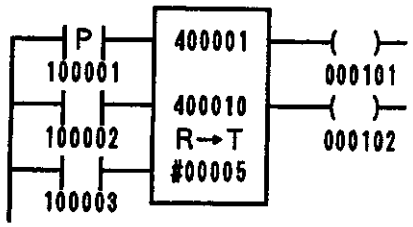
Inputs			Condition of n	R→T Operation	Outputs	
1	2	3			1	2
ON	OFF	OFF	$0 \leq n \leq Z-2$	1) The source data is copied to the $(n+1)^{th}$ register in the destination table. 2) The pointer value (n) is incremented by 1 after the transfer.	**ON**	OFF
			$n = Z-1$			**ON**
			$n = Z$	1) The transfer isn't executed. 2) The pointer value (n) isn't changed.		**ON**
	ON	OFF	$0 \leq n \leq Z-1$	1) The source data is copied to the $(n+1)^{th}$ register in the destination table. 2) The pointer value (n) isn't changed.		OFF
			$n = Z$	1) The transfer isn't executed. 2) The pointer value (n) isn't changed.		**ON**
	OFF	**ON**	None	1) After resetting the pointer value (n) to 0, the source data is copied to the leading register in the destination table. 2) The pointer value (n) is incremented by 1 after the transfer.		OFF
	ON	**ON**	None	1) After resetting the pointer value (n) to 0, the source data is copied to the leading register in the destination table. 2) The pointer value (n) isn't changed. (n=0)		OFF
OFF	Any	**ON**	None	1) The transfer isn't executed. 2) The pointer value (n) is reset to 0.	OFF	OFF
		OFF	$n \neq Z$	1) The transfer isn't executed.		OFF
			$n = Z$	2) The pointer value (n) isn't changed.		**ON**

Note The pointer value must be $0 \leq n \leq Z$, regardless of the status of inputs 1, 2, and 3. If the pointer value is less than zero, it will be set to zero; if it is greater than Z, it will be set to Z.

4. Application Examples

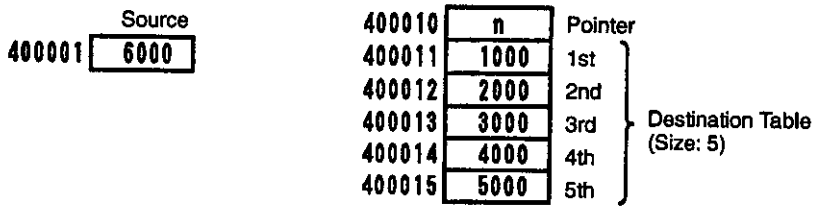
◀EXAMPLE▶ **Example 1**

1) Ladder Programming

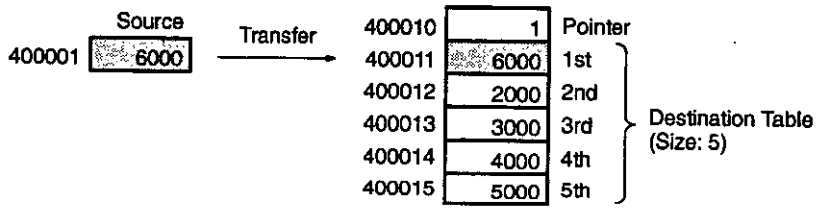


2) Transfer Operation

a) Status Before Execution



b) The following data transfer is executed when the pointer value (n) is 0 and input relay 100001 changes from OFF to ON. The transfer is completed in one scan.



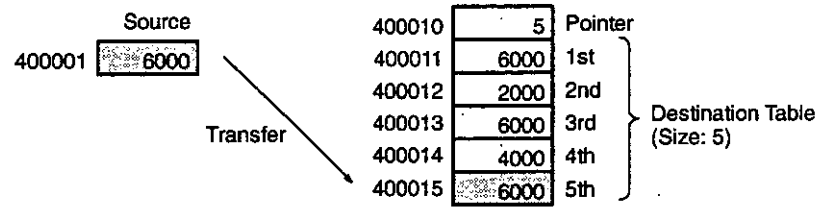
(1) The source data is copied to the leading register in the destination table.

(2) The pointer value will be incremented to 1 (n=1) if input relay 100002 is OFF. The pointer value will be left unchanged (n=0) if input relay 100002 is ON.

(3) The content of the source is left unchanged.

(4) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102:
Remains OFF.

c) The following data transfer is executed when input relay 100001 changes from OFF to ON and the pointer value is 4 (n=4). The transfer is completed in one scan.



(1) The source data is copied to the fifth register in the destination table.

(2) The pointer value will be incremented to 5 (n=5) if input relay 100002 is OFF. The pointer value will be left unchanged (n=4) if input relay 100002 is ON.

(3) The content of the source is left unchanged.

(4) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102:
Turns ON only when n=5.

d) The source data can be copied to any register in the table (1st to 5th) by setting the pointer value from 0 to 4.

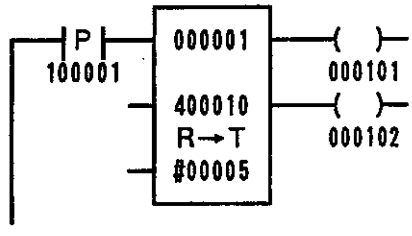
e) If n=5, the data transfer won't be performed when input relay 100001 changes from OFF to ON. In this case, the status of the outputs will be as follows:
Coil 000101: Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102: Turns ON.

f) The pointer value must be $0 \leq n \leq 5$, regardless of the status of input relays 100001, 100002, and 100003. If the pointer value is less than 0, it will be set to 0; if it is greater than 5, it will be set to 5.

g) When input relay 100003 is turned ON, the pointer value (n) will be reset to zero, regardless of the status of input relays 100001 and 100002.

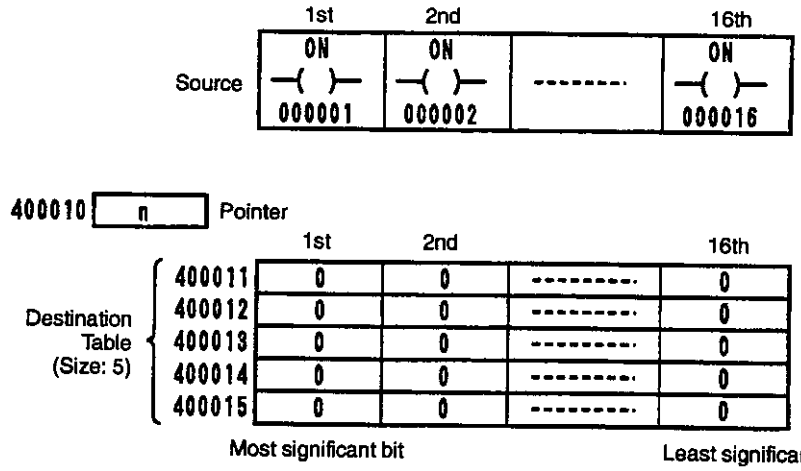
◀EXAMPLE▶ **Example 2**

1) Ladder Programming

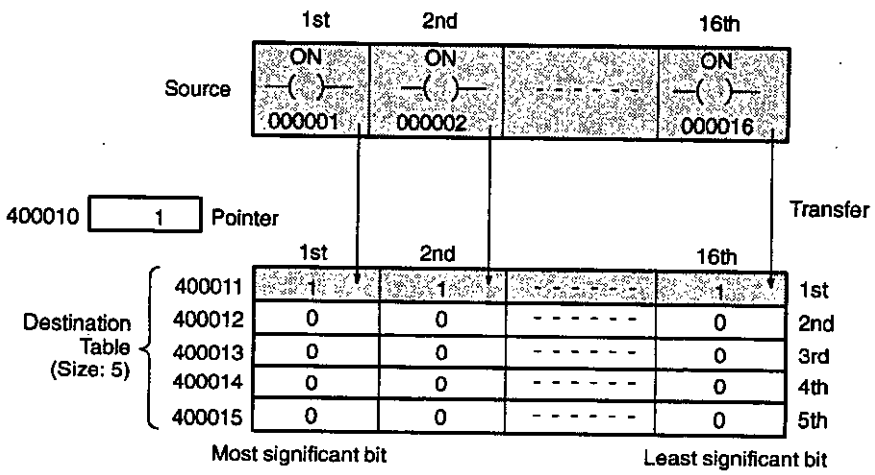


2) Transfer Operation

a) Status Before Execution



b) The following data transfer is executed when the pointer value (n) is 0 and input relay 100001 changes from OFF to ON. The transfer is completed in one scan.

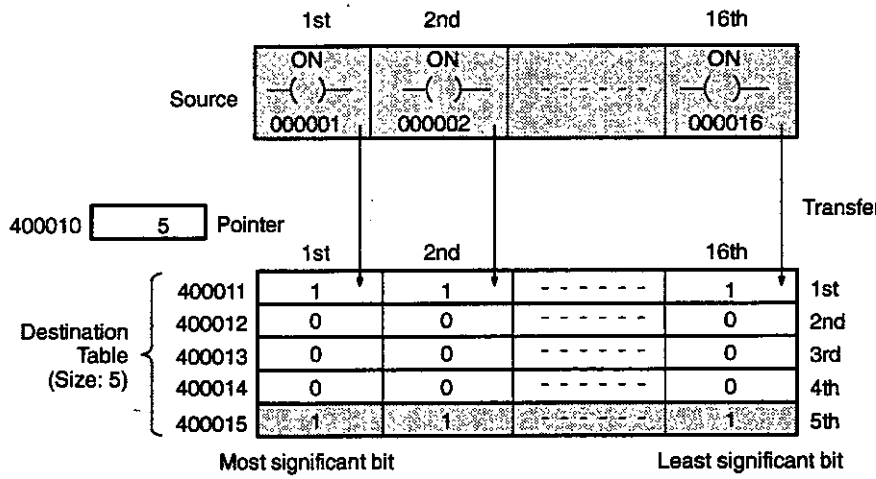


(1) The status of the 16 coils in the source is transmitted to the leading register in the destination table. When a coil is ON, the corresponding bit is set to 1; when a coil is OFF, the corresponding bit is set to 0.

(2) The pointer value is set to 1.

(3) Coil 000101 is turned ON only for the scan in which input relay 100001 changes from OFF to ON. Coil 000102 is left OFF.

c) The following data transfer is executed when input relay 100001 changes from OFF to ON and the pointer value is 4 (n=4). The transfer is completed in one scan.



(1) The status of the 16 coils in the source is transmitted to the fifth register in the destination table. When a coil is ON, the corresponding bit is set to 1; when a coil is OFF, the corresponding bit is set to 0.

(2) The pointer value is incremented to 5.

(3) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102:
Turns ON.

d) The source data can be copied to any register in the table (1^{st} to 5^{th}) by setting the pointer value from 0 to 4.

e) If n=5, the data transfer won't be performed even when input relay 100001 changes from OFF to ON. In this case, the status of the outputs will be as follows:
Coil 000101: Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102: Turns ON.

f) The pointer value must be $0 \leq n \leq 5$, regardless of the status of input relay 100001. If the pointer value is less than 0, it will be set to 0; if it is greater than 5, it will be set to 5.

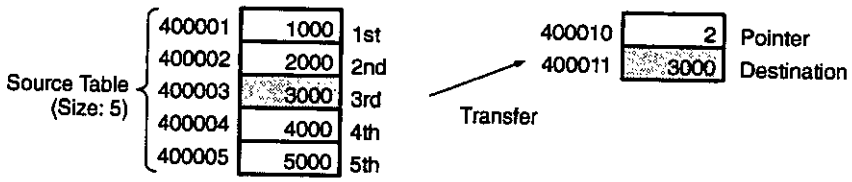
3.3.2 TABLE-TO-REGISTER MOVE (T→R)

1. Function

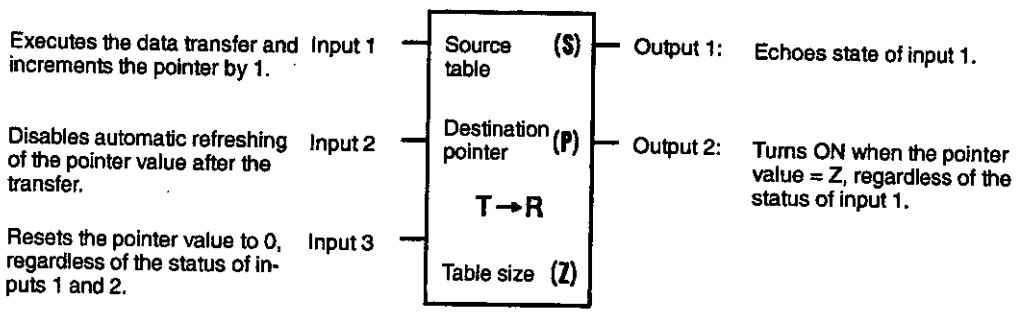
1) Data is transferred from a data table to a single register. The pointer is just before the destination register and the pointer value determines which register in the source table is the source register.

2) Any register in the source table can be specified by changing the pointer value. The data in the specified register is copied to the destination register.

Example
In this example, the instruction is executed with a pointer value of 2, which transfers the source data to the 3rd register in the destination table.



2. Structure



1) T→R is the symbol for TABLE-TO-REGISTER MOVE.

2) T→R requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 3.4* lists the reference numbers and constants that can be specified. The destination register is the one just after the pointer.

Example

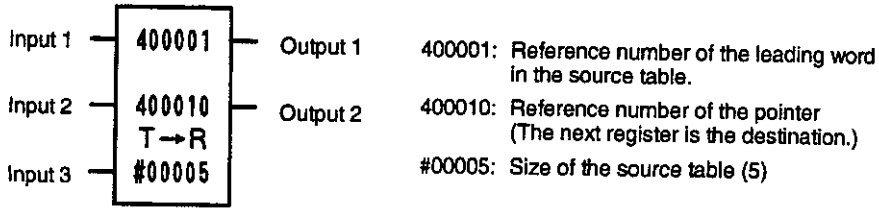


Table 3.4 Structural Elements of T→R

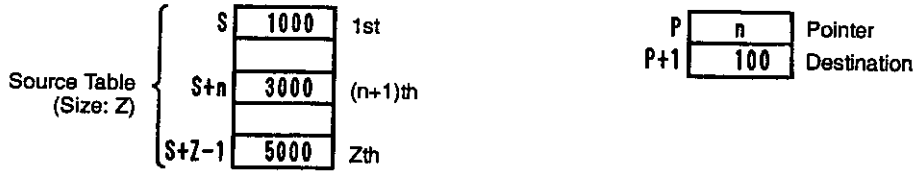
Element	Meaning	Possible Settings	
Top (S)	Reference number of the leading word in the source table	Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (P)	Reference number of the pointer	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 or R20001 to R21023
Bottom (Z)	Size of the source table	Specify the constant. The maximum value of the constant differs with specified reference type.	
		Coil:	#00001 to #00512
		Input relay:	#00001 to #00064
		Input register:	#00001 to #00512
		Holding register or constant register:	#00001 to #00999
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00999
		MC coil, MC relay or MC control relay:	#00001 to #00016
		MC control coil:	#00001 to #00010
		M code relay:	#00001 to #00006

Note (1) When a coil or relay is being specified, the last 5 digits of the reference number must be16n +1 (where n = 0, 1, 2, ...).

(2) The destination register is the one just after the pointer.

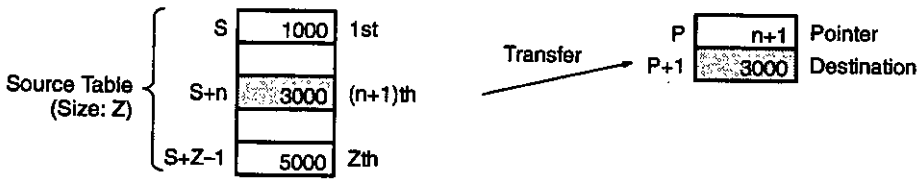
3. Operation

1) Before Execution



2) Pointer Value (n): $0 \leq n \leq Z-1$

If the pointer value is less than Z, the following data transfer will be executed when input 1 turns ON. The transfer is completed in one scan.



a) The data in the $(n+1)^{th}$ register of the source table is copied to the destination register.

b) The pointer value will be incremented by 1 if input 2 is OFF; it is left unchanged if input 2 is ON.

c) The content of the source is left unchanged.

d) The status of the outputs is as follows:

(1) Output 1: Turns ON.

(2) Output 2: Transfer result. Turns ON only when n=Z.

3) Pointer Value (n): n=Z

If the pointer value is equal to Z, the data transfer won't be executed even when input 1 turns ON. Both output 1 and output 2 will be ON.

4) The pointer value must be $0 \leq n \leq Z$, regardless of the status of inputs 1, 2, and 3. If the pointer value is less than zero, it will be set to zero; if it is greater than Z, it will be set to Z.

5) When input 3 is turned ON, the pointer will be reset to zero, regardless of the status of inputs 1 and 2.

6) The following table shows the operation of the T→R instruction for all possible input combinations. The pointer value is n and the source table size is Z.

Table 3.5 T→R Operation

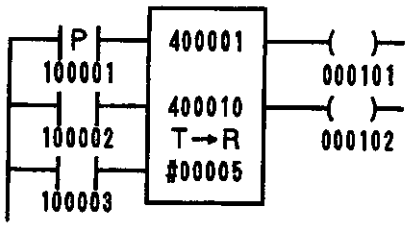
Inputs			Condition of n	T→R Operation	Outputs	
1	2	3			1	2
ON	OFF	OFF	$0 \leq n \leq Z-2$	1) The data in the $(n+1)^{th}$ register of the source table is copied to the destination.	**ON**	OFF
			$n = Z-1$	2) The pointer value (n) is incremented by 1 after the transfer.		**ON**
			$n = Z$	1) The transfer isn't executed. 2) The pointer value (n) isn't changed.		**ON**
	ON	OFF	$0 \leq n \leq Z-1$	1) The data in the $(n+1)^{th}$ register of the source table is copied to the destination. 2) The pointer value (n) isn't changed.		OFF
			$n = Z$	1) The transfer isn't executed. 2) The pointer value (n) isn't changed.		**ON**
	OFF	**ON**	None	1) After resetting the pointer value (n) to 0, the data in the leading register of the source table is copied to the destination. 2) The pointer value (n) is incremented by 1 after the transfer.		OFF
	ON	**ON**	None	1) After resetting the pointer value (n) to 0, the data in the leading register of the source table is copied to the destination. 2) The pointer value (n) isn't changed. (n=0)		OFF
OFF	Any	**ON**	None	1) The transfer isn't executed. 2) The pointer value (n) is reset to 0.	OFF	OFF
		OFF	$n \neq Z$	1) The transfer isn't executed.		OFF
			$n = Z$	2) The pointer value (n) isn't changed.		**ON**

Note The pointer value must be $0 \leq n \leq Z$, regardless of the status of inputs 1, 2, and 3. If the pointer value is less than zero, it will be set to zero; if it is greater than Z, it will be set to Z.

4. Application Examples

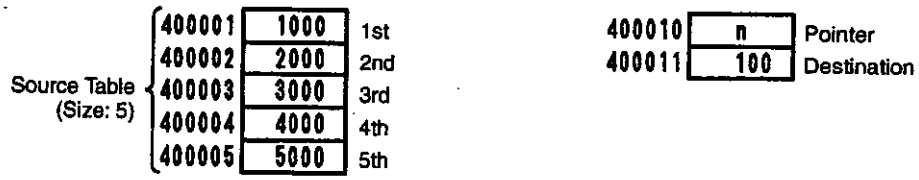
◀EXAMPLE▶ **Example 1**

1) Ladder Programming

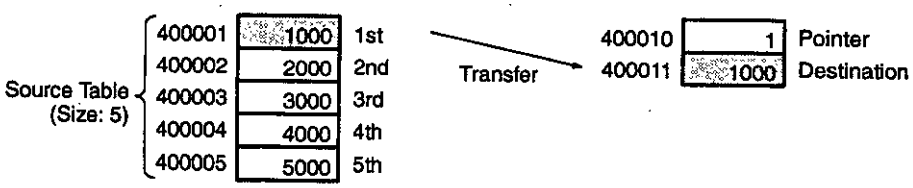


2) Transfer Operation

a) Status Before Execution



b) The following data transfer is executed when the pointer value (n) is 0 and input relay 100001 changes from OFF to ON. The transfer is completed in one scan.



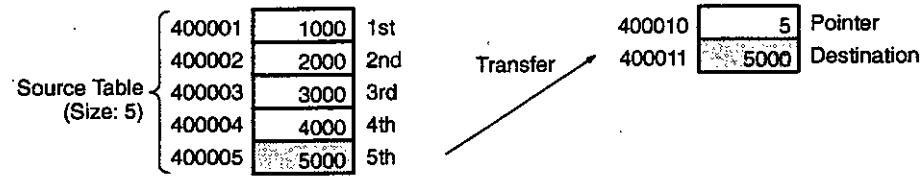
(1) The data from the leading register in the source table is copied to the destination register.

(2) The pointer value will be incremented to 1 (n=1) if input relay 100002 is OFF. The pointer value will be left unchanged (n=0) if input relay 100002 is ON.

(3) The content of the source is left unchanged.

(4) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102:
Remains OFF.

c) The following data transfer is executed when input relay 100001 changes from OFF to ON and the pointer value is 4 (n=4). The transfer is completed in one scan.



(1) The data from the fifth register in the source table is copied to the destination.

(2) The pointer value will be incremented to 5 (n=5) if input relay 100002 is OFF. The pointer value will be left unchanged (n=4) if input relay 100002 is ON.

(3) The content of the source is left unchanged.

(4) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102:
Turns ON only when n=5.

d) The data from any register in the source table (1^{st} to 5^{th}) can be copied to the destination by setting the pointer value from 0 to 4.

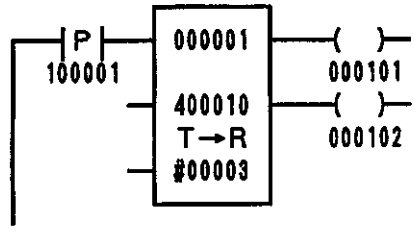
e) If n=5, the data transfer won't be performed even when input relay 100001 changes from OFF to ON. In this case, the status of the outputs will be as follows:
Coil 000101: Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102: Turns ON.

f) The pointer value must be $0 \leq n \leq 5$, regardless of the status of input relays 100001, 100002, and 100003. If the pointer value is less than 0, it will be set to 0; if it is greater than 5, it will be set to 5.

g) When input relay 100003 is turned ON, the pointer value (n) will be reset to zero, regardless of the status of input relays 100001 and 100002.

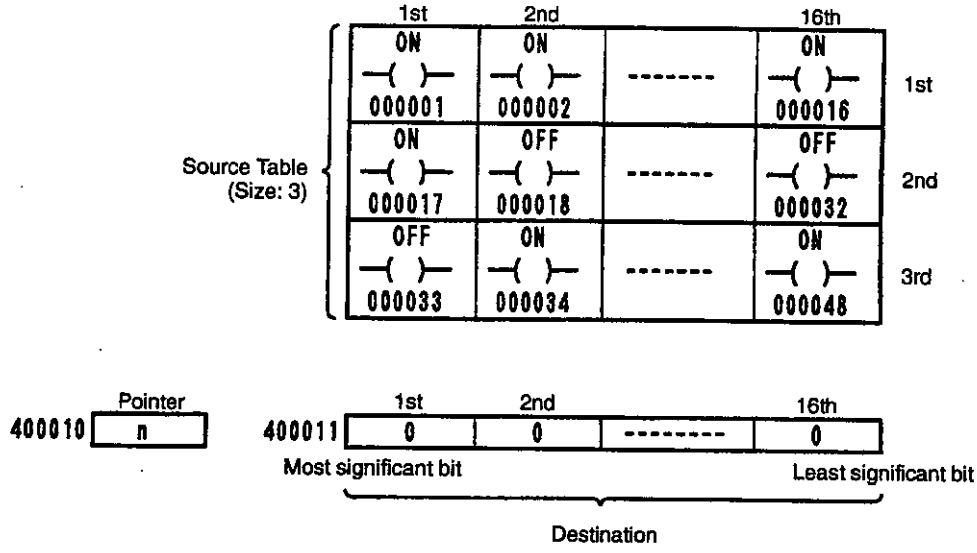
◀EXAMPLE▶ **Example 2**

1) Ladder Programming

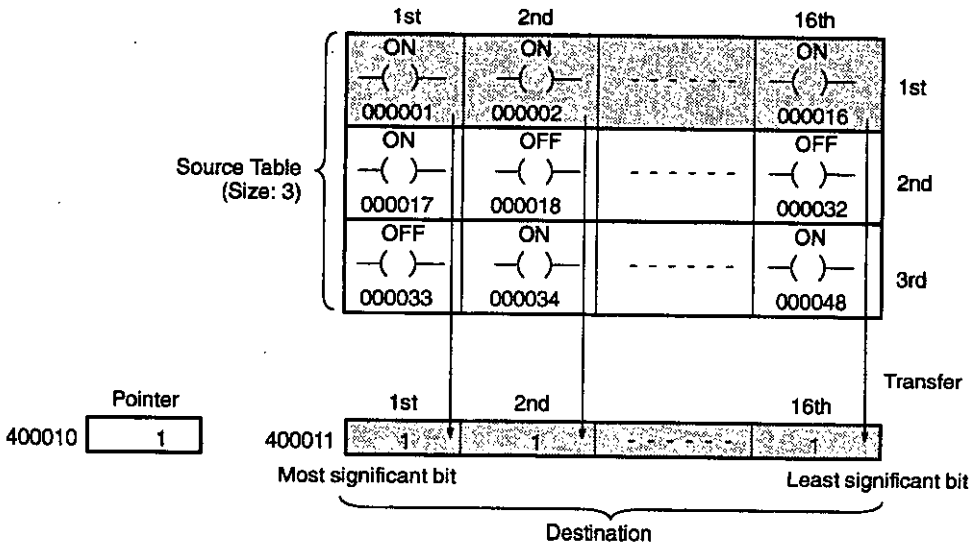


2) Transfer Operation

a) Status Before Execution



b) The following data transfer is executed when the pointer value (n) is 0 and input relay 100001 changes from OFF to ON. The transfer is completed in one scan.

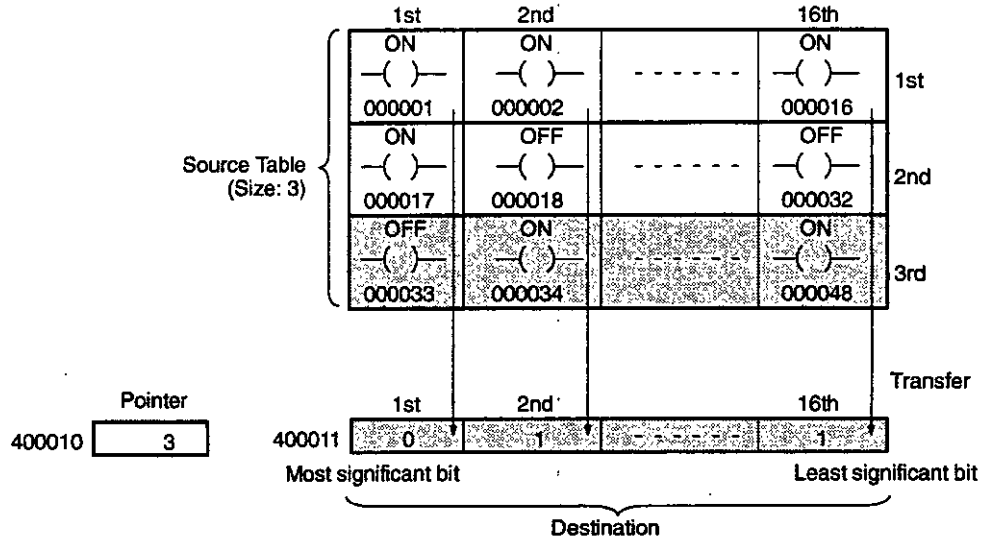


(1) The status of the 16 coils in the leading register of the source table is transmitted to the destination. When a coil is ON, the corresponding bit is set to 1; when a coil is OFF, the corresponding bit is set to 0.

(2) The pointer value is set to 1.

(3) Coil 000101 is turned ON only for the scan in which input relay 100001 changes from OFF to ON. Coil 000102 is left OFF.

c) The following data transfer is executed when input relay 100001 changes from OFF to ON and the pointer value is 2 (n=2). The transfer is completed in one scan.



(1) The status of the 16 coils in the third register of the source table is transmitted to the destination. When a coil is ON, the corresponding bit is set to 1; when a coil is OFF, the corresponding bit is set to 0.

(2) The pointer value is incremented to 3.

(3) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102:
Turns ON.

d) The data from any register in the source table (1^{st} to 3^{rd}) can be copied to the destination by setting the pointer value from 0 to 2.

e) If n=3, the data transfer won't be performed even when input relay 100001 changes from OFF to ON. In this case, the status of the outputs will be as follows:
Coil 000101: Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102: Turns ON.

f) The pointer value must be $0 \leq n \leq 3$, regardless of the status of input relay 100001. If the pointer value is less than 0, it will be set to 0; if it is greater than 3, it will be set to 3.

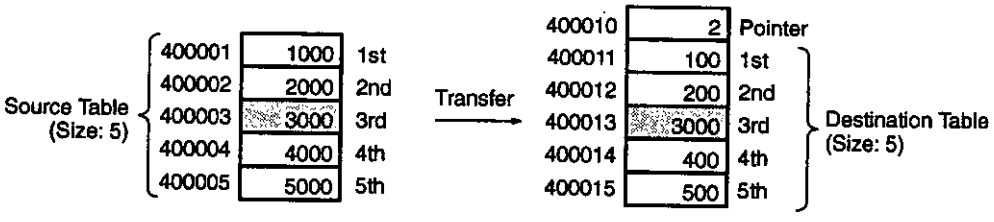
3.3.3 TABLE-TO-TABLE MOVE (T→T)

1. Function

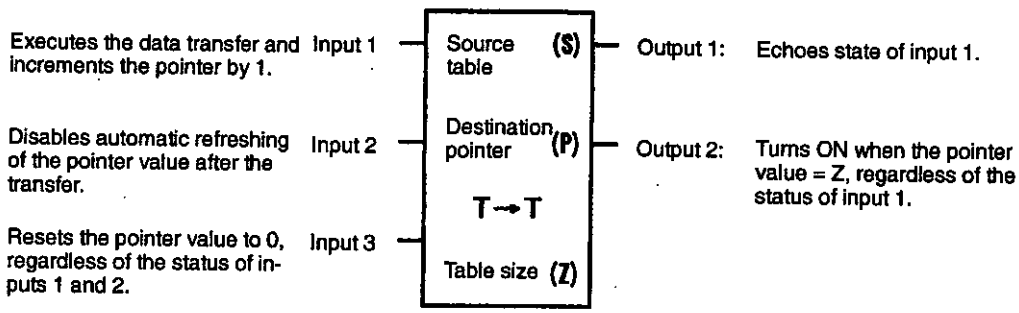
1) Data is transferred from a data table to another data table of the same size. The pointer is just before the destination table and the pointer value determines which register in the table is copied.

2) The data from the specified register in the source table is copied to the corresponding register in the destination table.

Example
In this example, the instruction is executed with a pointer value of 2, which copies the data in the 3rd register of the source table to the 3rd register of the destination table.



2. Structure



1) T→T is the symbol for TABLE-TO-TABLE MOVE.

2) T→T requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 3.6* lists the reference numbers and constants that can be specified. The leading register in the destination table is the one just after the pointer.

Example

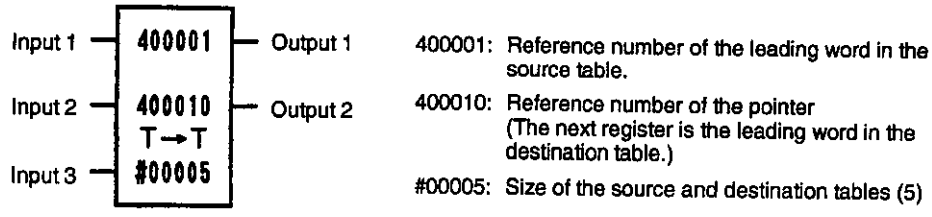


Table 3.6 Structural Elements of T→T

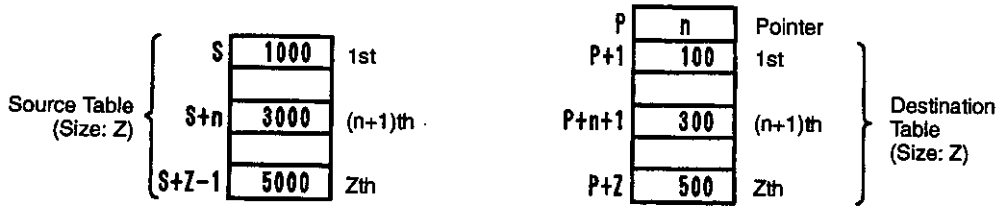
Element	Meaning	Possible Settings	
Top (S)	Reference number of the leading word in the source table	Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (P)	Reference number of the pointer	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 or R20001 to R21023
Bottom (Z)	Size of the source and destination tables	Specify the constant. The maximum value of the constant differs with specified reference type.	
		Coil:	#00001 to #00512
		Input relay:	#00001 to #00064
		Input register:	#00001 to #00512
		Holding register or constant register:	#00001 to #00999
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00999
		MC coil, MC relay or MC control relay:	#00001 to #00016
		MC control coil:	#00001 to #00010
		M code relay:	#00001 to #00006

Note (1) When a coil or relay is being specified, the last 5 digits of the reference number must be16n +1 (where n = 0, 1, 2, ...).

(2) The destination table starts from the register just after the pointer.

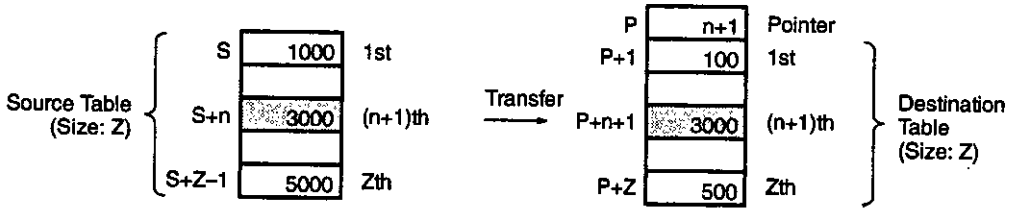
3. Operation

1) Before Execution



2) Pointer Value (n): 0 ≤ n ≤ Z–1

If the pointer value is less than Z, the following data transfer will be executed when input 1 turns ON. The transfer is completed in one scan.



a) The data in the $(n+1)^{th}$ register of the source table is copied to the $(n+1)^{th}$ register of the destination table.

b) The pointer value will be incremented by 1 if input 2 is OFF; it is left unchanged if input 2 is ON.

c) The content of the source is left unchanged.

d) The status of the outputs is as follows:

(1) Output 1: Turns ON.

(2) Output 2: Transfer result. Turns ON only when n=Z.

3) Pointer Value (n): n=Z

If the pointer value is equal to Z, the data transfer won't be executed even when input 1 turns ON. Both output 1 and output 2 will be ON.

4) The pointer value must be 0 ≤ n ≤ Z, regardless of the status of inputs 1, 2, and 3. If the pointer value is less than zero, it will be set to zero; if it is greater than Z, it will be set to Z.

5) When input 3 is turned ON, the pointer will be reset to zero, regardless of the status of inputs 1 and 2.

6) The following table shows the operation of the T→T instruction for all possible input combinations. The pointer value is n and the table size is Z.

Table 3.7 T→T Operation

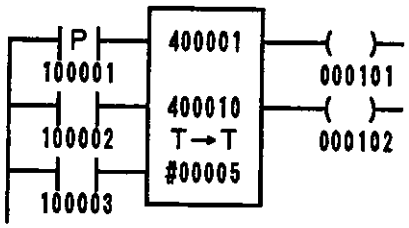
Inputs			Condition of n	T→T Operation	Outputs	
1	2	3			1	2
ON	OFF	OFF	$0 \leq n \leq Z-2$	1) The data in the $(n+1)^{th}$ register of the source table is copied to the $(n+1)^{th}$ register of the destination table. 2) The pointer value (n) is incremented by 1 after the transfer.	**ON**	OFF
			n = Z–1			**ON**
			n = Z	1) The transfer isn't executed. 2) The pointer value (n) isn't changed.		**ON**
	ON	OFF	$0 \leq n \leq Z-1$	1) The data in the $(n+1)^{th}$ register of the source table is copied to the $(n+1)^{th}$ register of the destination table. 2) The pointer value (n) isn't changed.		OFF
			n = Z	1) The transfer isn't executed. 2) The pointer value (n) isn't changed.		**ON**
	OFF	**ON**	None	1) After resetting the pointer value (n) to 0, the data in the leading register of the source table is copied to the leading register of the destination table. 2) The pointer value (n) is incremented by 1 after the transfer.		OFF
	ON	**ON**	None	1) After resetting the pointer value (n) to 0, the data in the leading register of the source table is copied to the leading register of the destination table. 2) The pointer value (n) isn't changed. (n=0)		OFF
OFF	Any	**ON**	None	1) The transfer isn't executed. 2) The pointer value (n) is reset to 0.	OFF	OFF
		OFF	$n \neq Z$	1) The transfer isn't executed.		OFF
			n = Z	2) The pointer value (n) isn't changed.		**ON**

Note The pointer value must be $0 \leq n \leq Z$, regardless of the status of inputs 1, 2, and 3. If the pointer value is less than zero, it will be set to zero; if it is greater than Z, it will be set to Z.

4. Application Examples

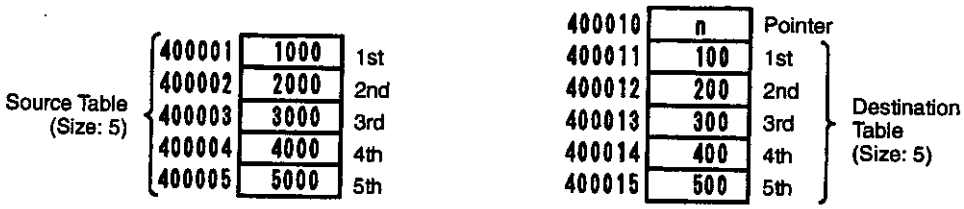
◀EXAMPLE▶ **Example 1**

1) Ladder Programming

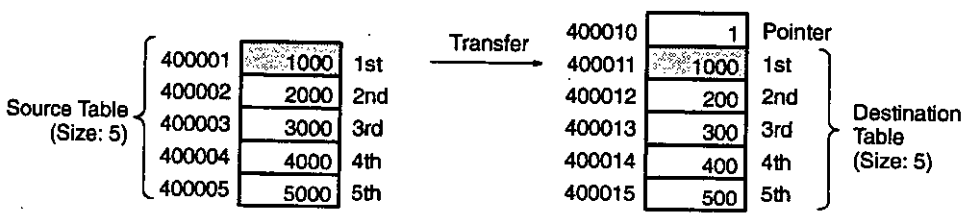


2) Transfer Operation

a) Status Before Execution



b) The following data transfer is executed when the pointer value (n) is 0 and input relay 100001 changes from OFF to ON. The transfer is completed in one scan.



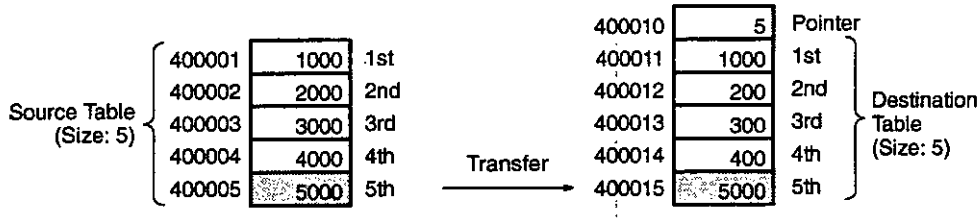
(1) The data from the leading register in the source table is copied to the leading register in the destination table.

(2) The pointer value will be incremented to 1 (n=1) if input relay 100002 is OFF. The pointer value will be left unchanged (n=0) if input relay 100002 is ON.

(3) The content of the source is left unchanged.

(4) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102:
Remains OFF.

c) The following data transfer is executed when input relay 100001 changes from OFF to ON and the pointer value is 4 (n=4). The transfer is completed in one scan.



(1) The data from the fifth register in the source table is copied to the fifth register in the destination table.

(2) The pointer value will be incremented to 5 (n=5) if input relay 100002 is OFF. The pointer value will be left unchanged (n=4) if input relay 100002 is ON.

(3) The content of the source is left unchanged.

(4) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102:
Turns ON only when n=5.

d) The data from any register in the source table (1^{st} to 5^{th}) can be copied to the corresponding register in the destination table by setting the pointer value from 0 to 4.

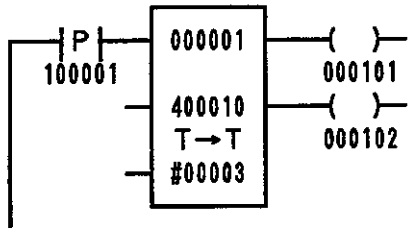
e) If n=5, the data transfer won't be performed even when input relay 100001 changes from OFF to ON. In this case, the status of the outputs will be as follows:
Coil 000101: Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102: Turns ON.

f) The pointer value must be $0 \leq n \leq 5$, regardless of the status of input relays 100001, 100002, and 100003. If the pointer value is less than 0, it will be set to 0; if it is greater than 5, it will be set to 5.

g) When input relay 100003 is turned ON, the pointer value (n) will be reset to zero, regardless of the status of input relays 100001 and 100002.

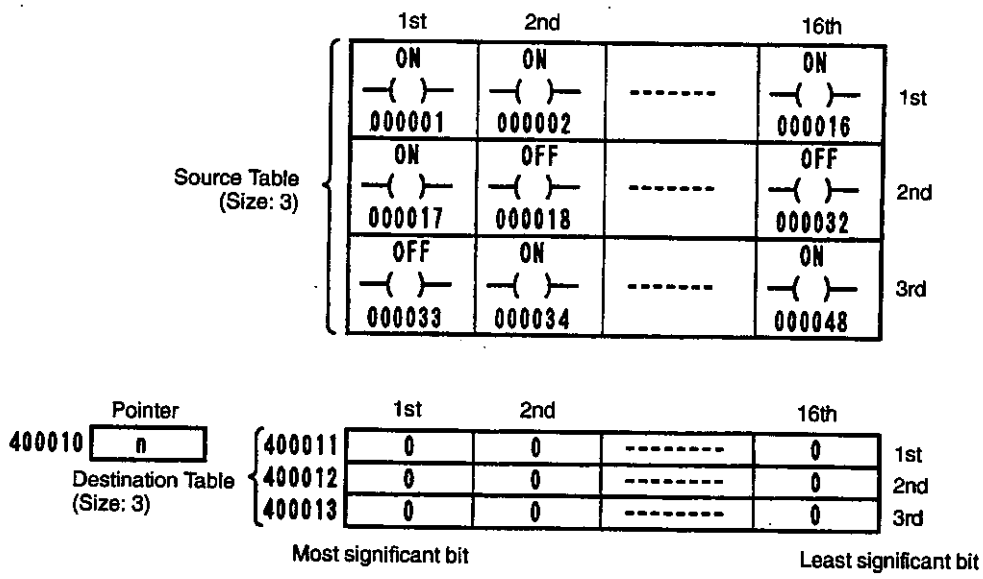
◀EXAMPLE▶ **Example 2**

1) Ladder Programming

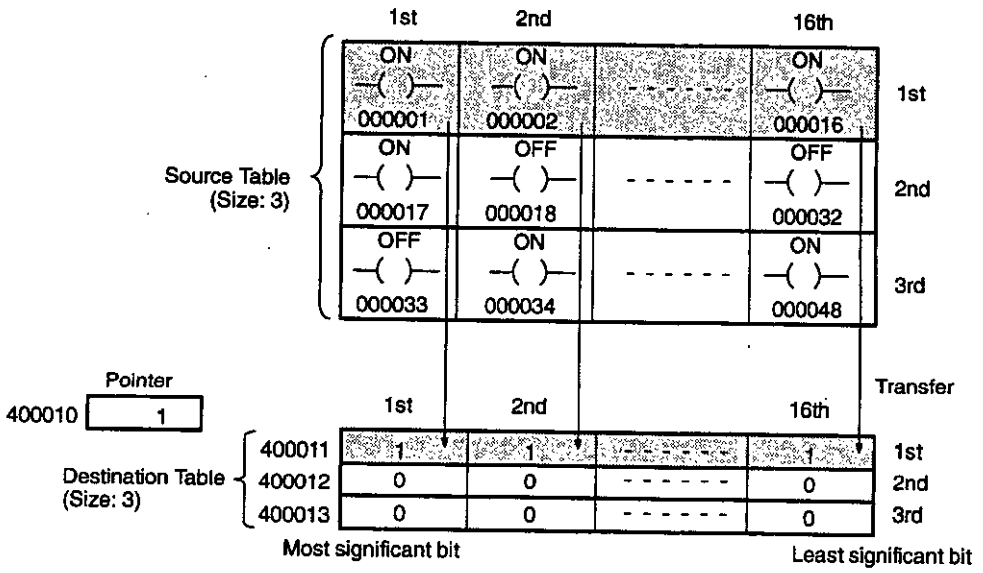


2) Transfer Operation

a) Status Before Execution



b) The following data transfer is executed when the pointer value (n) is 0 and input relay 100001 changes from OFF to ON. The transfer is completed in one scan.

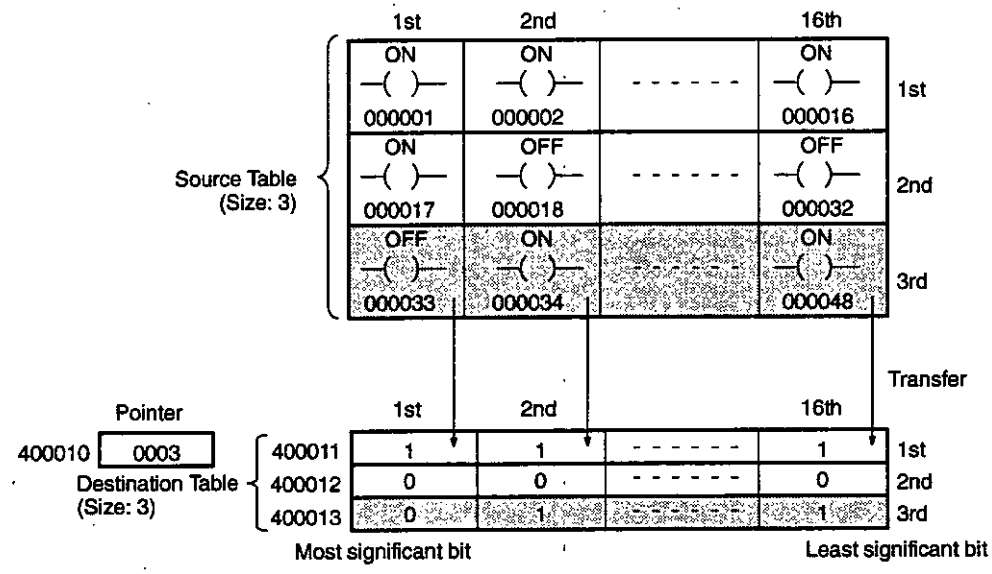


(1) The status of the 16 coils in the leading register of the source table is transmitted to the leading register of the destination table. When a coil is ON, the corresponding bit is set to 1; when a coil is OFF, the corresponding bit is set to 0.

(2) The pointer value is set to 1.

(3) Coil 000101 is turned ON only for the scan in which input relay 100001 changes from OFF to ON. Coil 000102 is left OFF.

c) The following data transfer is executed when input relay 100001 changes from OFF to ON and the pointer value is 2 (n=2). The transfer is completed in one scan.



(1) The status of the 16 coils in the third register of the source table is transmitted to the third register of the destination table. When a coil is ON, the corresponding bit is set to 1; when a coil is OFF, the corresponding bit is set to 0.

(2) The pointer value is incremented to 3.

(3) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102:
Turns ON.

d) The data from any register in the source table (1^{st} to 3^{rd}) can be copied to the corresponding register in the destination table by setting the pointer value from 0 to 2.

e) If n=3, the data transfer won't be performed even if input relay 100001 changes from OFF to ON. In this case, the status of the outputs will be as follows:
Coil 000101: Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102: Turns ON.

f) The pointer value must be $0 \leq n \leq 3$, regardless of the status of input relay 100001. If the pointer value is less than 0, it will be set to 0; if it is greater than 3, it will be set to 3.

3.3.4 FIRST IN (FIN)

1. Function

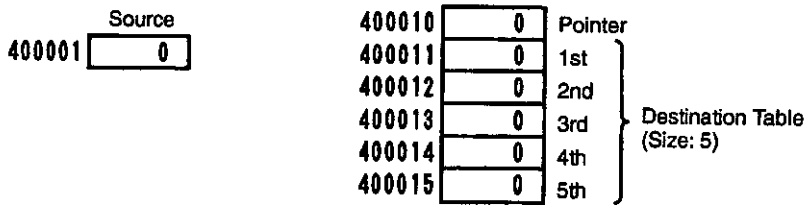
1) Data is transferred from a single register to a data table. The pointer value indicates how many words of data have been stored in the table by the FIRST IN instruction.

2) When there is an "empty register" in the destination table, all of the data is shifted down by one word to empty the leading register in the table and the contents of the source word are copied to that empty register.

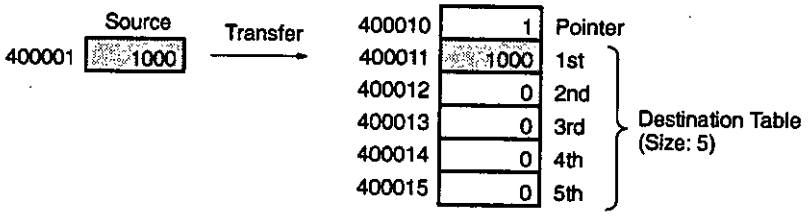
3) The registers farther down in the table contain older data (data that was transferred earlier).

4) The FIRST IN instruction is usually used in conjunction with the FIRST OUT instruction. The FIRST OUT instruction transfers the oldest data from the table to a specified destination register. Refer to *3.3.5 FIRST OUT (FOUT)* for details.

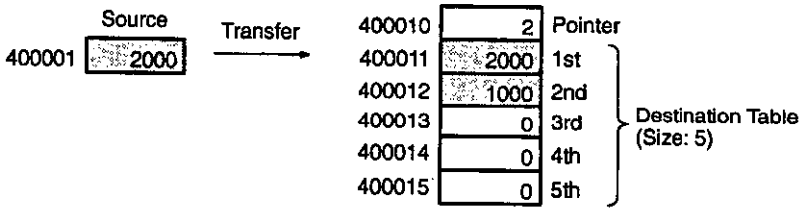
Example
This example shows the operation of the FIRST IN instruction and the changes that occur in the source, pointer, and destination.



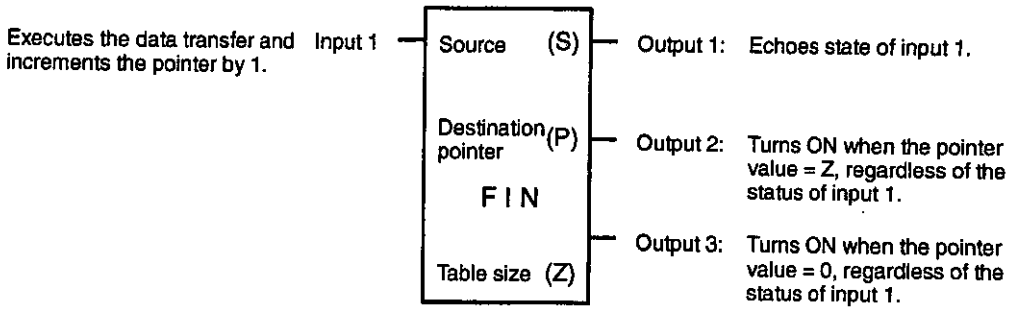
a) First, the instruction is executed with the source data set to 1000; the source data is copied to the leading register in the destination table and the pointer value is incremented to 1.



b) When the instruction is executed again with the source data set to 2000, the older data is shifted down one register, the source data is copied to the leading register in the destination table, and the pointer value is incremented to 2.



2. Structure



1) FIN is the symbol for FIRST IN.

2) FIN requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 3.8* lists the reference numbers and constants that can be specified. The leading register in the destination table is the one just after the pointer.

Example

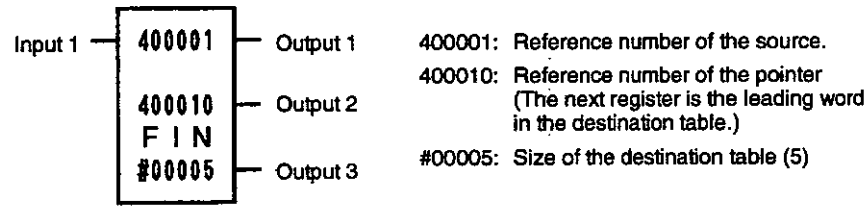


Table 3.8 Structural Elements of FIN

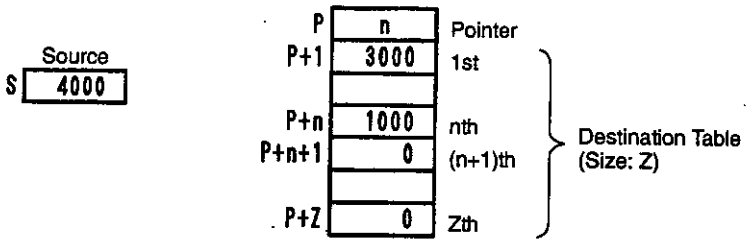
Element	Meaning	Possible Settings	
Top (S)	Reference number of the source	Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (P)	Reference number of the pointer	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 or R20001 to R21023
Bottom (Z)	Size of the destination table	Constant:	#00001 to #00100

Note (1) When a coil or relay is being specified, the last 5 digits of the reference number must be16n +1 (where n = 0, 1, 2, ...).

(2) The destination table starts from the register just after the pointer.

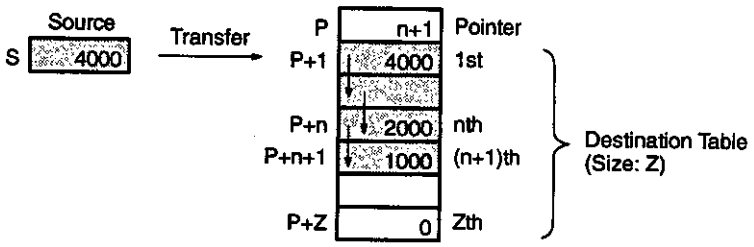
3. Operation

1) Before Execution



2) Pointer Value (n): 0 ≤ n ≤ Z–1

If the pointer value is less than Z, the following data transfer will be executed when input 1 turns ON. The transfer is completed in one scan.



a) All of the data in the destination table is shifted down one word, emptying the leading register in the table.

b) The source data is copied to the leading register of the destination table.

c) The pointer value is incremented by 1.

d) The content of the source is left unchanged.

e) The status of the outputs is as follows:

(1) Output 1: ON

(2) Output 2: Transfer result. ON only when n=Z.

(3) Output 3: OFF

3) Pointer Value (n): n=Z

If the pointer value is equal to Z, the data transfer won't be executed even when input 1 turns ON. The status of the outputs is as follows:

Output 1: ON
Output 2: ON
Output 3: OFF

4) The pointer value must be $0 \leq n \leq Z$, regardless of the status of inputs 1, 2, and 3. If the pointer value is less than zero, it will be set to zero; if it is greater than Z, it will be set to Z.

5) The pointer value indicates how many words of data have been stored in the table by the FIRST IN instruction.

6) The following table shows the operation of the FIN instruction for all combinations of input 1 and the pointer value. The pointer value is "n" and the destination table size is "Z."

Table 3.9 FIN Operation

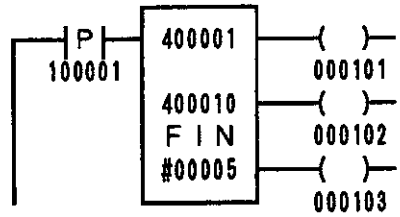
Input 1	Condition of n	FIN Operation	Outputs		
			1	2	3
ON	$0 \leq n \leq Z-2$	1) The data in the destination table is shifted down one word, emptying the leading register in the table. 2) The source data is copied to the leading register of the destination table. 3) The pointer value (n) is incremented by 1 after the transfer.	**ON**	OFF	OFF
	n = Z–1			**ON**	OFF
	n = Z	1) The transfer isn't executed. 2) The pointer value (n) isn't changed.		**ON**	OFF
OFF	n = 0	1) The transfer isn't executed. 2) The pointer value (n) isn't changed.	OFF	OFF	**ON**
	$1 \leq n \leq Z-1$			OFF	OFF
	n = Z			**ON**	OFF

Note The pointer value must be $0 \leq n \leq Z$, regardless of the status of inputs 1, 2, and 3. If the pointer value is less than zero, it will be set to zero; if it is greater than Z, it will be set to Z.

◀EXAMPLE▶

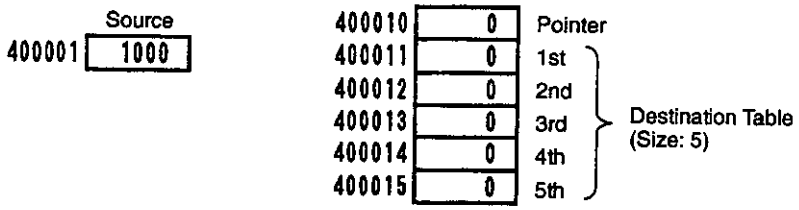
4. Application Example

1) Ladder Programming

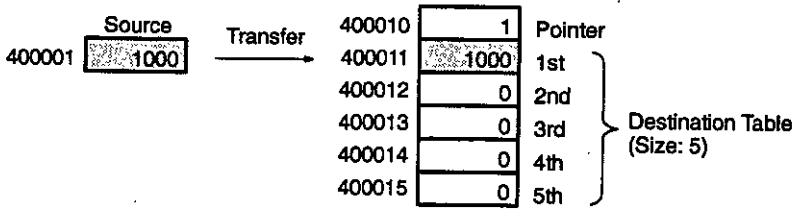


2) Transfer Operation

a) Status Before Execution



b) The following data transfer is executed when input relay 100001 changes from OFF to ON. The transfer is completed in one scan.

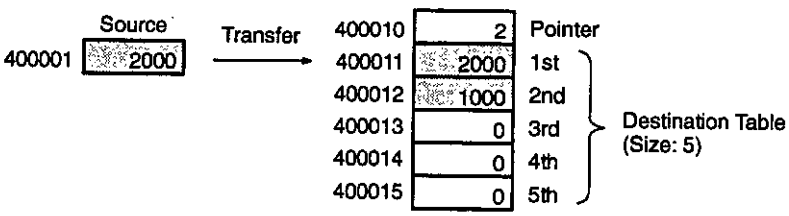


(1) The source data is copied to the leading register of the destination table.

(2) The pointer value is incremented to 1.

(3) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102:
Remains OFF.
Coil 000103:
Remains OFF.

c) The following data transfer is executed when the source data is changed to 2000 and input relay 100001 is turned from OFF to ON. The transfer is completed in one scan.



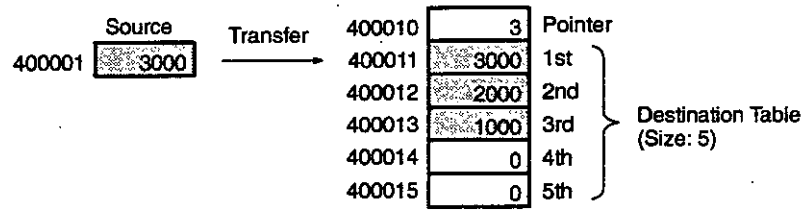
(1) The data in the leading register of the destination table is shifted to the second register, so the leading register is empty.

(2) The source data is copied to the leading register of the destination table.

(3) The pointer value is incremented to 2.

(4) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102:
Remains OFF.
Coil 000103:
Remains OFF.

d) The following data transfer is executed when the source data is changed to 3000 and input relay 100001 is turned from OFF to ON. The transfer is completed in one scan.



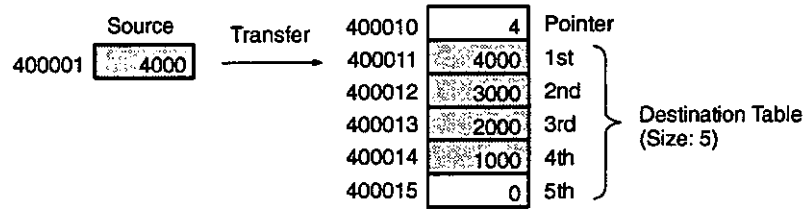
(1) All of the data in the destination table is shifted down one word, so the leading register is empty.

(2) The source data is copied to the destination table's leading register.

(3) The pointer value is incremented to 3.

(4) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102:
Remains OFF.
Coil 000103:
Remains OFF.

e) The following data transfer is executed when the source data is changed to 4000 and input relay 100001 is turned from OFF to ON. The transfer is completed in one scan.



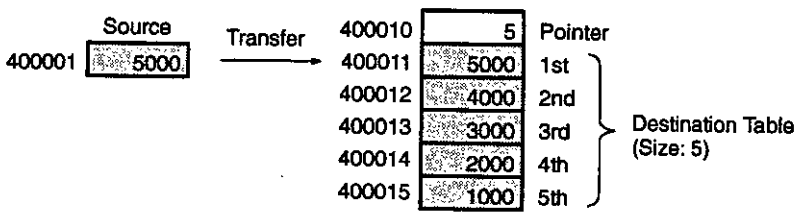
(1) All of the data in the destination table is shifted down one word, so the leading register is empty.

(2) The source data is copied to the leading regiser of the destination table.

(3) The pointer value is incremented to 4.

(4) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102:
Remains OFF.
Coil 000103:
Remains OFF.

f) The following data transfer is executed when the source data is changed to 5000 and input relay 100001 is turned from OFF to ON. The transfer is completed in one scan.



(1) All of the data in the destination table is shifted down one word, so the leading register is empty.

(2) The source data is copied to the leading register of the destination table.

(3) The pointer value is incremented to 5.

(4) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan where input 100001 changes from OFF to ON.
Coil 000102:
Turns ON.
Coil 000103:
Remains OFF.

g) Since there aren't any empty registers in the table, the source data won't be transferred to the table if it is changed to 6000 and input relay 100001 is turned from OFF to ON. Also, the pointer value will remain unchanged (n=5) and the outputs will have the following status:

Coil 000101:
Turns ON only in scan where input 100001 changes from OFF to ON.

Coil 000102:
Turns ON.
Coil 000103
Remains OFF.

h) The pointer value must be $0 \leq n \leq 5$, regardless of the status of input relay 100001. If the pointer value is less than 0, it will be set to 0; if it is greater than 5, it will be set to 5.

3.3.5 FIRST OUT (FOUT)

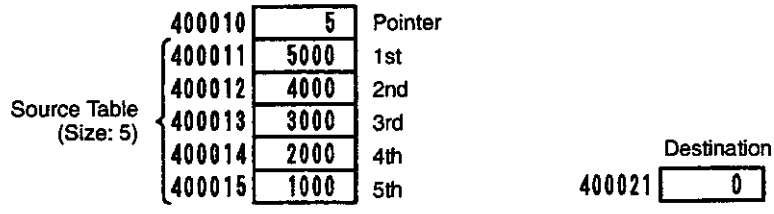
1. Function

1) Data is transferred from a data table to a single register. The pointer is located just before the source table and it indicates how many words of data can be read by the FIRST OUT instruction.

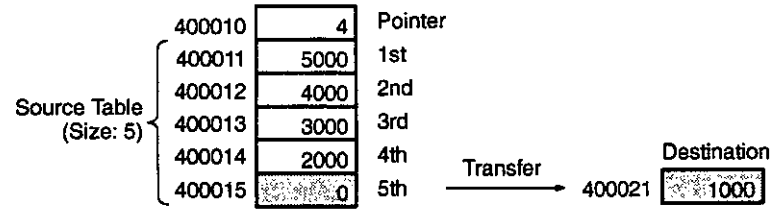
2) The data that was transferred to the table first (the oldest data) is transferred to the destination.

3) The FIRST OUT instruction is usually used to read data entered into the table with the FIRST IN instruction.

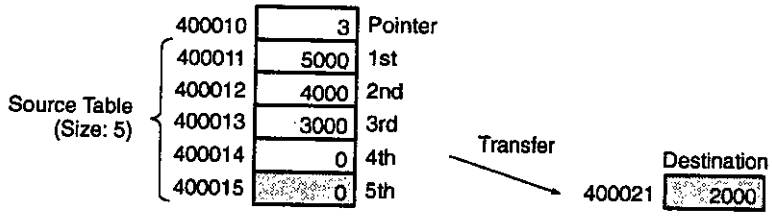
Example
This example shows the operation of the FIRST OUT instruction and the changes that occur in the source table, pointer, and destination.



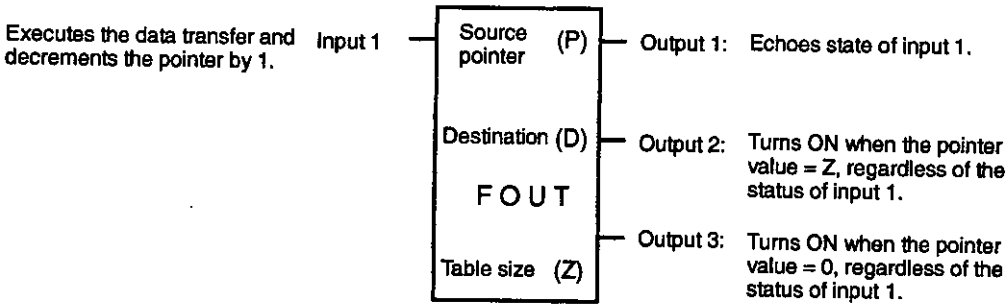
a) When the instruction is executed, the data in the 5th register of the source table is transferred to the destination, the pointer value is decremented to 4, and the 5th register is set to 0.



b) When the instruction is executed again, the data in the 4th register of the source table is transferred to the destination, the pointer value is decremented to 3, and the 4th register is set to 0.



2. Structure



1) FOUT is the symbol for FIRST OUT.

2) FOUT requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 3.10* lists the reference numbers and constants that can be specified. The leading register in the source table is the one just after the pointer.

Example

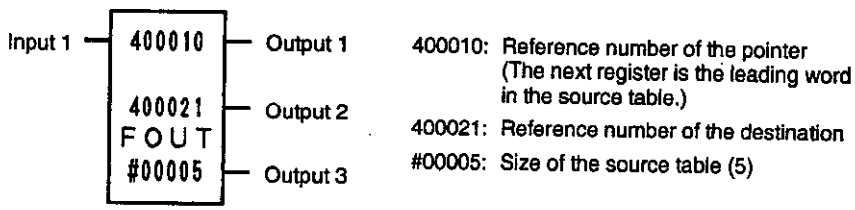


Table 3.10 Structural Elements of FOUT

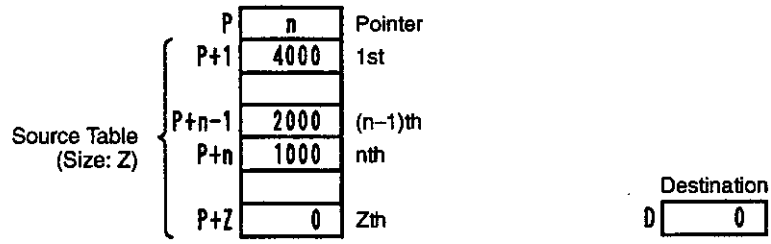
Element	Meaning	Possible Settings	
Top (P)	Reference number of the pointer	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 or R21001 to R21023
Middle (D)	Reference number of the destination	Coil:	000001 to 008161 (O00001 to O08161)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11023 or R21001 to R21023
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
Bottom (Z)	Size of the source table	Constant:	#00001 to #00100

Note (1) When a coil or relay is being specified, the last 5 digits of the reference number must be16n +1 (where n = 0, 1, 2, ...).

(2) The source table starts from the register just after the pointer.

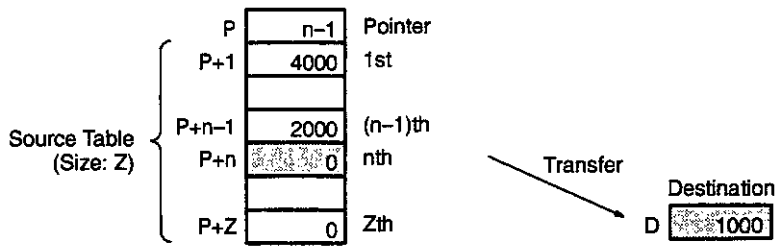
3. Operation

1) Before Execution



2) Pointer Value (n): 1 ≤ n ≤ Z

If the pointer value (n) is1 ≤ n ≤ Z, the following data transfer will be executed when input 1 turns ON. The transfer is completed in one scan.



a) The data in the n^{th} register of the source table is copied to the destination.

b) After the source data is transferred, the n^{th} register is set to 0.

c) The pointer value is decremented by 1.

d) The status of the outputs is as follows:

Output 1: ON
Output 2: OFF
Output 3: Transfer result. ON only when n=0.

3) Pointer Value (n): n=0

If the pointer value is 0, the data transfer won't be executed even when input 1 turns ON. The status of the outputs is as follows:

Output 1: ON
Output 2: OFF
Output 3: ON

4) The pointer value must be $0 \leq n \leq Z$, regardless of the status of input 1. If the pointer value is less than zero, it will be set to zero; if it is greater than Z, it will be set to Z.

5) The pointer value indicates how many words of data can be read by the FIRST OUT instruction.

6) The following table shows the operation of the FOUT instruction for combinations of input 1 and the pointer values. The pointer value is "n" and the destination table size is "Z."

Table 3.11 FOUT Operation

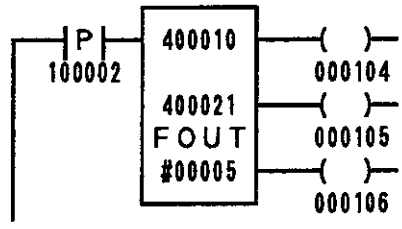
Input 1	Condition of n	FOUT Operation	Outputs		
			1	2	3
ON	n = 1	1) The data in the n^{th} register of the source table is transferred to the destination. 2) After the source data is transferred, the n^{th} register is set to 0. 3) The pointer value (n) is decremented by 1 after the transfer.	ON	OFF	ON
	$2 \leq n \leq Z$			OFF	OFF
	n = 0	1) The transfer isn't executed. 2) The pointer value (n) isn't changed.		OFF	ON
OFF	$1 \leq n \leq Z-1$	1) The transfer isn't executed. 2) The pointer value (n) isn't changed.	OFF	OFF	OFF
	n = Z			ON	OFF
	n = 0			OFF	ON

Note The pointer value must be $0 \leq n \leq Z$, regardless of the status of input 1. If the pointer value is less than zero, it will be set to zero; if it is greater than Z, it will be set to Z.

◀EXAMPLE▶

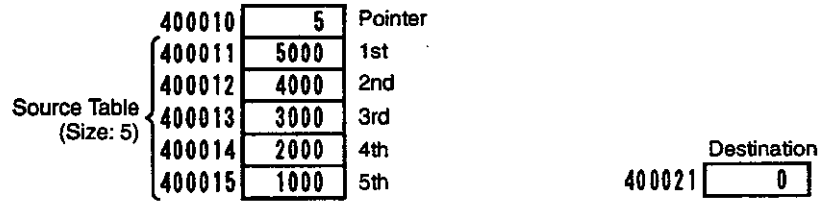
4. Application Example

1) Ladder Programming

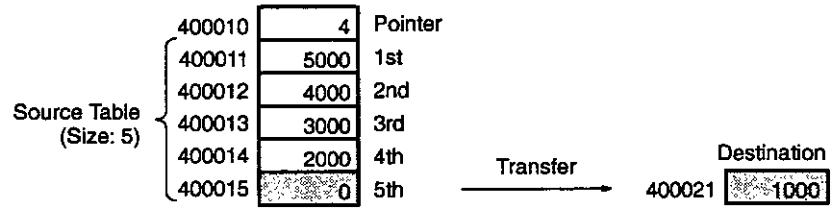


2) Transfer Operation

a) Status Before Execution



b) The following data transfer is executed when input relay 100002 changes from OFF to ON. The transfer is completed in one scan.



(1) The data in the 5th register of the source table (1000) is copied to the destination.

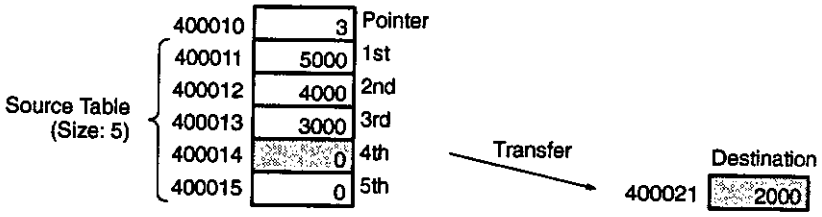
(2) After the transfer, the 5th register is set to 0.

(3) The pointer value is decremented to 4.

(4) The status of the outputs is as follows:
Coil 000104:
Turns ON only in scan where input 100002 changes from OFF to ON.

Coil 000105:
Changes from ON to OFF.
Coil 000106:
Remains OFF.

c) The following data transfer is executed when input relay 100002 is turned from OFF to ON again. The transfer is completed in one scan.



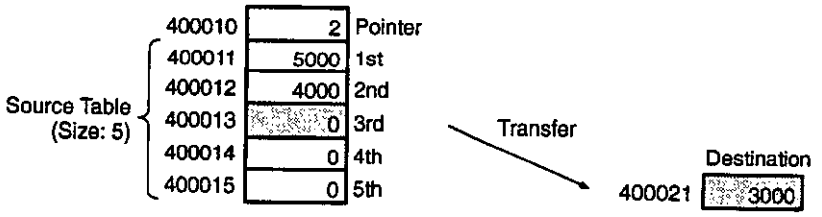
(1) The data in the 4th register of the source table (2000) is copied to the destination.

(2) After the transfer, the 4th register is set to 0.

(3) The pointer value is decremented to 3.

(4) The status of the outputs is as follows:
Coil 000104:
Turns ON only in scan where input 100002 changes from OFF to ON.
Coil 000105:
Remains OFF.
Coil 000106:
Remains OFF.

d) The following data transfer is executed when input relay 100002 is turned from OFF to ON again. The transfer is completed in one scan.



(1) The data in the 3rd register of the source table (3000) is copied to the destination.

(2) After the transfer, the 3rd register is set to 0.

(3) The pointer value is decremented to 2.

(4) The status of the outputs is as follows:
Coil 000104:

Turns ON only in scan where input 100002 changes from OFF to ON.

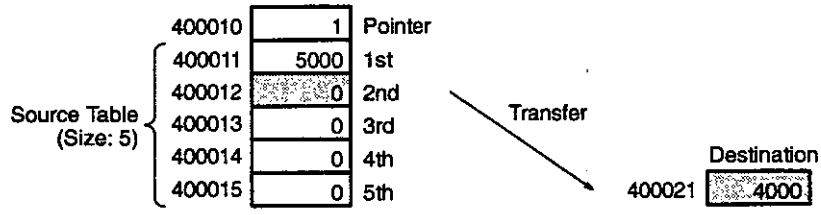
Coil 000105:

Remains OFF.

Coil 000106:

Remains OFF.

e) The following data transfer is executed when input relay 100002 is turned from OFF to ON again. The transfer is completed in one scan.



(1) The data in the 2nd register of the source table (4000) is copied to the destination.

(2) After the transfer, the 2nd register is set to 0.

(3) The pointer value is decremented to 1.

(4) The status of the outputs is as follows:

Coil 000104:

Turns ON only in scan where input 100002 changes from OFF to ON.

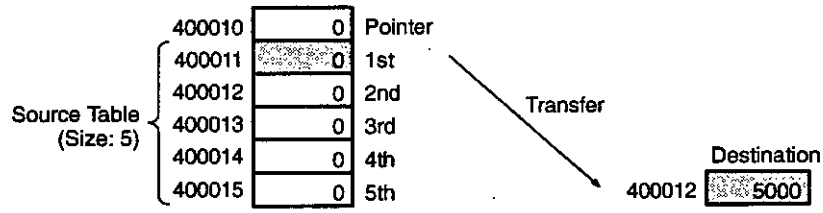
Coil 000105:

Remains OFF.

Coil 000106:

Remains OFF.

f) The following data transfer is executed when input relay 100002 is turned from OFF to ON again. The transfer is completed in one scan.



(1) The data in the 1st register of the source table (5000) is copied to the destination.

(2) After the transfer, the 1st register is set to 0.

(3) The pointer value is decremented to 0.

(4) The status of the outputs is as follows:

Coil 000104:

Turns ON only in scan where input 100002 changes from OFF to ON.

Coil 000105:
Remains OFF.

Coil 000106:
Turns ON.

g) Since there isn't any data stored in the table, the data transfer won't be executed if input relay 100002 is turned from OFF to ON again. Also, the pointer value will remain unchanged (n=0) and the outputs will have the following status:

Coil 000104: Turns ON only in scan where input 100002 changes from OFF to ON.
Coil 000105: Remains OFF.
Coil 000106: Remains ON.

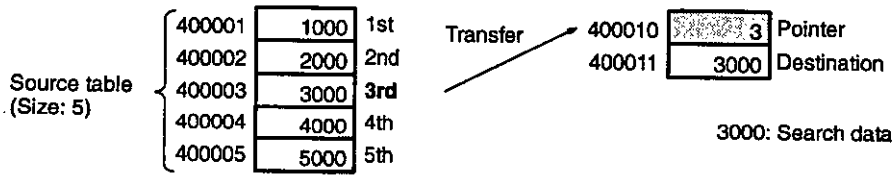
3.3.6 TABLE SEARCH (SRCH)

1. Function

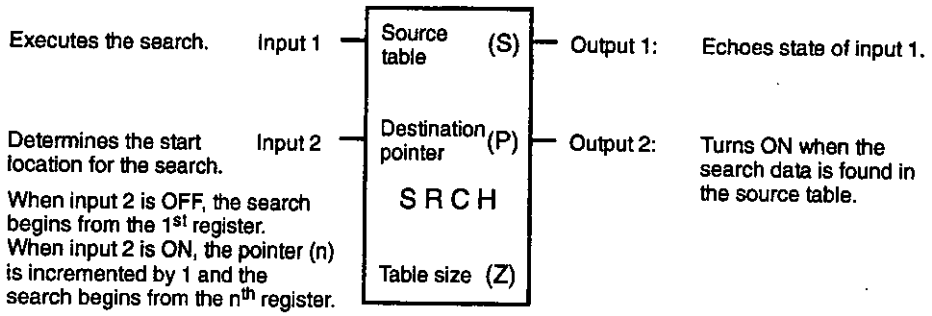
1) The source is a data table and the destination is a single register (the next register after the pointer). When the specified search data is found in the source table, that table position is stored in the pointer.

2) Searches the source table for the search data and writes that table position in the pointer.

Example
This example shows the operation of the TABLE SEARCH instruction and the changes that occur in the source table, pointer, and destination. When the instruction is executed after setting the search data (3000) in the destination, the instruction searches the table for the same data and writes that position (3) in the pointer.



2. Structure



1) SRCH is the symbol for TABLE SEARCH.

2) SRCH requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 3.12* lists the register reference numbers and constants that can be specified. The destination is the register just after the pointer.

Example

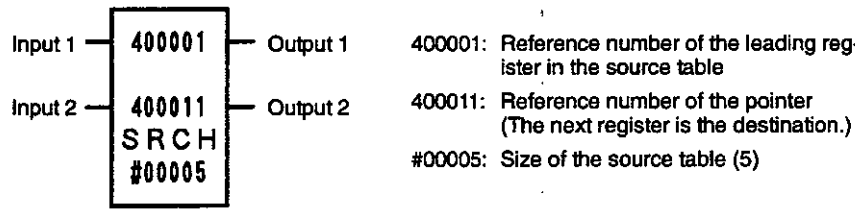


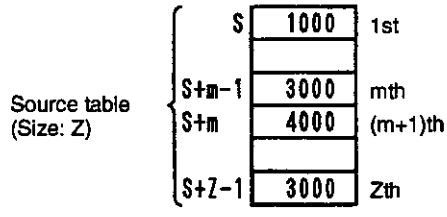
Table 3.12 Structural Elements of SRCH

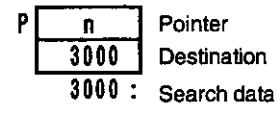
Element	Meaning	Possible Settings	
Top (S)	Reference number of the leading register in the source table	Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 R20001 to R21024
Middle (P)	Reference number of the pointer	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 R20001 to R21023
Bottom (Z)	Size of the source table	Constant:	#00001 to #00100

Note The register just after the pointer is the destination register.

3. Operation

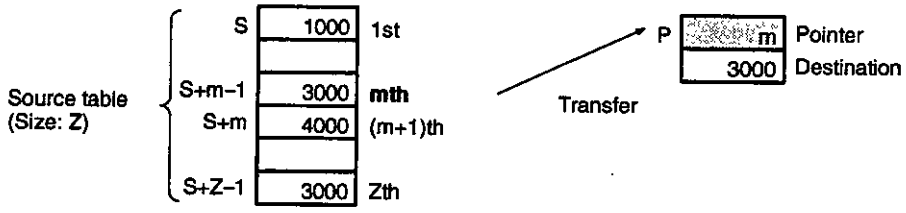
1) Before Execution





The search data can be found in the m^{th} and z^{th} register of the source table.

2) The following data transfer will be executed when input 1 alone is turned ON. The transfer is completed in one scan.



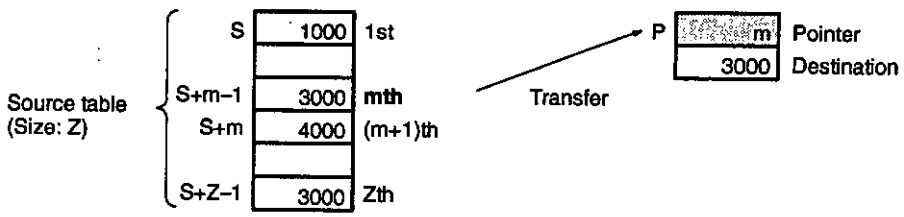
a) The search begins from the 1st register in the source table, regardless of the pointer's original value (n).

b) In every scan, the first match for the search data is found in register m and this position is written to the pointer. Outputs 1 and 2 are both turned ON.

c) Even if several registers in the source table contain the search data, only the first register containing the search data is found.

3) The following data transfer will be executed when both input 1 and input 2 are turned ON. The transfer is completed in one scan.

a) Pointer Value (n): n < m



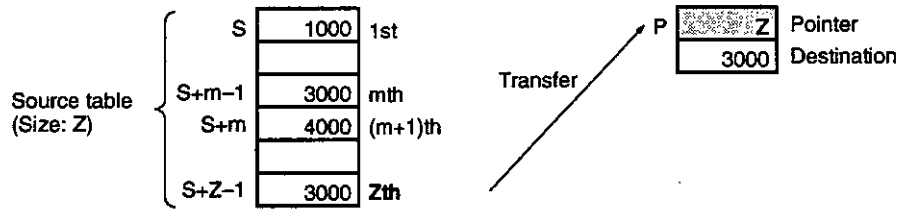
(1) In the first scan, the search begins from the $(n+1)^{th}$ register in the source table. The first match for the search data is found in register m and this position is written to the pointer. Outputs 1 and 2 are both turned ON.

(2) In the next scan, the search begins from the $(m+1)^{th}$ register in the source table. The first match for the search data is found in register Z and this position is written to the pointer. Outputs 1 and 2 both remain ON.

(3) In the next scan, the search begins from the leading register in the source table. The first match for the search data is found in register m and this position is written to the pointer. Outputs 1 and 2 both remain ON.

(4) Steps 2 and 3 repeat in subsequent scans.

b) Pointer Value (n): n ≥ m



(1) In the first scan, the search begins from the $(n+1)^{th}$ register in the source table. The first match for the search data is found in register Z and this position is written to the pointer. Outputs 1 and 2 are both turned ON.

(2) In the next scan, the search begins from the leading register in the source table. The first match for the search data is found in register m and this position is written to the pointer. Outputs 1 and 2 both remain ON.

(3) In the next scan, the search begins from the $(m+1)^{th}$ register in the source table. The first match for the search data is found in register Z and this position is written to the pointer. Outputs 1 and 2 both remain ON.

(4) Steps 2 and 3 are repeated in subsequent scans.

4) If the source table doesn't contain the search data, the pointer is set to 0 and that scan's search ends.

5) The pointer value must be $0 \leq n \leq Z$, regardless of the status of input 1. If the pointer value is less than zero, it will be set to zero; if it is greater than Z, it will be set to Z.

6) The following table shows the operation of the SRCH instruction for all combinations of inputs 1 and 2 and the pointer values. The pointer value is n and the source table size is Z.

Table 3.13 SRCH Operation

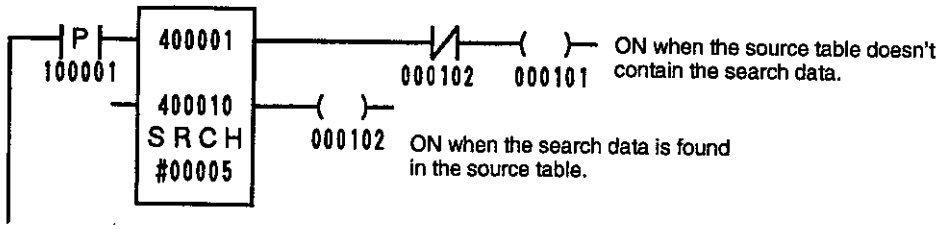
Inputs 1	Inputs 2	Condition of n	SRCH Operation	Outputs 1	Outputs 2
ON	OFF	$0 \leq n \leq Z$	1) The search begins from the 1st register in the source table. 2) When the search data is found, that position (m) is transferred to the pointer and the search is completed for that scan. 3) When the source table doesn't contain the search data, the pointer is set to 0 and the search is completed for that scan.	ON	1) **ON when the search data is found.** 2) OFF when the source table doesn't contain the search data.
	ON	n = Z			
		$0 \leq n \leq Z-1$	1) The search begins from the (n+1)th register in the source table. 2) When the search data is found, that position (m) is transferred to the pointer and the search is completed for that scan. 3) When the source table doesn't contain the search data, the pointer is set to 0 and the search is completed for that scan.		
OFF	Any	None	1) The search isn't executed. 2) The pointer value (n) is set to 0.	OFF	OFF

Note The pointer value must be $0 \leq n \leq Z$, regardless of the status of inputs 1 and 2. If the pointer value is less than zero, it will be set to zero; if it is greater than Z, it will be set to Z.

4. Application Examples

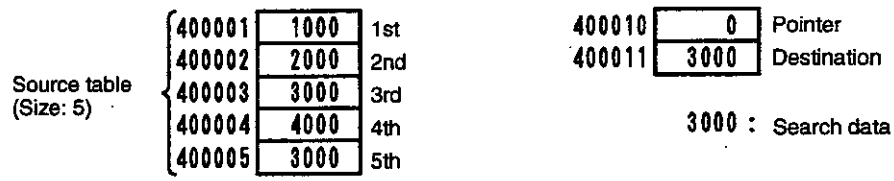
◀EXAMPLE▶ **Example 1**

1) Ladder Programming

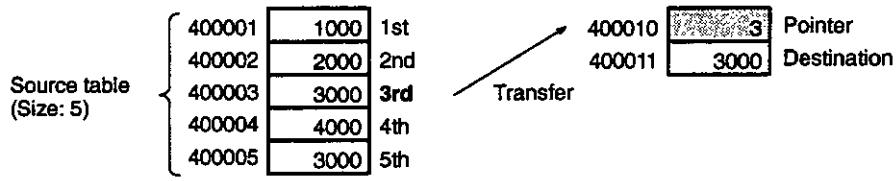


2) Transfer Operation

a) Before Transfer



b) The following data transfer is executed when input relay 100001 changes from OFF to ON. The transfer is completed in one scan.



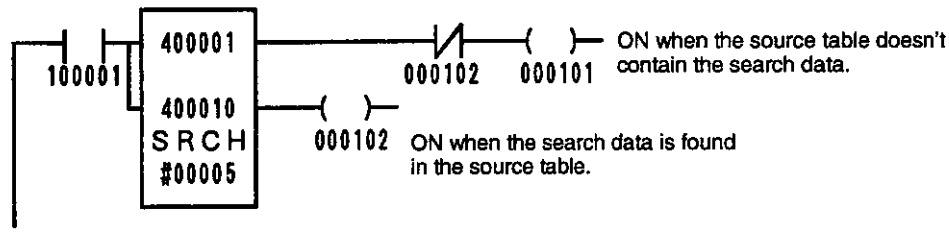
(1) The search begins from the leading register in the source table. The first match for the search data is found in the third register and this position (3) is written to the pointer.

(2) The status of the outputs is as follows:
Coil 000101:
Remains OFF.
Coil 000102:
Turns ON only in scan in which input 100001 changes from OFF to ON.

(3) Since input 2 is OFF, the search begins from the leading register in the source table and always finds the search data in the third register. The search data in the fifth register can't be found with this ladder programming.

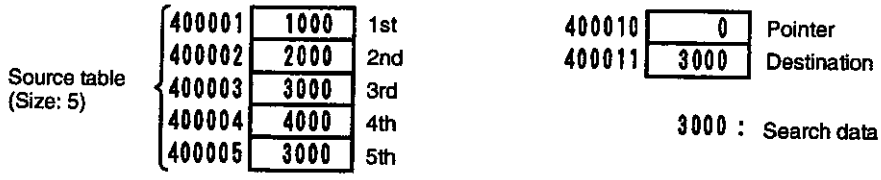
◀EXAMPLE▶ **Example 2**

1) Ladder Programming

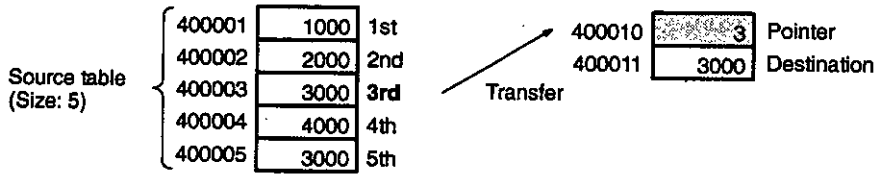


2) Transfer Operation

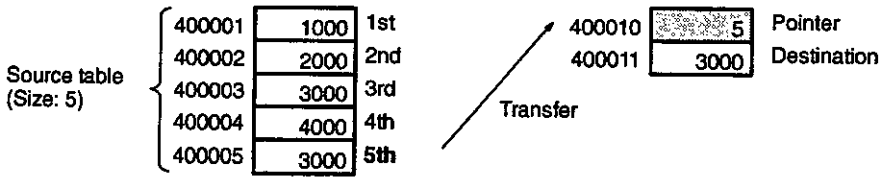
a) Before Transfer



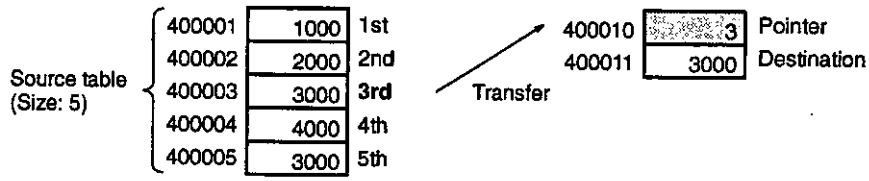
b) When input relay 100001 changes from OFF to ON, the search begins from the leading register in the source table. The first match for the search data is found in the third register, this position (3) is written to the pointer, and this scan's search is completed. Coil 000101 is OFF and coil 000102 is ON.



c) In the next scan, the search begins from the fourth register in the source table. The search data is found in the fifth register, this position (5) is written to the pointer, and this scan's search is completed.
Coil 000101 is OFF and coil 000102 is ON.



d) In the next scan, the search begins from the leading register in the source table. The search data is found in the third register, this position (3) is written to the pointer, and this scan's search is completed.
Coil 000101 is OFF and coil 000102 is ON.



e) As long as input relay 100001 is ON, steps 2 through 4 are repeated in subsequent scans.

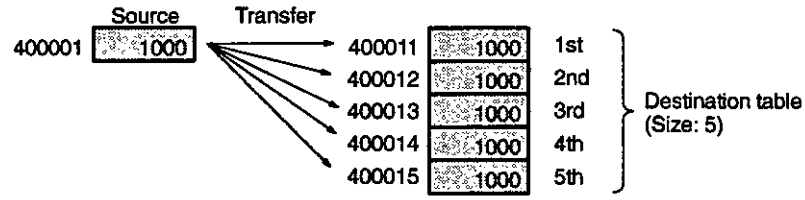
3.3.7 TABLE SET (TSET)

1. Function

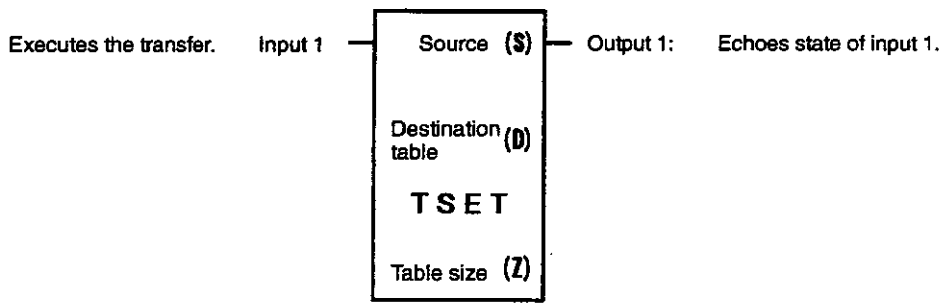
1) The source is a single register and the destination is a data table. There is no pointer.

2) Copies the content of the source (1 word of data) to all of the registers in the destination table. The transfer is completed in one scan.

Example
This example shows the operation of the TABLE SET instruction and the changes that occur in the source and destination table. When the instruction is executed, the source data is copied to all of the registers in the destination table.



2. Structure



1) TSET is the symbol for TABLE SET.

2) TSET requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 3.14* lists the register reference numbers and constants that can be specified.

Example

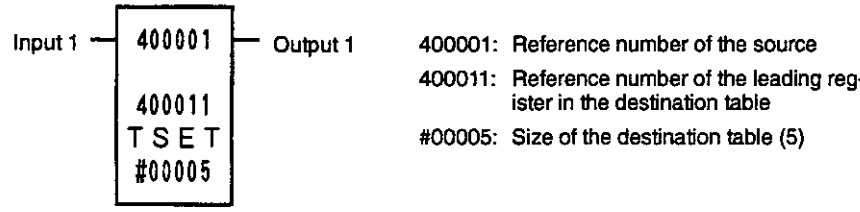
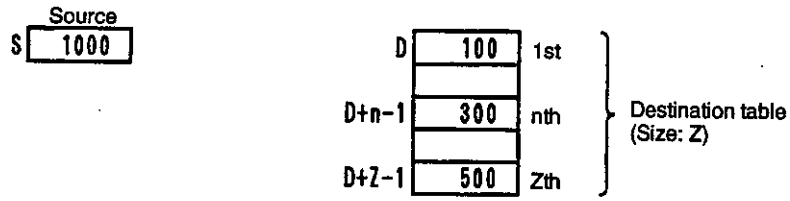


Table 3.14 Structural Elements of TSET

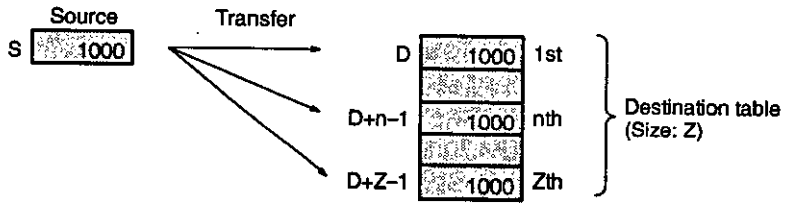
Element	Meaning	Possible Settings	
Top (S)	Reference number of the source	Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 R20001 to R21024
Middle (D)	Reference number of the leading register in the destination table	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 R20001 to R21024
Bottom (Z)	Size of the destination table	Constant:	#00001 to #00100

3. Operation

1) Before Transfer



2) The following data transfer is executed when input 1 is turned ON. The transfer is completed in one scan.



a) The source data is copied to all of the registers in the destination table in one scan.

b) Output 1 is turned ON.

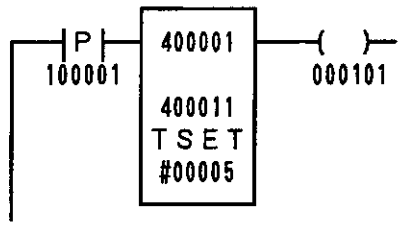
3) The following table summarizes the operation of TSET.

Table 3.15 TSET Operation

Input 1	TSET Operation	Output 1
ON	The source data is copied to all of the registers in the destination table in one scan.	**ON**
OFF	The transfer is not executed.	OFF

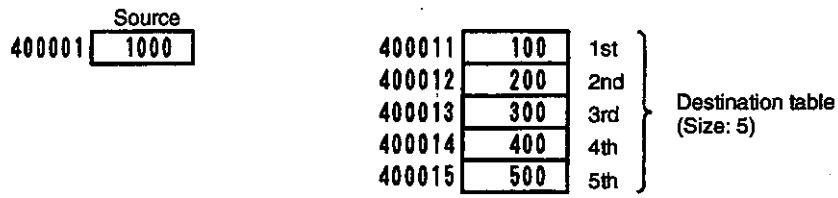
◀EXAMPLE▶ **4. Application Example**

1) Ladder Programming

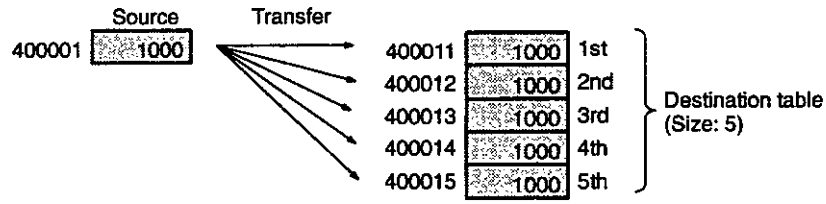


2) Transfer Operation

a) Before Transfer



b) The following data transfer is executed when input relay 100001 changes from OFF to ON. The transfer is completed in one scan.



(1) The source data is copied to all of the registers in the destination table in one scan.

(2) Coil 000101 is ON only in scan in which input 100001 changes from OFF to ON.

3.3.8 BLOCK MOVE (BLKM)

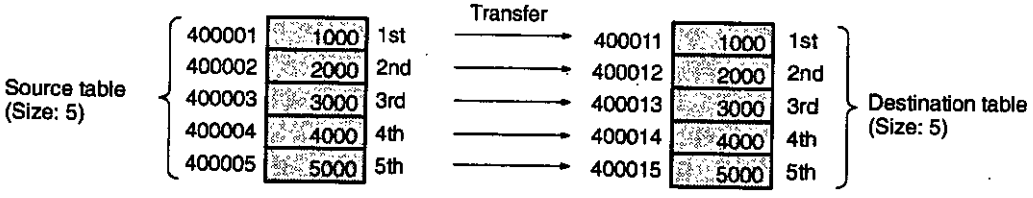
1. Function

1) The source and destination are data tables of the same size. There is no pointer.

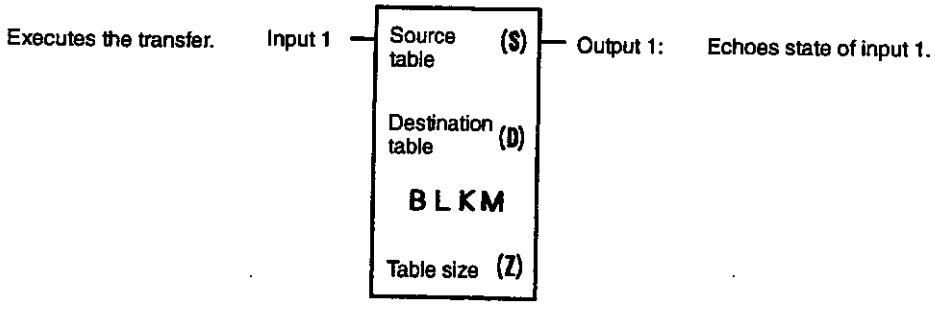
2) Copies the contents of the registers in the source table to the corresponding registers in the destination table in a single scan.

Example

This example shows the operation of the BLOCK MOVE instruction and the changes that occur in the source and destination tables. When the instruction is executed, all of the data in the source table is copied to the destination table in one scan.



2. Structure



1) BLKM is the symbol for BLOCK MOVE.

2) BLKM requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 3.16* lists the register reference numbers and constants that can be specified.

Example

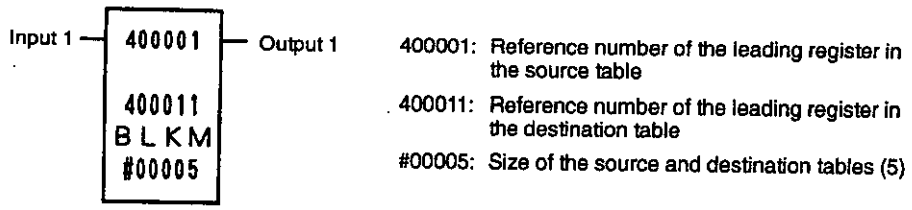


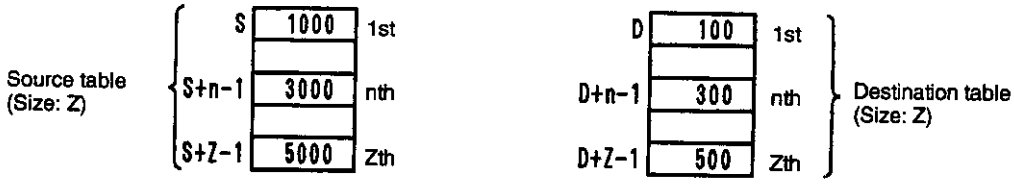
Table 3.16 Structural Elements of BLKM

Element	Meaning	Possible Settings	
Top (S)	Reference number of the leading register in the source table	Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (D)	Reference number of the leading register in the destination table	Coil:	000001 to 008161 (O00001 to O08161)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
Bottom (Z)	Size of the source and destination tables	Specify the constant. The maximum value of the constant differs with specified reference type.	
		Coil:	#00001 to #00100
		Input relay:	#00001 to #00064
		Input register, holding register or constant register:	#00001 to #00100
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00100
		MC coil, MC relay or MC control relay:	#00001 to #00016
		MC control coil:	#00001 to #00010
		M code relay:	#00001 to #00006

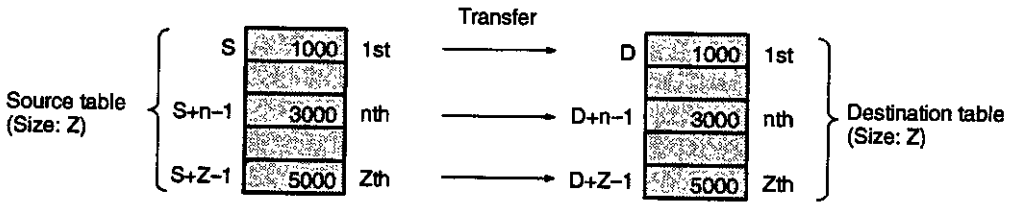
Note When a coil or relay is being specified, the lower 5 digits of the reference number must be16n +1 (where n = 0, 1, 2, ...).

3. Operation

1) Before Transfer



2) The following data transfer is executed when input 1 is turned ON. The transfer is completed in one scan.



a) All of the data in the source table is copied to the corresponding registers in the destination table. The transfer is completed in one scan.

b) The contents of the n^{th} (n = 1 to Z) register of the source table are copied to the n^{th} (n = 1 to Z) register of the destination table.

c) Output 1 is turned ON.

3) The following table summarizes the operation of BLKM.

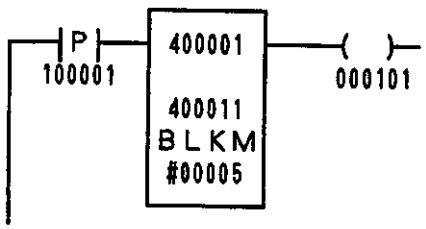
Table 3.17 BLKM Operation

Input 1	BLKM Operation	Output 1
ON	All of the data in the source table is copied to the corresponding registers in the destination table. The transfer is completed in one scan.	ON
OFF	The transfer is not executed.	OFF

4. Application Examples

◀EXAMPLE▶ **Example 1**

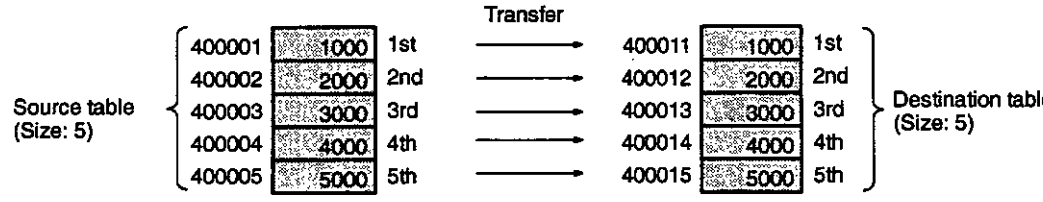
1) Ladder Programming



2) Transfer Operation

a) Before Transfer

b) The following data transfer is executed when input relay 100001 changes from OFF to ON. The transfer is completed in one scan.



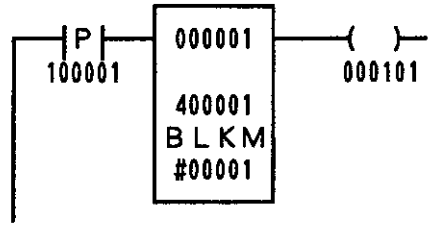
(1) All of the data in the source table is copied to the corresponding registers in the destination table.

(2) The contents of the n^{th} (n = 1 to 5) register of the source table are copied to the n^{th} (n = 1 to 5) register of the destination table.

(3) Coil 000101 is ON only in scan in which input 100001 changes from OFF to ON.

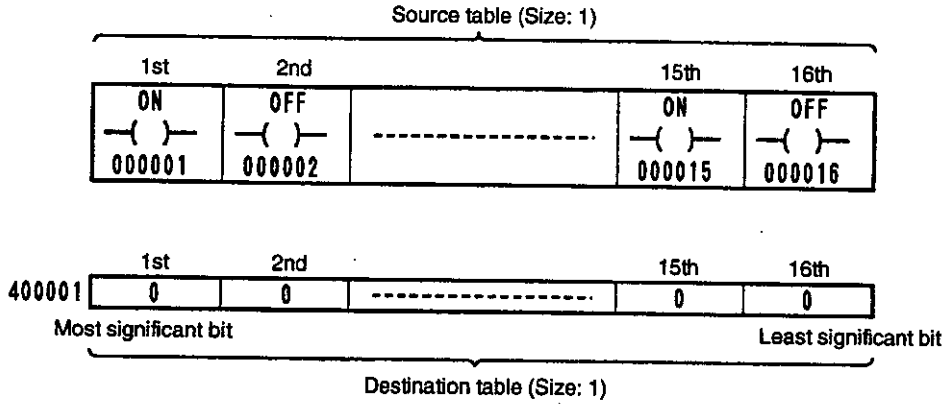
◀EXAMPLE▶ **Example 2**

1) Ladder Programming

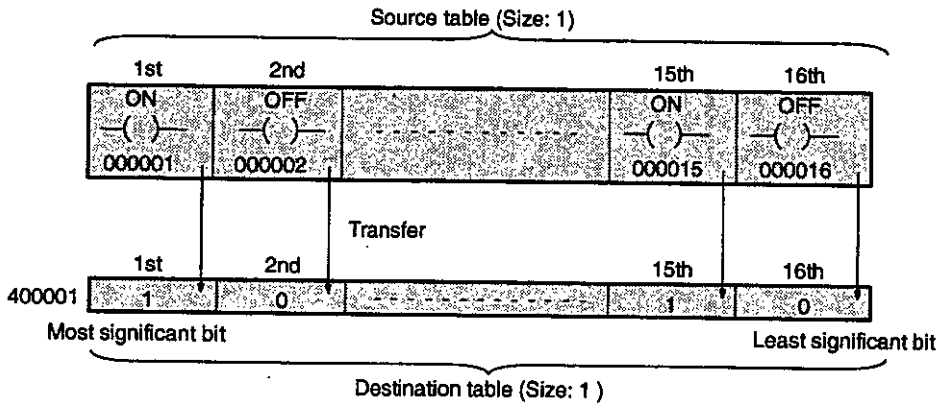


2) Transfer Operation

a) Before Transfer



b) The following data transfer is executed when input relay 100001 changes from OFF to ON. The transfer is completed in one scan.

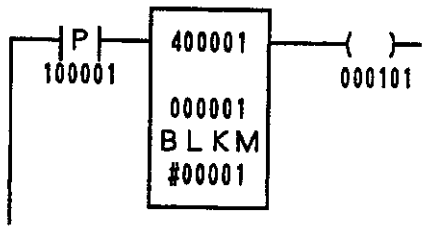


(1) The status of the 16 coils in the source table are transferred to the corresponding 16 bits in the destination table. When a coil is ON, the corresponding bit is set to 1; when a coil is OFF, the corresponding bit is set to 0.

(2) Coil 000101 is ON only in scan in which input 100001 changes from OFF to ON.

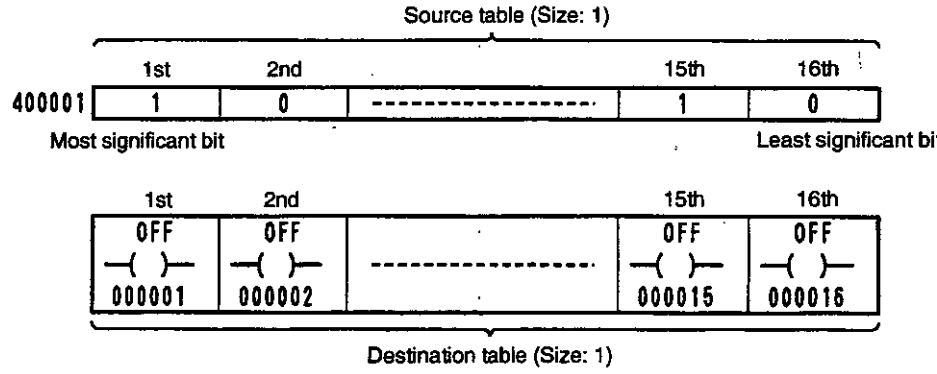
◀EXAMPLE▶ **Example 3**

1) Ladder Programming

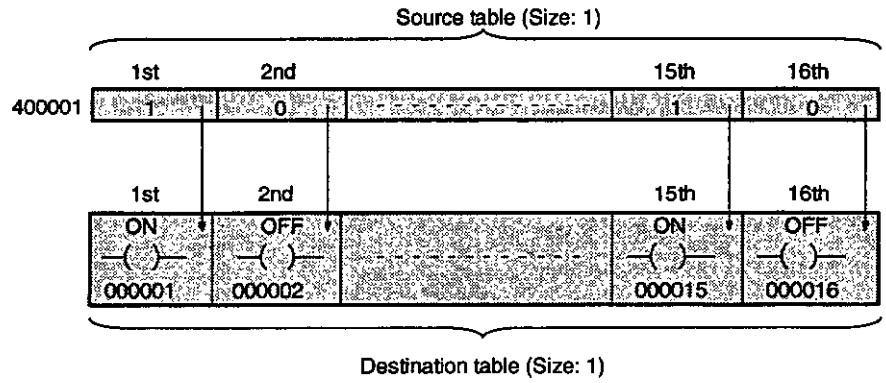


2) Transfer Operation

a) Before Transfer



b) The following data transfer is executed when input relay 100001 changes from OFF to ON. The transfer is completed in one scan.



(1) The status of the 16 bits in the source table are transferred to the corresponding 16 coils in the destination table. When a bit is "1", the corresponding coil is turned ON; when a bit is "0", the corresponding coil is turned OFF.

(2) Coil 000101 is ON only in scan in which input 100001 changes from OFF to ON.

Note When coils 000001 through 000016 are used like they are in the BLKM instruction in this example, the same reference numbers cannot be programmed as coils in other locations in the program.

3.3.9 BLOCK-TO-TABLE MOVE (BLKT)

1. Function

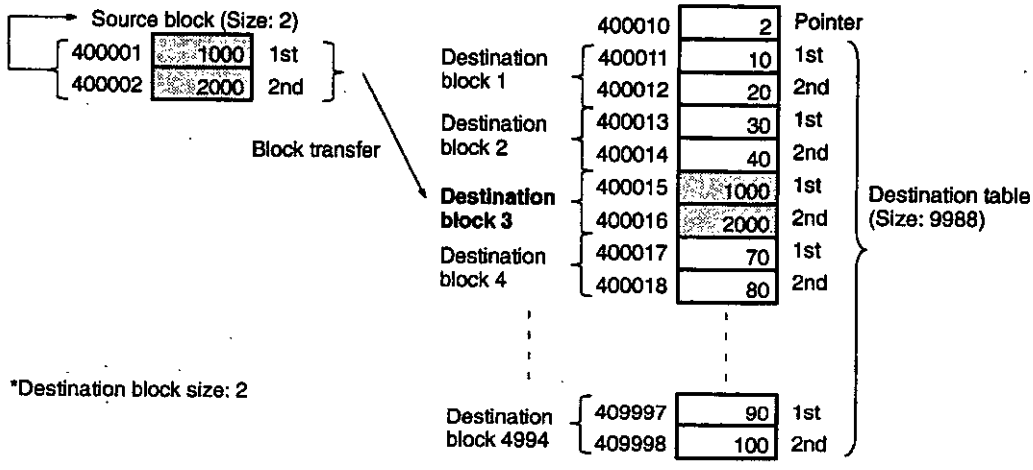
1) Data is transferred from a data block to a data table. The pointer is the register just before the destination table and the pointer value determines which block in the table is the destination block.

2) The source block is a single data block. (The size of a data block is defined by the user in the third element of the instruction.)

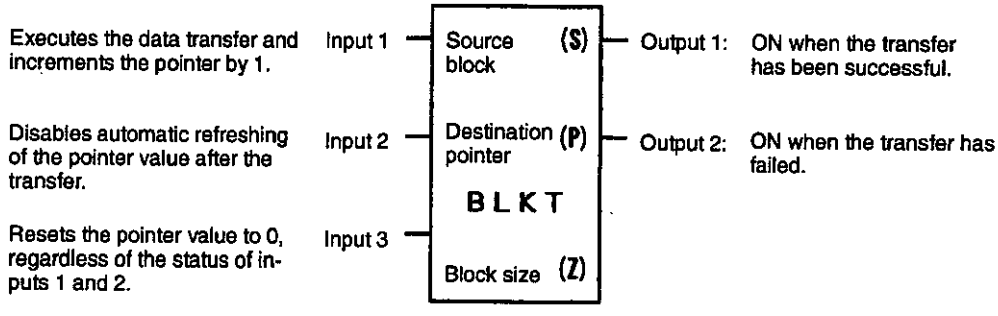
3) The destination table is made up of a series of data blocks which are referred to as destination block 1, destination block 2, destination block 3, etc.

4) The data in the source block can be copied to any destination block in the destination table by adjusting the pointer value. The transfer is completed in one scan.

Example
This example shows the operation of the BLOCK-TO-TABLE MOVE instruction and the changes that occur in the source block, pointer, and destination table. When the instruction is executed with a pointer value of 2, the data in the source block is copied to destination block 3 (400015 and 400016).



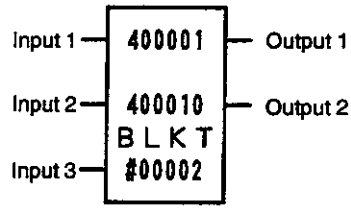
2. Structure



1) BLKT is the symbol for BLOCK-TO-TABLE MOVE.

2) BLKT requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 3.18* lists the register reference numbers and constants that can be specified. The register just after the pointer is the leading register in the destination table.

Example



400001: Reference number of the leading register in the source block

400010: Reference number of the pointer (The next register is the leading register in the destination table.)

#00002: Size of the source and destination blocks

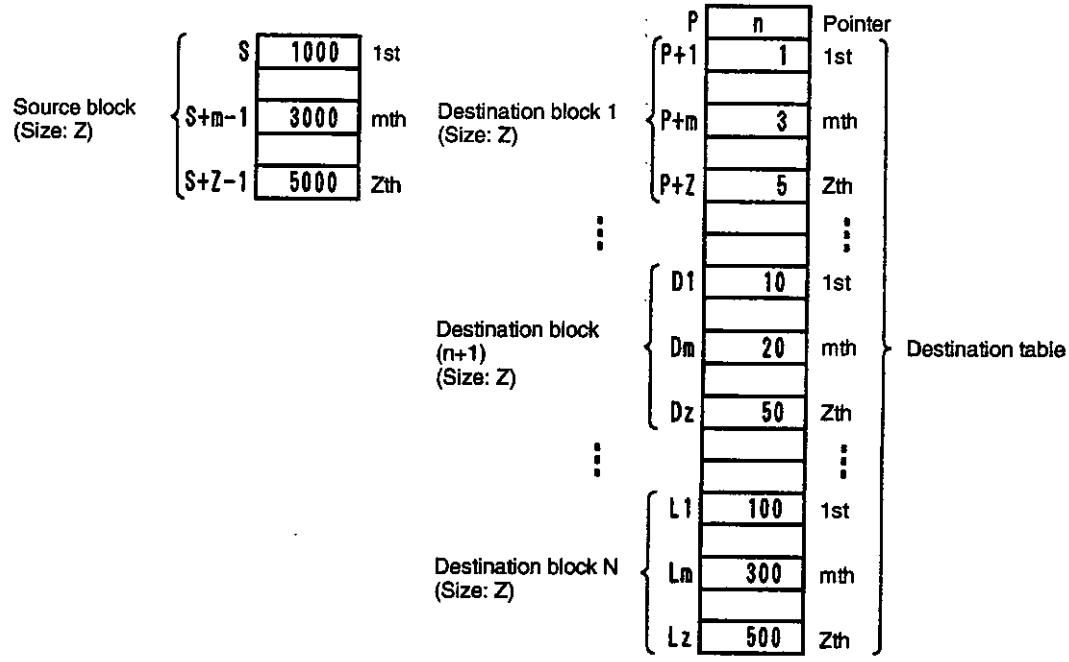
Table 3.18 Structural Elements of BLKT

Element	Meaning	Possible Settings	
Top (S)	Reference number of the leading register in the source block	Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (P)	Reference number of the pointer	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 or R20001 to R21023
Bottom (Z)	Size of the source and destination blocks	Specify the constant. The maximum value of the constant differs with specified reference type.	
		Coil:	#00001 to #00100
		Input relay:	#00001 to #00064
		Input register, holding register or constant register:	#00001 to #00100
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00100
		MC coil, MC relay or MC control relay:	#00001 to #00016
		MC control coil:	#00001 to #00010
		M code relay:	#00001 to #00006

Note (1) When a coil or relay is being specified, the lower 5 digits of the reference number must be16n +1 (where n = 0, 1, 2, ...).

(2) The destination table starts from the register just after the pointer.

3) The Destination Table and Destination Blocks



The destination table and destination blocks are determined by the source block size (Z) and the reference number of the pointer (P), as shown below.

a) Destination Table

(1) Reference number of the leading register $= P + 1$

(2) Reference number of the last register (L_{Z}) $= P + \left[\frac{409999 - P}{Z}\right] \times Z$

The [ ] symbols are Gauss' notation. For any real number V, the term [V] yields the maximum integer that doesn't exceed V. For example: [1.5] = 1.

b) Destination Blocks

(1) The number of blocks (N) $= \left[\frac{409999 - P}{Z}\right]$

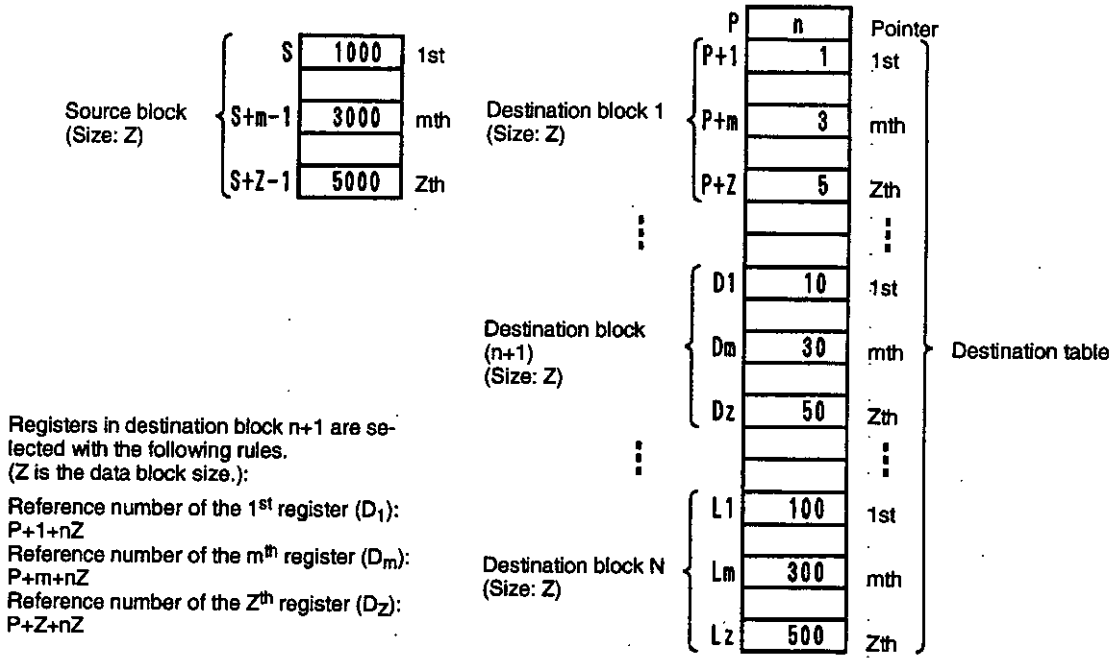
(2) The reference numbers of registers in block n+1 are calculated from the following equations:
Reference number of the 1st register (D_1) = P+1+nZ
Reference number of the m^{th} register (D_m) = P+m+nZ
Reference number of the Z^{th} register (D_Z) = P+Z+nZ

4) Valid Pointer Values

If the number of destination blocks is N, the valid pointer values (n) are: $0 \leq n \leq N-1$.

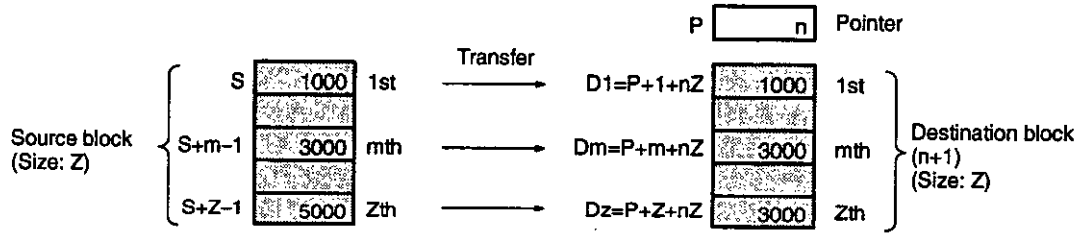
3. Operation

1) Before Execution



2) Operation with a Valid Pointer Value (0 ≤ n ≤ N–1):

If the pointer value is valid, the following data transfer will be executed when input 1 turns ON. The transfer is completed in one scan.



a) The block of registers D_1 through D_Z is selected as the destination block (destination block n+1) based on the pointer value (n) and the source block size (Z).

b) All of the data in the source block is copied to the corresponding registers of destination block n+1.

c) The pointer value is incremented by 1 if input 2 is OFF; it is left unchanged if input 2 is ON.

d) The status of the outputs is as follows:
Output 1: Turns ON.
Output 2: Remains OFF.

3) Operation with an Invalid Pointer Value (n < 0 or n ≥ N):

If the pointer value is invalid, the data transfer won't be executed even when input 1 turns ON. Output 1 will remain OFF and output 2 will be turned ON.

4) When input 3 is turned ON, the pointer value (n) will be reset to zero, regardless of the status of inputs 1 and 2.

5) The following table shows the operation of the BLKT instruction for all possible input combinations. The pointer value is n and the number of destination blocks is N.

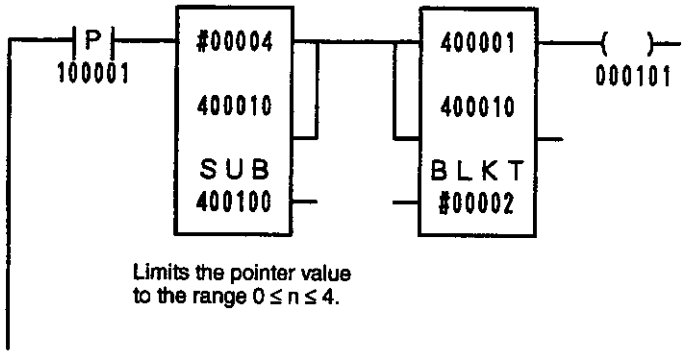
Table 3.19 BLKT Operation

Inputs			Condition of n	BLKT Operation	Outputs	
1	2	3			1	2
ON	OFF	OFF	$0 \leq n \leq N-1$	1) All of the data in the source block is copied to the corresponding registers of destination block n+1. 2) The pointer value (n) is incremented by 1 after the transfer.	**ON**	OFF
			$n < 0, n \geq N$	1) The transfer isn't executed. 2) The pointer value (n) isn't changed.	OFF	**ON**
	ON	OFF	$0 \leq n \leq N-1$	1) The data in the source block is copied to the corresponding registers of destination block n+1. 2) The pointer value (n) isn't changed.	**ON**	OFF
			$n < 0, n \geq N$	1) The transfer isn't executed. 2) The pointer value (n) isn't changed.	OFF	**ON**
	OFF	**ON**	None	1) After resetting the pointer value (n) to 0, the data in the source block is copied to the corresponding registers of destination block 1. 2) The pointer value (n) is incremented to 1 after the transfer.	**ON**	OFF
	ON	**ON**	None	1) After resetting the pointer value (n) to 0, the data in the source block is copied to the corresponding registers of destination block 1. 2) The pointer value (n) isn't changed. (n=0)		
OFF	Any	**ON**	None	1) The transfer isn't executed. 2) The pointer value (n) is reset to 0.	OFF	OFF
		OFF		1) The transfer isn't executed. 2) The pointer value (n) isn't changed.		

◀EXAMPLE▶

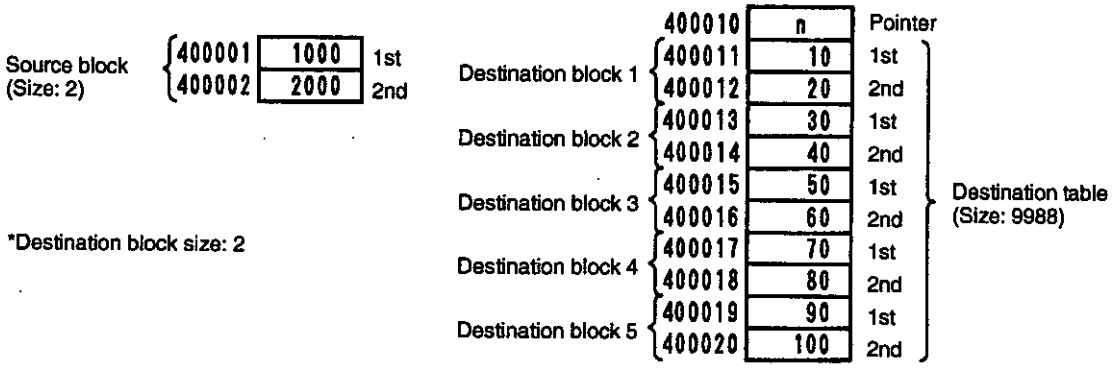
4. Application Example

1) Ladder Programming

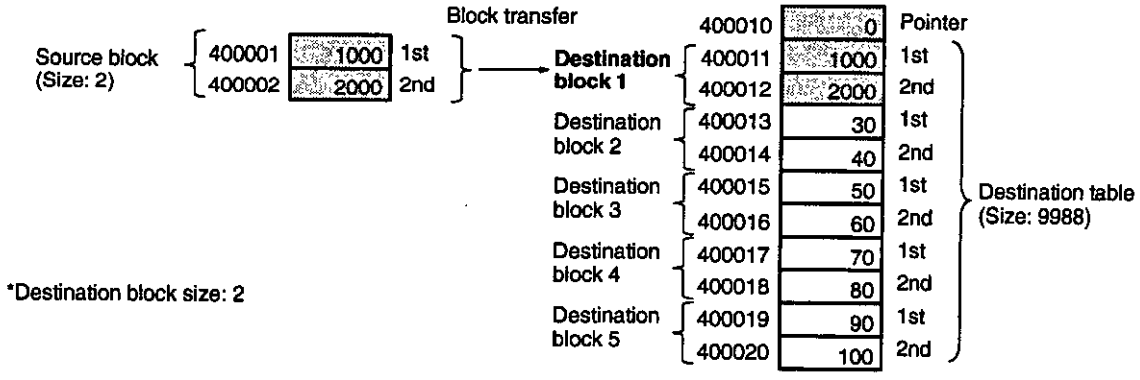


2) Transfer Operation

a) Before Transfer



b) The following data transfer is executed when the pointer value (n) is 0 and input relay 100001 changes from OFF to ON. The transfer is completed in one scan.

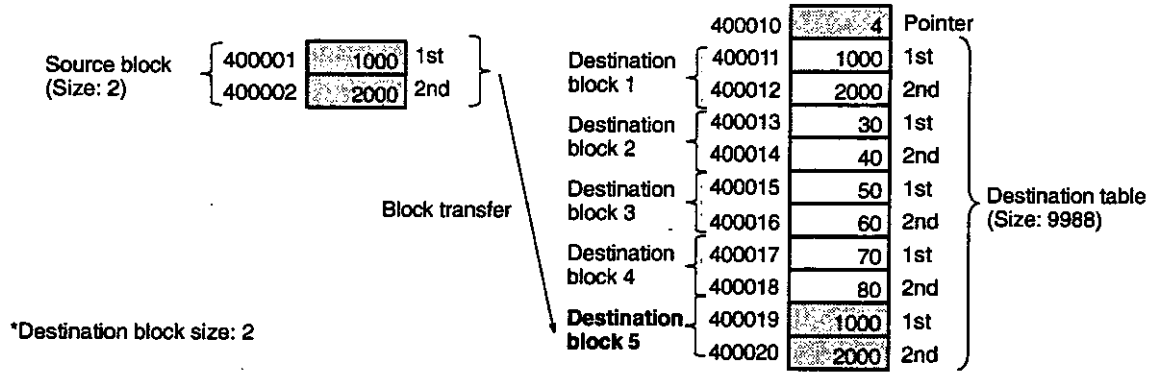


(1) All of the data in the source block is copied to the corresponding registers (400011 and 400012) of destination block 1.

(2) The pointer value is left unchanged.

(3) Coil 000101 turns ON only in scan in which input 100001 changes from OFF to ON.

c) The following data transfer is executed when input relay 100001 changes from OFF to ON and the pointer value is 4 (n=4). The transfer is completed in one scan.



(1) All of the data in the source block is copied to the corresponding registers (400019 and 400020) of destination block 5.

(2) The pointer value is left unchanged.

(3) Coil 000101 turns ON only in scan in which input 100001 changes from OFF to ON.

d) The source block can be copied to any destination block in the table (1 to 5) by setting the pointer value from 0 to 4.

e) In this example, the pointer value is limited to the range $0 \leq n \leq 4$ by the UNSIGNED SINGLE PRECISION DECIMAL SUBTRACTION (SUB) instruction, so the BLKT instruction won't operate if $n<0$ or $n \geq 5$.

3.3.10 TABLE-TO-BLOCK MOVE (TBLK)

1. Function

1) Data is transferred from a data table to a data block. The pointer is the register just before the destination block and the pointer value determines which block in the source table is the source block.

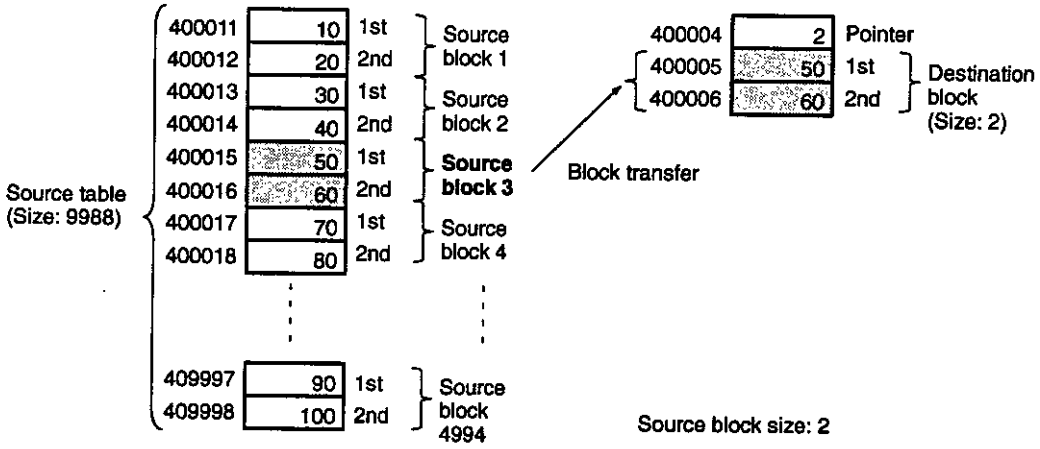
2) The destination block is a single data block. (The size of a data block is defined by the user in the third element of the instruction.)

3) The source table is made up of a series of data blocks which are referred to as source block 1, source block 2, source block 3, etc.

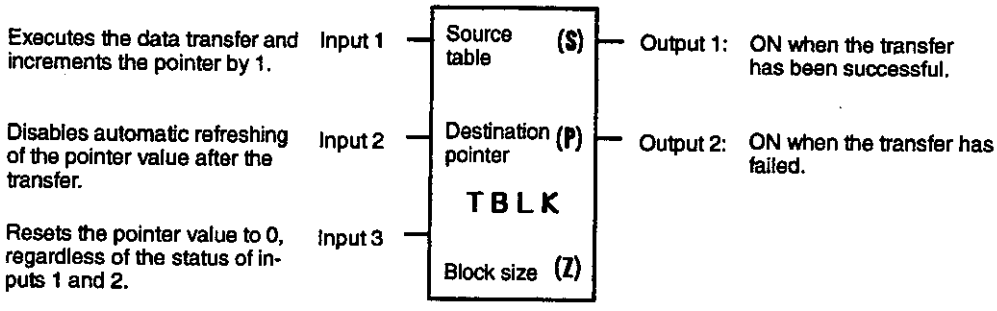
4) The data from any source block in the source table can be copied to the destination block by adjusting the pointer value. The transfer is completed in one scan.

Example
This example shows the operation of the TABLE-TO-BLOCK MOVE instruction and the

changes that occur in the source table, pointer, and destination block. When the instruction is executed with a pointer value of 2, all of the data in source block 3 (400015 and 400016) is copied to the corresponding registers in the destination block.



2. Structure



1) TBLK is the symbol for TABLE-TO-BLOCK MOVE.

2) TBLK requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 3.20* lists the register reference numbers and constants that can be specified. The register just after the pointer is the leading register in the destination block.

Example

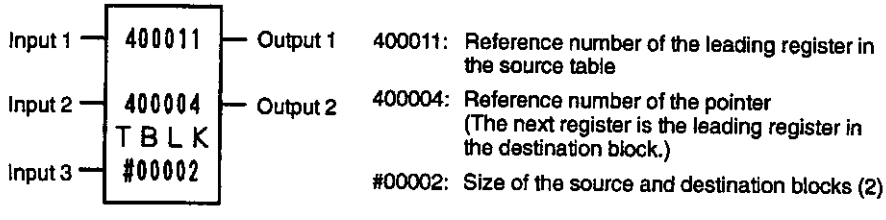


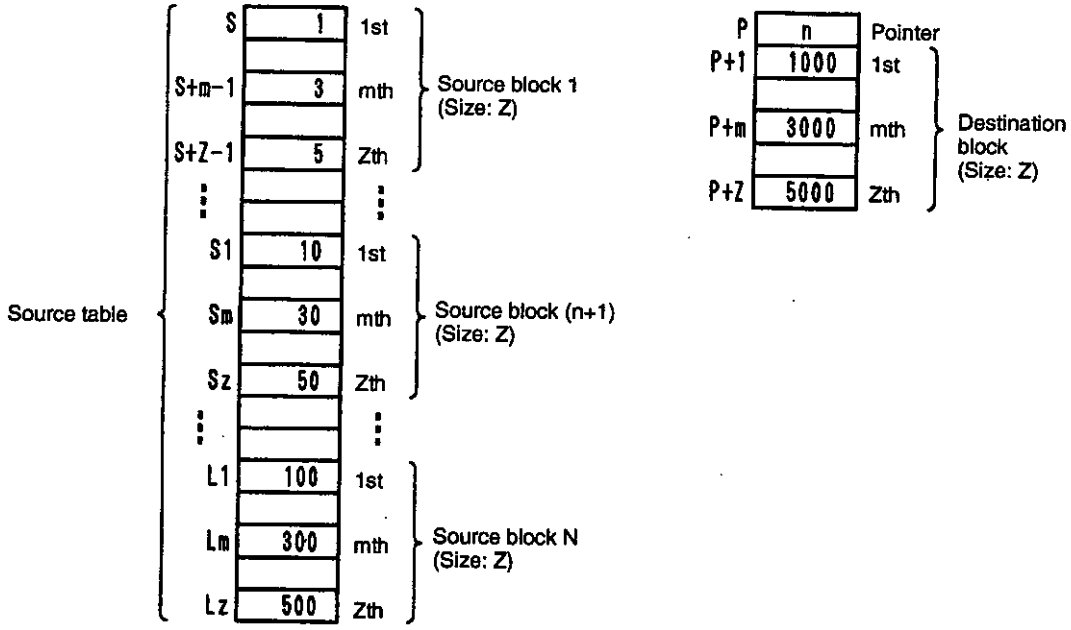
Table 3.20 Structural Elements of TBLK

Element	Meaning	Possible Settings	
Top (S)	Reference number of the leading register in the source table	Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (P)	Reference number of the pointer	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 or R20001 to R21023
Bottom (Z)	Size of the source and destination blocks	Specify the constant. The maximum value of the constant differs with specified reference type.	
		Coil:	#00001 to #00100
		Input relay:	#00001 to #00064
		Input register, holding register or constant register:	#00001 to #00100
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00100
		MC coil, MC relay or MC control relay:	#00001 to #00016
		MC control coil:	#00001 to #00010
		M code relay:	#00001 to #00006

Note (1) When a coil or relay is being specified, the lower 5 digits of the reference number must be16n +1 (where n = 0, 1, 2, ...).

(2) The destination block starts from the register just after the pointer.

3) The Source Table and Source Blocks



The source table and source blocks are determined by the destination block size (Z) and the reference number of the leading register in the source table (S), as shown below.

a) Source Table

(1) Reference number of the leading register $= S$

(2) Reference number of the last register $(L_Z) = S + \left[\frac{409999 - S + 1}{Z}\right] \times Z - 1$

The [ ] symbols are Gauss' notation. For any real number V, the term [V] yields the maximum integer that doesn't exceed V. For example: [1.5] = 1.

b) Source Blocks

(1) The number of blocks (N) $= \left[\frac{409999 - S + 1}{Z}\right]$

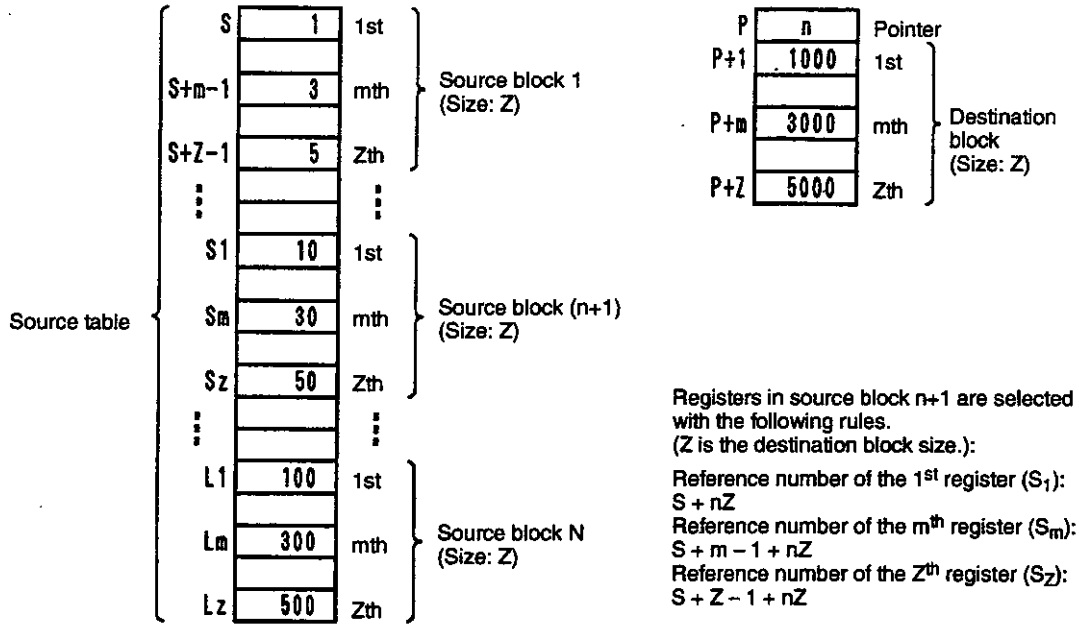
(2) The reference numbers of registers in block n+1 are calculated from the following equations:
Reference number of the 1st register (S_1) = S+nZ
Reference number of the m^{th} register (S_m) = S+m−1+nZ
Reference number of the Z^{th} register (S_Z) = S+Z−1+nZ

4) Valid Pointer Values

If the number of source blocks is N, the valid pointer values (n) are: $0 \leq n \leq N-1$.

3. Operation

1) Before Execution

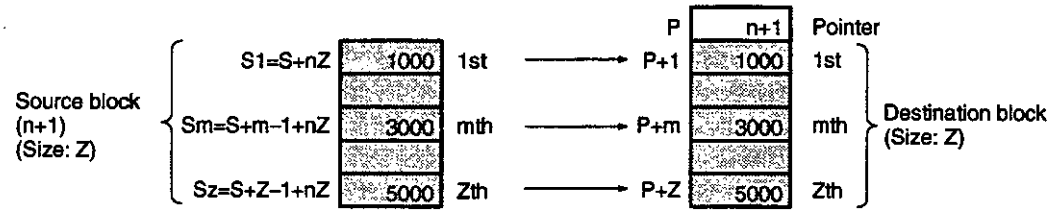


Registers in source block n+1 are selected with the following rules.
(Z is the destination block size.):

Reference number of the 1st register (S_1):
$S + nZ$
Reference number of the m^{th} register (S_m):
$S + m - 1 + nZ$
Reference number of the Z^{th} register (S_Z):
$S + Z - 1 + nZ$

2) Operation with a Valid Pointer Value ($0 \leq n \leq N-1$):

If the pointer value is valid, the following data transfer will be executed when input 1 turns ON. The transfer is completed in one scan.



a) The block of registers S_1 through S_Z is selected as the source block (source block n+1) based on the reference number of the leading register in the source table (S) and the destination block size (Z).

b) All of the data in source block n+1 is copied to the corresponding registers of the destination block.

c) The pointer value is incremented by 1 if input 2 is OFF; it is left unchanged if input 2 is ON.

d) The status of the outputs is as follows:
Output 1: Turns ON.
Output 2: Remains OFF.

3) Operation with an Invalid Pointer Value (n < 0 or n ≥ N):

If the pointer value is invalid, the data transfer won't be executed even when input 1 turns ON. Output 1 will remain OFF and output 2 will be turned ON.

4) When input 3 is turned ON, the pointer value (n) will be reset to zero, regardless of the status of inputs 1 and 2.

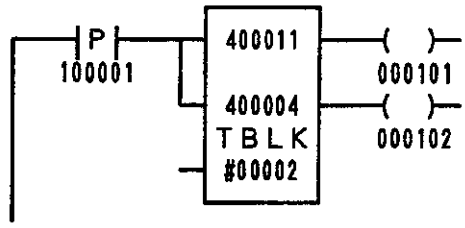
5) The following table shows the operation of the TBLK instruction for all possible input combinations. The pointer value is n and the number of source blocks is N.

Table 3.21 TBLK Operation

Inputs			Condition of n	TBLK Operation	Outputs	
1	2	3			1	2
ON	OFF	OFF	$0 \leq n \leq N-1$	1) All of the data in source block n+1 is copied to the corresponding registers of the destination block. 2) The pointer value (n) is incremented by 1 after the transfer.	**ON**	OFF
			$n < 0, n \geq N$	1) The transfer isn't executed. 2) The pointer value (n) isn't changed.	OFF	**ON**
	ON	OFF	$0 \leq n \leq N-1$	1) The all data in source block n+1 is copied to the corresponding registers of the destination block. 2) The pointer value (n) isn't changed.	**ON**	OFF
			$n < 0, n \geq N$	1) The transfer isn't executed. 2) The pointer value (n) isn't changed.	OFF	**ON**
	OFF	**ON**	None	1) After resetting the pointer value (n) to 0, the data in source block 1 is copied to the corresponding registers of the destination block. 2) The pointer value (n) is incremented to 1 after the transfer.	**ON**	OFF
	ON	**ON**	None	1) After resetting the pointer value (n) to 0, the data in source block 1 is copied to the corresponding registers of the destination block. 2) The pointer value (n) isn't changed. (n=0)		
OFF	Any	**ON**	None	1) The transfer isn't executed. 2) The pointer value (n) is reset to 0.	OFF	OFF
		OFF		1) The transfer isn't executed. 2) The pointer value (n) isn't changed.		

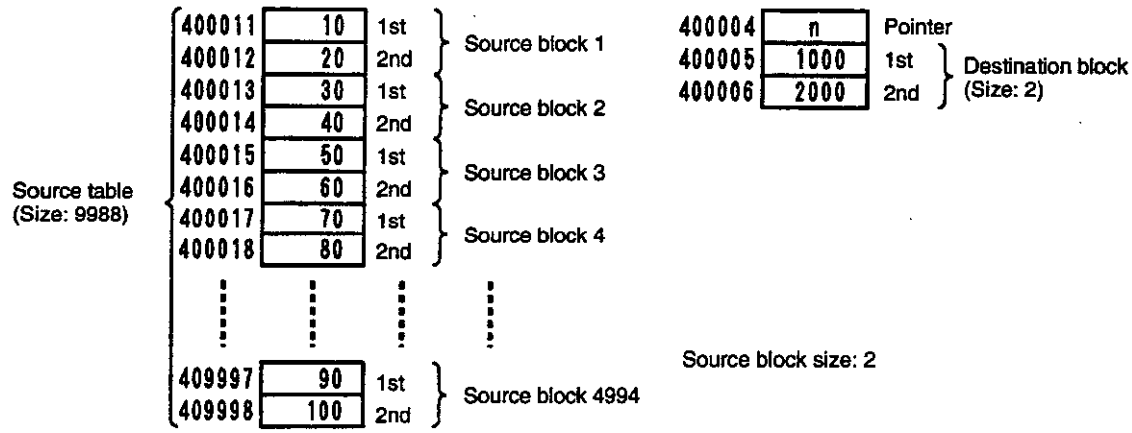
◀EXAMPLE▶ **4. Application Example**

1) Ladder Programming

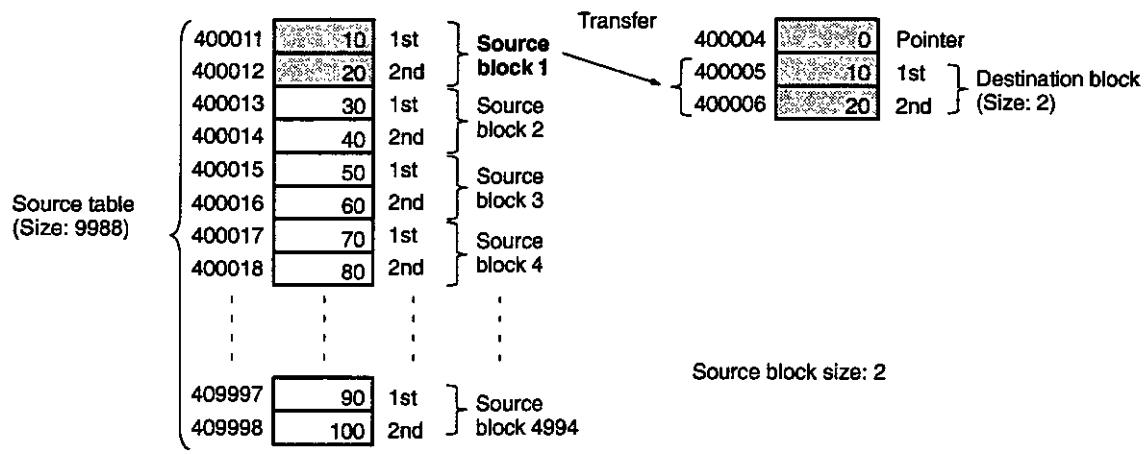


2) Transfer Operation

a) Before Execution



b) The following data transfer is executed when the pointer value (n) is 0 and input relay 100001 changes from OFF to ON. The transfer is completed in one scan.

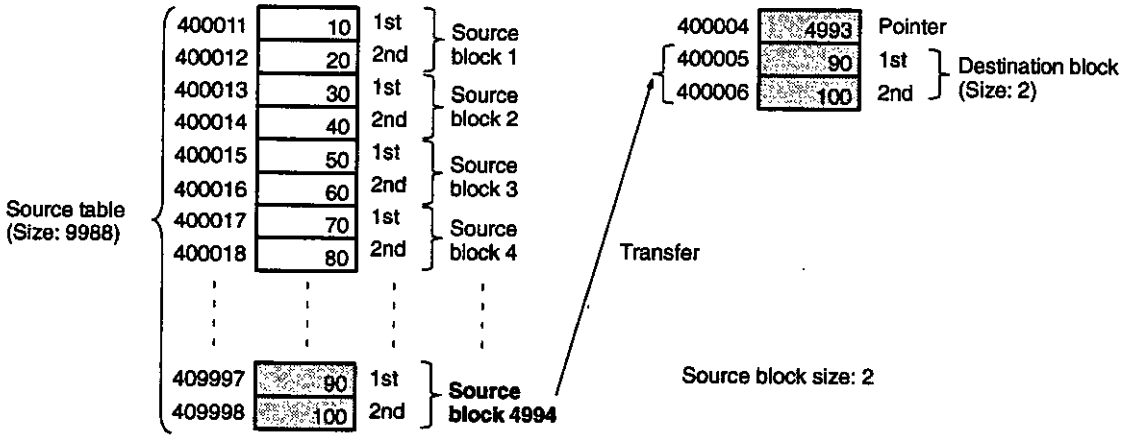


(1) All of the data in source block 1 (400011 and 400012) is copied to the corresponding registers of the destination block.

(2) The pointer value is left unchanged.

(3) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan in which input 100001 changes from OFF to ON.
Coil 000102:
Remains OFF.

c) The following data transfer is executed when the pointer value (n) is 4993 and input relay 100001 changes from OFF to ON. The transfer is completed in one scan.



(1) All of the data in source block 4994 (409997 and 409998) is copied to the corresponding registers of the destination block.

(2) The pointer value is left unchanged.

(3) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan in which input 100001 changes from OFF to ON.
Coil 000102:
Remains OFF.

d) All of the data from any source block in the source table (1 to 4994) can be copied to the destination block by adjusting the pointer value from 0 to 4993.

e) If the pointer value is invalid ($n \leq -1$ or $n \geq 4994$), the data transfer won't be executed even when input relay 100001 turns ON. In this case, the pointer value is left unchanged, coil 000101 is turned OFF, and coil 000102 is turned ON.

3.3.11 INDIRECT BLOCK WRITE (IBKW)

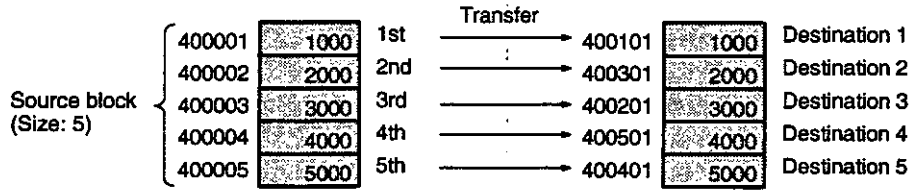
1. Function

1) Data is transferred between a source table and destination registers using a pointer table that is the same size as the source table. The destination registers are determined by the pointer values in the pointer table, so the destination registers do not have to be consecutive or in any particular order.

2) The source table is known as the source block and the data table containing the pointers is known as the pointer block.

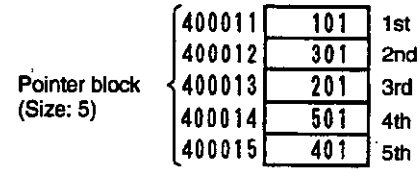
3) The content of each word in the source block can be copied to any destination holding register by adjusting the content of the corresponding registers in the pointer block.

Example
This example shows the operation of the INDIRECT BLOCK WRITE instruction and the functions of the source block and pointer block. The data in the source block is copied to the 5 destination registers shown in the following diagram if IBKW is executed with the shown pointer values. The data can be copied to other registers by changing the pointer values.

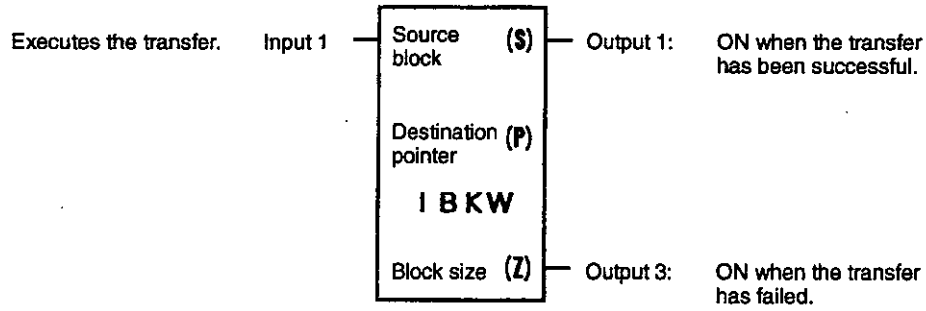


- The data in the n^{th} register of the source block is copied to destination n.

- 400000 is added to the content of the n^{th} pointer to calculate the reference number of destination n, a holding register.
  For example, 400000 is added to the content of the 1^{st} pointer (101) to calculate the reference number of destination 1 (400101).



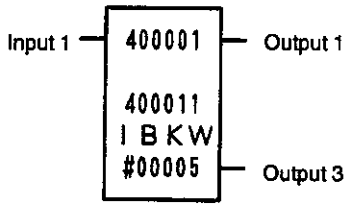
2. Structure



1) IBKW is the symbol for INDIRECT BLOCK WRITE.

2) IBKW requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 3.22* lists the register reference numbers and constants that can be specified.

Example



400001: Reference number of the leading register in the source block

400011: Reference number of the leading register in the pointer block

#00005: Size of the source and pointer blocks (5)

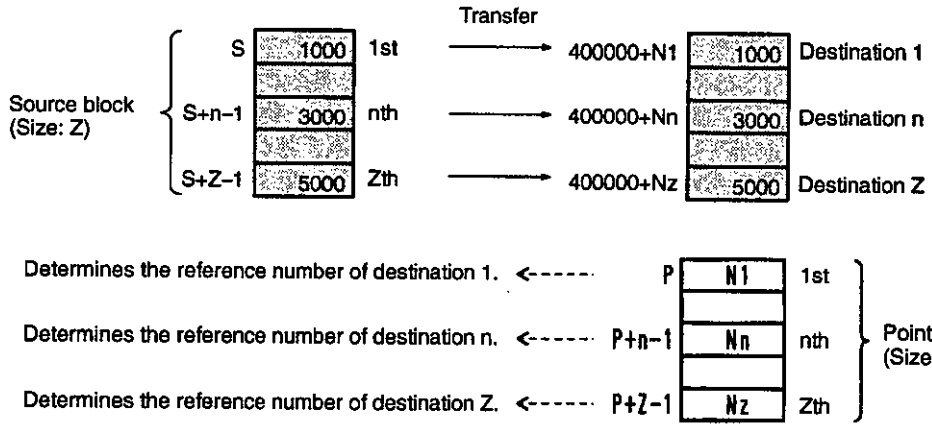
Table 3.22 Structural Elements of IBKW

Element	Meaning	Possible Settings
Top (S)	Reference number of the leading register in the source block	Holding register: 400001 to 409999 (W00001 to W09999) Constant register: 700001 to 704096 (K00001 to K04096) Link register: R10001 to R11024 R20001 to R21024
Middle (P)	Reference number of the leading register in the pointer block	Holding register: 400001 to 409999 (W00001 to W09999) Link register: R10001 to R11024 R20001 to R21024
Bottom (Z)	Size of the source and pointer blocks	Constant: #00001 to #00255

3) The Destination Registers

a) The destination registers are holding registers which are specified by the content of the registers in the pointer block. Any holding register can be selected as the destination for a register in the source block by changing the content of the corresponding register in the pointer block.

b) The content of the n^{th} register of the pointer block (Nn) determines the destination (Dn) for the n^{th} register of the source block. The following equation shows the relationship between Dn and Nn: Dn = 400000 + Nn.



4) Valid Pointer Values

a) Pointer values (Nn) must meet the following three conditions. The content of a register in the pointer block is a "valid pointer value" if it meets these three conditions.

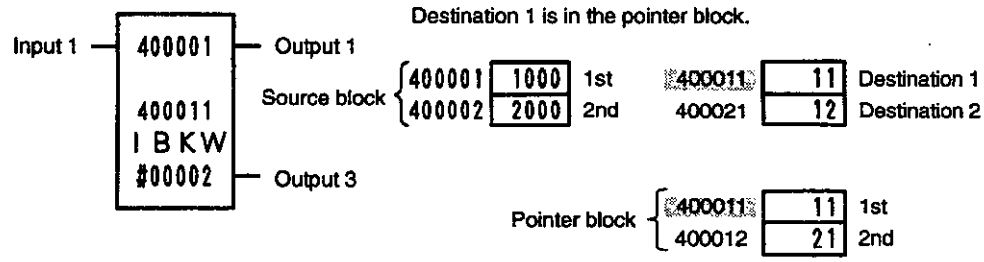
Condition 1: $1 \leq Nn \leq 9999$

Condition 2: The destination register specified by Nn isn't in the pointer block.

Condition 3: The destination register specified by Nn isn't in the source block.

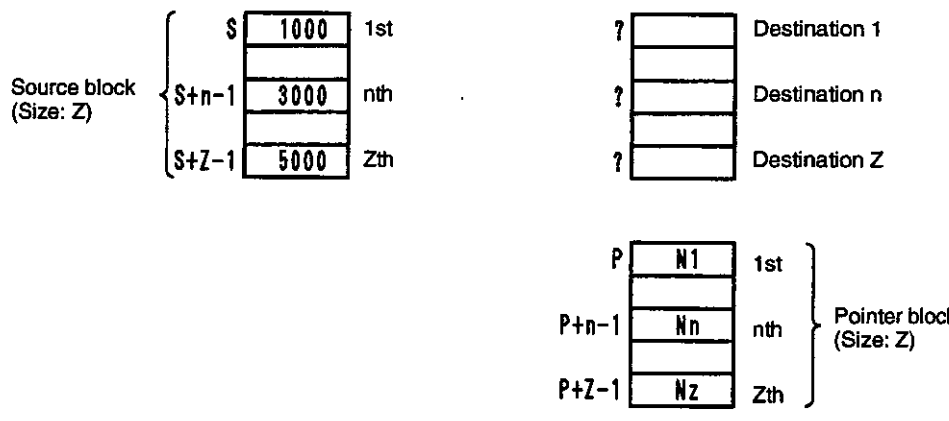
b) Condition 2 is not met in the following example. In this case, the data transfer wouldn't be executed even when input 1 is turned ON. Output 1 would be turned OFF and output 2 would be turned ON.

Example



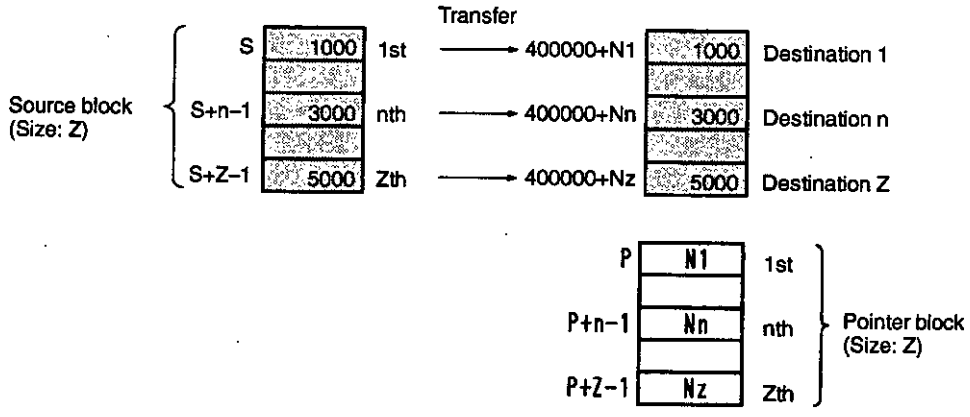
3. Operation

1) Before Execution



2) **All Pointers (N1 to NZ) Valid:**
If all of the registers in the pointer block contain valid pointer values, the following data

transfer is executed when input relay 100001 changes from OFF to ON. The transfer is completed in one scan.



a) The holding registers shown in the diagram are selected as destination registers 1 to Z based on the contents of the pointer block (N1 to NZ).

b) The contents of the registers in the source block are copied to the selected holding registers.

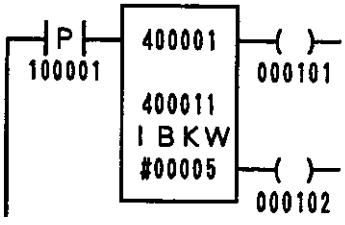
c) The status of the outputs is as follows:
Output 1: Turns ON.
Output 3: Remains OFF.

3) **Any Pointer (N1 to NZ) Invalid:**
If any of the registers in the pointer block contains an invalid pointer value, the data transfer isn't executed, output 1 remains OFF, and output 1 is turned ON.

◀EXAMPLE▶

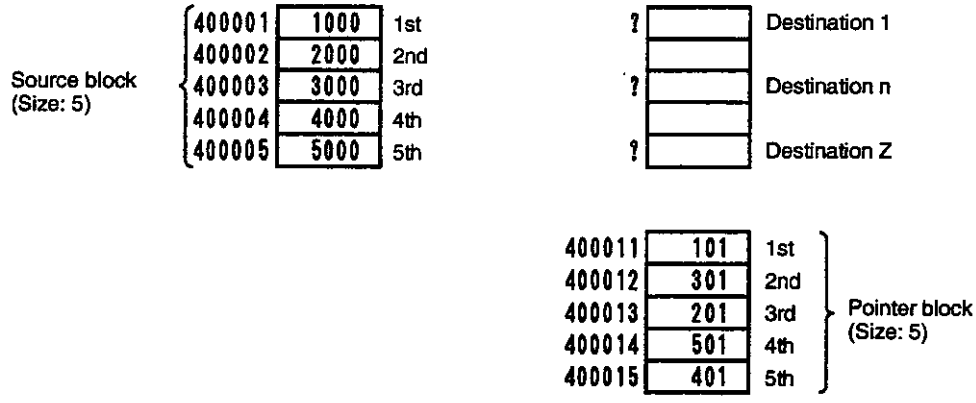
4. Application Example

1) Ladder Programming

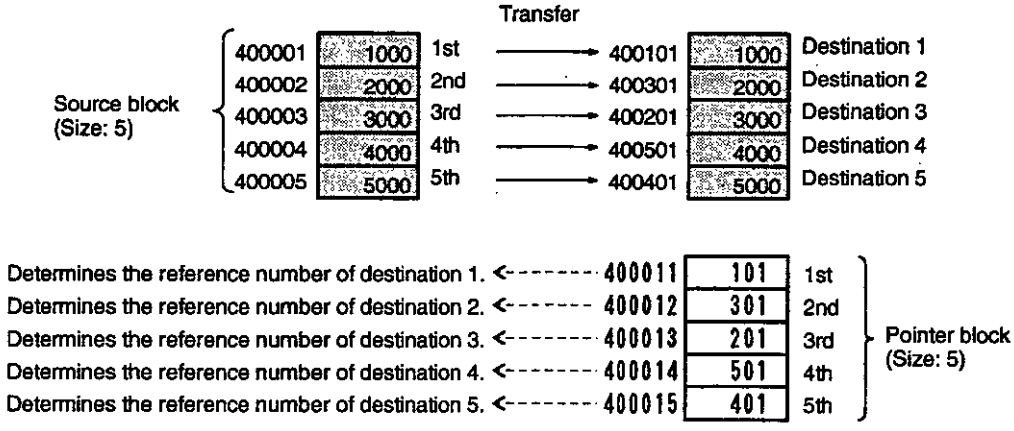


2) Transfer Operation

a) Before Execution



b) The following data transfer is executed when input relay 100001 changes from OFF to ON. The transfer is completed in one scan.

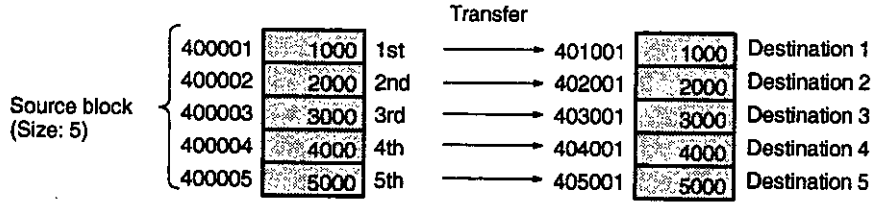


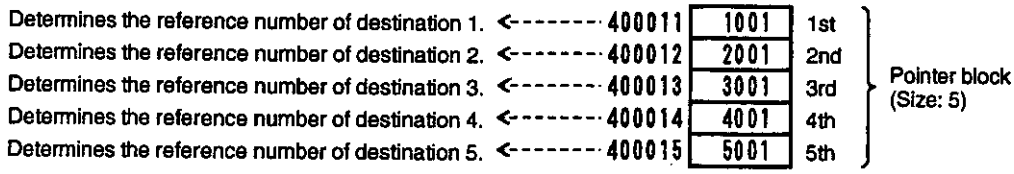
(1) The holding registers shown in the diagram are selected as destination registers 1 to 5 based on the contents of the pointer block.

(2) The contents of the registers in the source block are copied to the selected holding registers.

(3) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan in which input 100001 changes from OFF to ON.
Coil 000102:
Remains OFF.

c) The following data transfer is executed when the contents of the pointer block are changed as shown and input relay 100001 changes from OFF to ON. The transfer is completed in one scan.



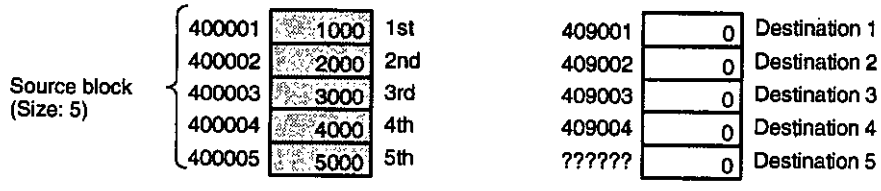


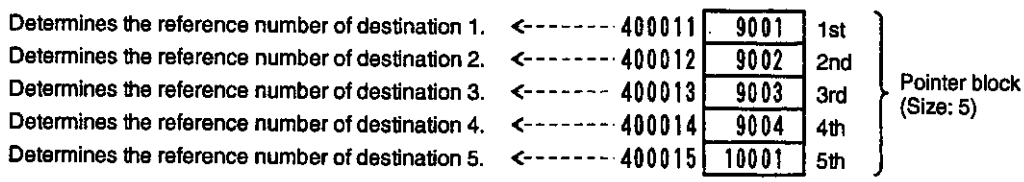
(1) The holding registers shown in the diagram are selected as destination registers 1 to 5 based on the contents of the pointer block.

(2) The contents of the registers in the source block are copied to the selected holding registers.

(3) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan in which input 100001 changes from OFF to ON.
Coil 000102:
Remains OFF.

d) Since the 5th register in the pointer block contains an invalid pointer value, the transfer won't be executed when input relay 100001 changes from OFF to ON. Coil 000101 remains OFF and coil 000102 turns ON only in the scan in which input 100001 changes from OFF to ON.





3.3.12 INDIRECT BLOCK READ (IBKR)

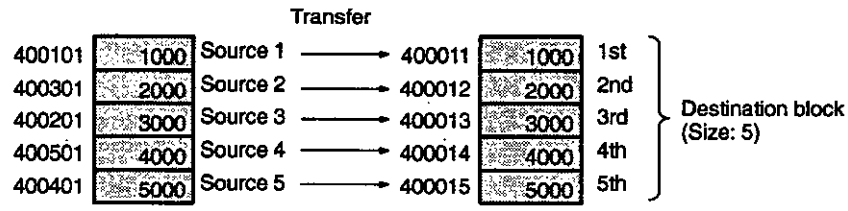
1. Function

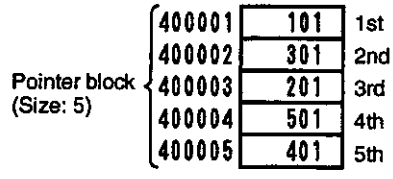
1) Data is transferred between a source table and destination registers using a pointer table that is the same size as the destination table. The source registers are determined by the pointers values in the pointer table, so the source registers do not have to be consecutive or in any particular order.

2) The data table containing the pointers is known as the pointer block and the data table containing the destination registers is known as the destination block.

3) The content of any holding register can be copied to a destination register by adjusting the content of the corresponding register in the pointer block. The transfer is completed in one scan.

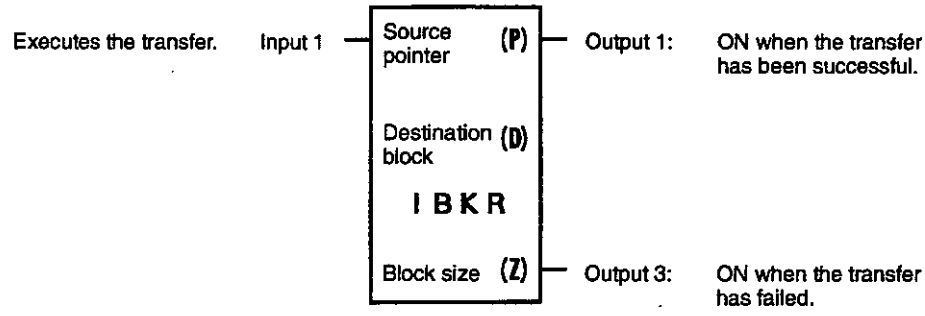
Example
This example shows the operation of the INDIRECT BLOCK READ instruction and the functions of the source block and pointer block. The data from source registers 1 to 5 is copied to the destination block as shown in the following diagram if IBKR is executed with the pointer values shown. Data from any holding registers can be copied to the destination block by changing the pointer values.





- The data in source n is copied to the n^{th} register of the destination block.
- 400000 is added to the content of the n^{th} pointer to calculate the reference number of source n.
  For example, 400000 is added to the content of the 1^{st} pointer (101) to calculate the reference number of source 1 (400101).

2. Structure



1) IBKR is the symbol for INDIRECT BLOCK READ.

2) IBKR requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 3.23* lists the register reference numbers and constants that can be specified.

Example

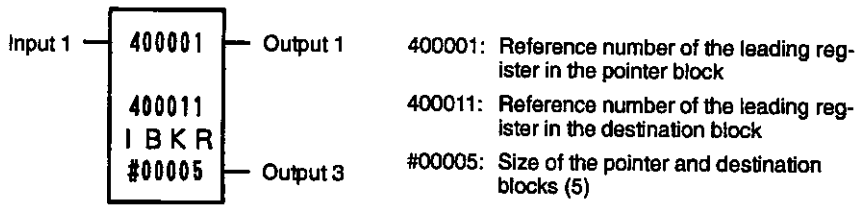


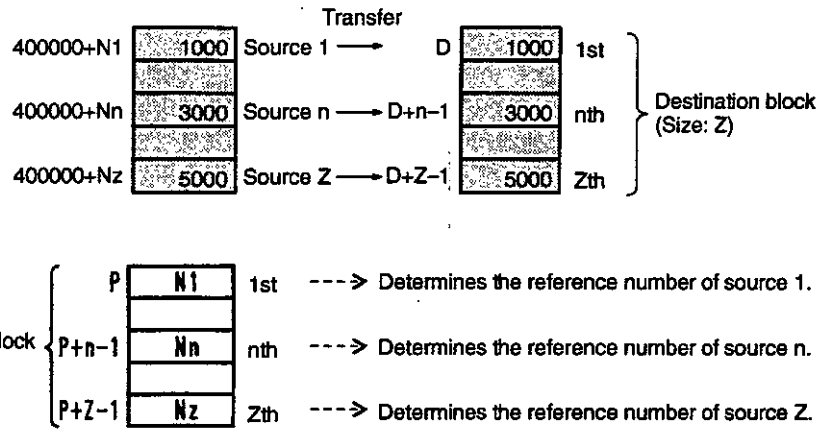
Table 3.23 Structural Elements of IBKR

Element	Meaning	Possible Settings	
Top (P)	Reference number of the leading register in the pointer block	Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 R20001 to R21024
Middle (D)	Reference number of the leading register in the destination block	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 R20001 to R21024
Bottom (Z)	Size of the pointer and destination blocks	Constant:	#00001 to #00255

3) The Source Registers

a) The source registers are holding registers which are specified by the content of the registers in the pointer block. Any holding register can be selected as the source for a particular register in the destination block by changing the content of the corresponding register in the pointer block.

b) The content of the n^{th} register of the pointer block (Nn) determines the source (Sn) for the n^{th} register of the destination block. The following equation shows the relationship between Sn and Nn: Sn = 400000 + Nn.



4) Valid Pointer Values

a) Pointer values must meet the following three conditions. The content of a register in the pointer block is a "valid pointer value" if it meets these three conditions.

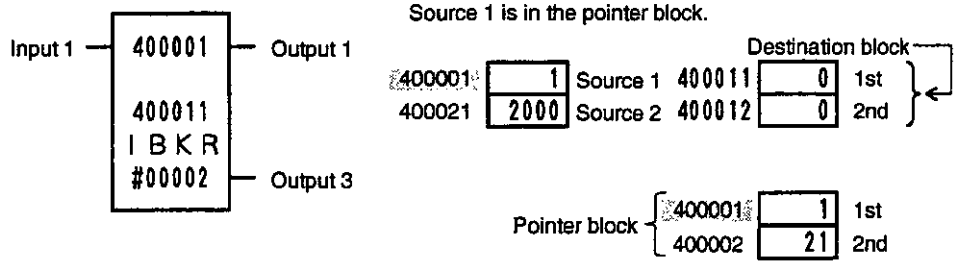
Condition 1: . . $1 \leq Nn \leq 9999$

Condition 2: . . The source register specified by Nn isn't in the pointer block.

Condition 3: . . The source register specified by Nn isn't in the destination block.

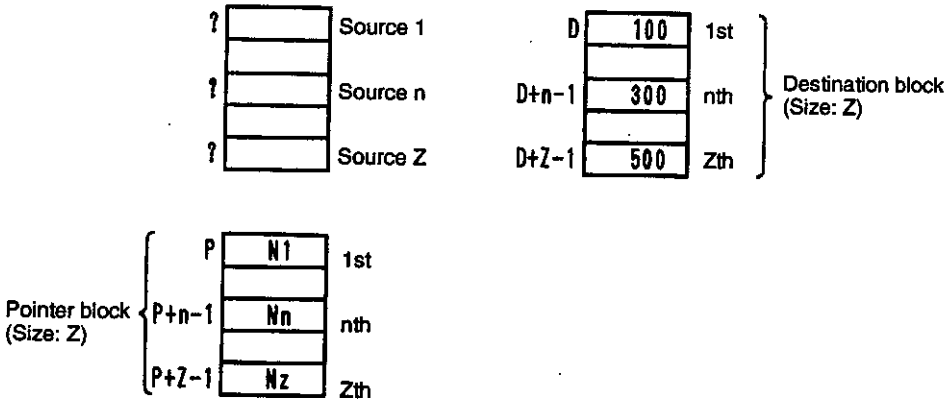
b) Condition 2 is not met in the following example. In this case, the data transfer wouldn't be executed even when input 1 is turned ON. Output 1 would be turned OFF and output 3 would be turned ON.

Example



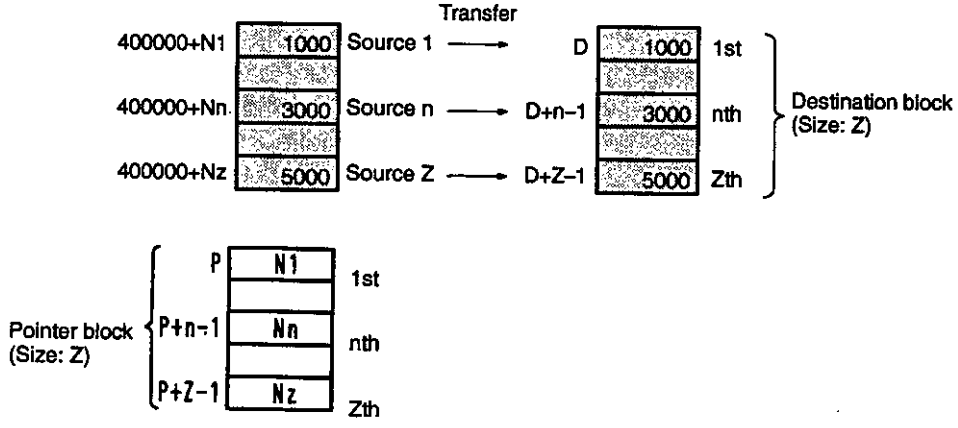
3. Operation

1) Before Execution of the Instruction



2) All Pointers (N1 to NZ) Valid:

If all of the registers in the pointer block contain valid pointer values, the following data transfer is executed when input relay 100001 changes from OFF to ON. The transfer is completed in one scan.



a) The holding registers shown in the diagram are selected as source registers 1 to Z based on the contents of the pointer block (N1 to NZ).

b) The contents of the selected source registers are copied to the corresponding registers in the destination block.

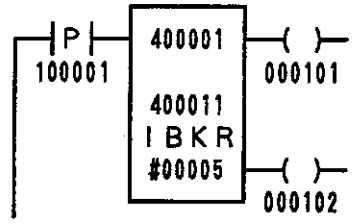
c) The status of the outputs is as follows:
Output 1: Turns ON.
Output 3: Remains OFF.

3) Any Pointer (N1 to NZ) Invalid:

If any of the registers in the pointer block contains an invalid pointer value, the data transfer isn't executed, output 1 remains OFF, and output 1 is turned ON.

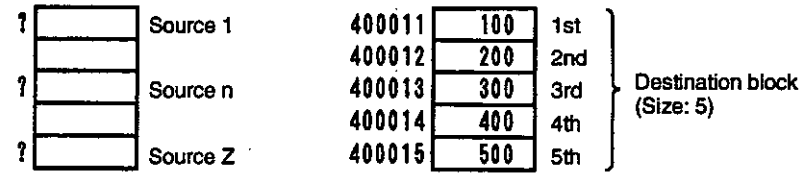
◀EXAMPLE▶ **4. Application Example**

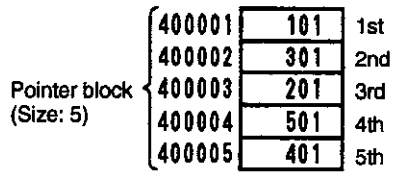
1) Ladder Programming



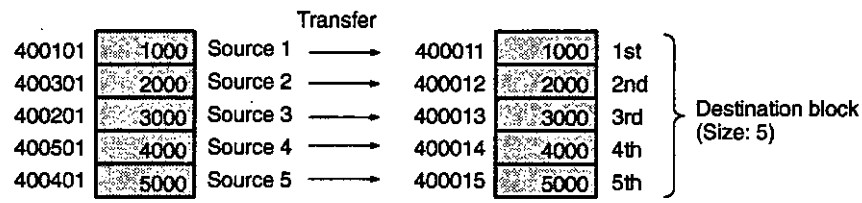
2) Transfer Operation

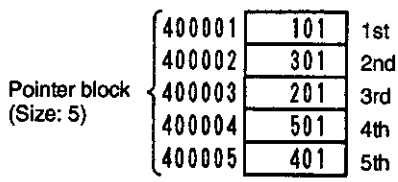
a) Before Execution of the Instruction





b) The following data transfer is executed when input relay 100001 changes from OFF to ON. The transfer is completed in one scan.



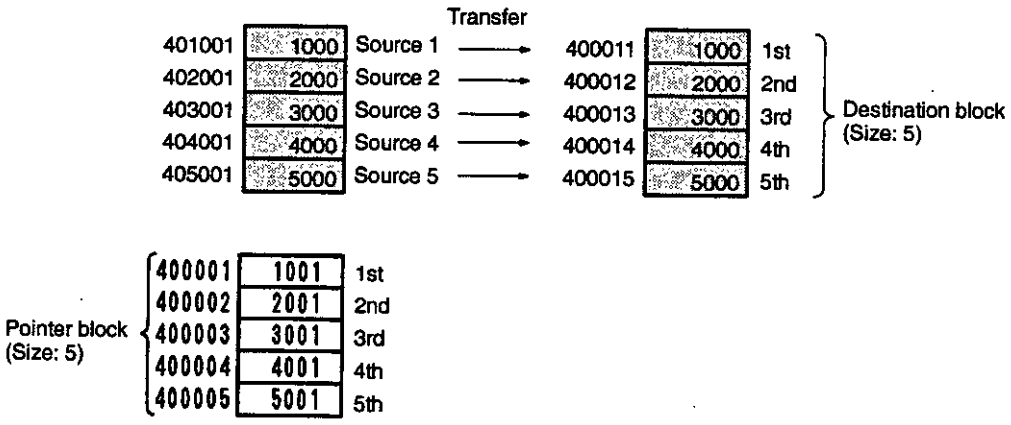


(1) The holding registers shown in the diagram are selected as source registers 1 to 5 based on the contents of the pointer block.

(2) The contents of the selected source registers are copied to the corresponding registers in the destination block.

(3) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan in which input 100001 changes from OFF to ON.
Coil 000102:
Remains OFF.

c) The following data transfer is executed when the contents of the pointer block are changed as shown and input relay 100001 changes from OFF to ON. The transfer is completed in one scan.



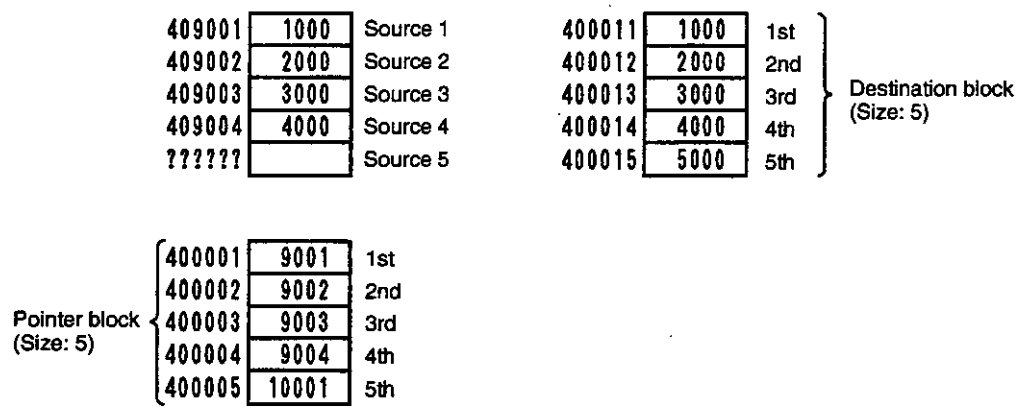
(1) The holding registers shown in the diagram are selected as source registers 1 to 5 based on the contents of the pointer block.

(2) The contents of the selected source registers are copied to the corresponding registers in the destination block.

(3) The status of the outputs is as follows:
Coil 000101:
Turns ON only in scan in which input 100001 changes from OFF to ON.
Coil 000102:
Remains OFF.

d) Since the 5th register in the pointer block contains an invalid pointer value, the transfer won't be executed when input relay 100001 changes from OFF to ON. Coil 000101

remains OFF and coil 000102 turns ON only in the scan in which input 100001 changes from OFF to ON.



3.4 Building Programs

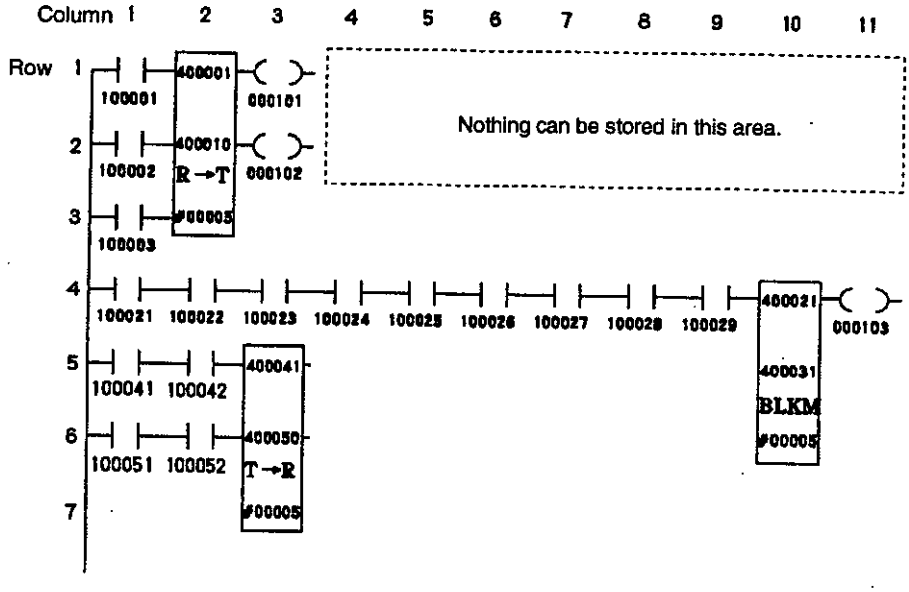
This section describes precautions that should be taken when designing programs that contain data transfer instructions.

3.4.1 Storage Locations on Networks

All data transfer instructions require three elements (top, middle, and bottom) located vertically on the network, so they can be stored anywhere on a 5-row by 10-column matrix (rows 1 through 5 and columns 1 through 10).

Note Data transfer instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

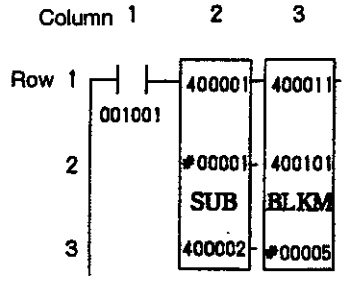
Example



3.4.2 Inputs

Inputs to data transfer instruction can be connected to relay elements (except coils) and outputs from timers, counters, math instructions, data transfer instructions, other instructions, etc.

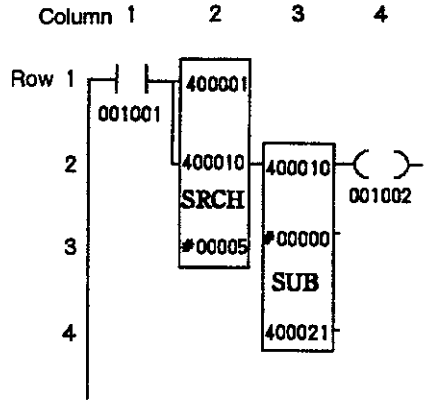
Example



3.4.3 Outputs

Outputs from data transfer instruction can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data transfer instructions, etc.

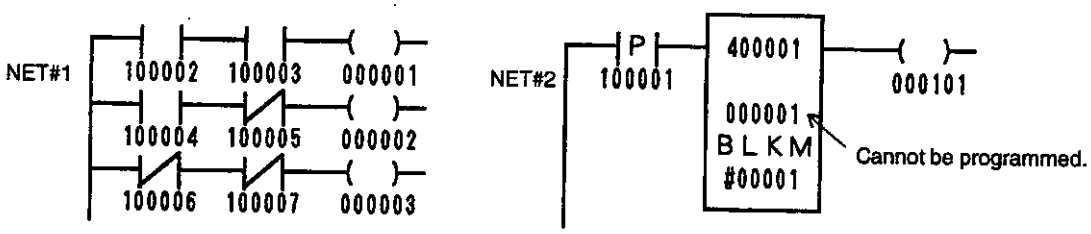
Example



3.4.4 Duplicate Coil Usage

1) A coil table that includes coils that have already been used cannot be used as a destination.

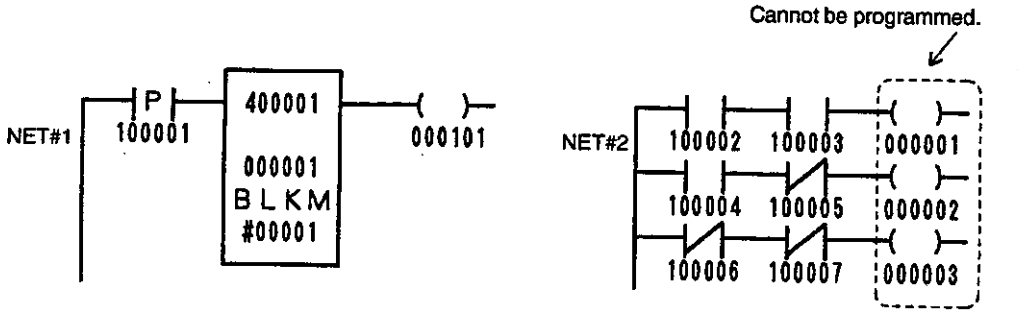
Example: Incorrect Application



Coils 000001 to 000003 are used in network #1, so coil 000001 cannot be used as the reference for a destination, such as the one shown above in network #2.

2) The coils in coil tables used as destinations cannot be used again as coils. The coils on the right in the following diagram cannot be used because they have already been used as a destination.

Example: Incorrect Application

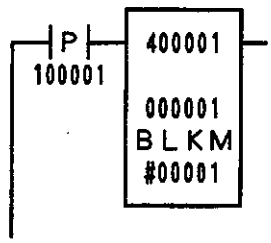


Coils 000001 to 000016 are used in network #1, so coils 000001 to 000003 cannot be used as the references in network #2.

3.4.5 Operation of Disabled Coils

Do not execute data transfer instructions containing disabled coils (including output coils, internal coils, link coils, MC coils, and MC control coils) as destinations. If disabled coils are used as destinations, their status will be overwritten by the data transfer instruction, as shown in the following example. The disabled coils, however, will not be enabled.

Example



1) Assume the following status for holding register 400001 and coils 000001 to 000016:

400001	1111	1111	1111	1111

Coils 000001 to 000016: All disabled OFF

2) When input relay 100001 changes from OFF to ON, all coils from 000001 to 000016 will be disabled ON.

Indexed Block Transfer Instructions

4

This chapter describes indexed block transfer instructions.

4.1 Indexed Block Transfer Instructions

The indexed block transfer instructions transfer the content of a source block in one scan in a similar manner to the BLOCK MOVE (BLKM) instruction (see *3.3.8 BLOCK MOVE (BLKM)*) except for the following:

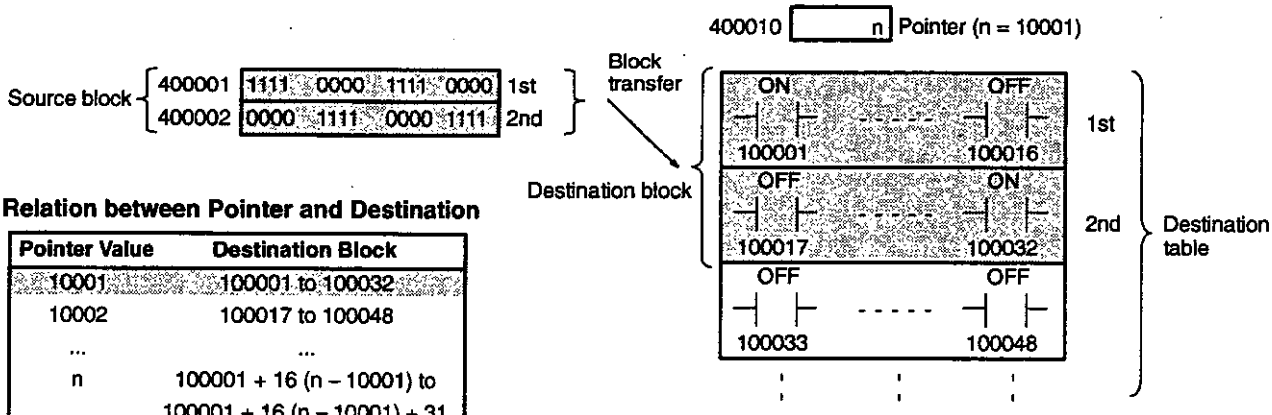
1. The source block and the destination block can be specified by the pointer value.
2. Input relay blocks can be used as destination blocks.

- The four indexed block transfer instructions are described in the following table.

Table 4.1 Indexed Block Transfer Instructions

Name	Symbol	Function	Page
DESTINATION INDEXED BLOCK TRANSFER 1	DIBT	The contents of the source block is copied to the destination block (input relay table) specified by the pointer value. The transfer is completed in one scan.	4-6

Example

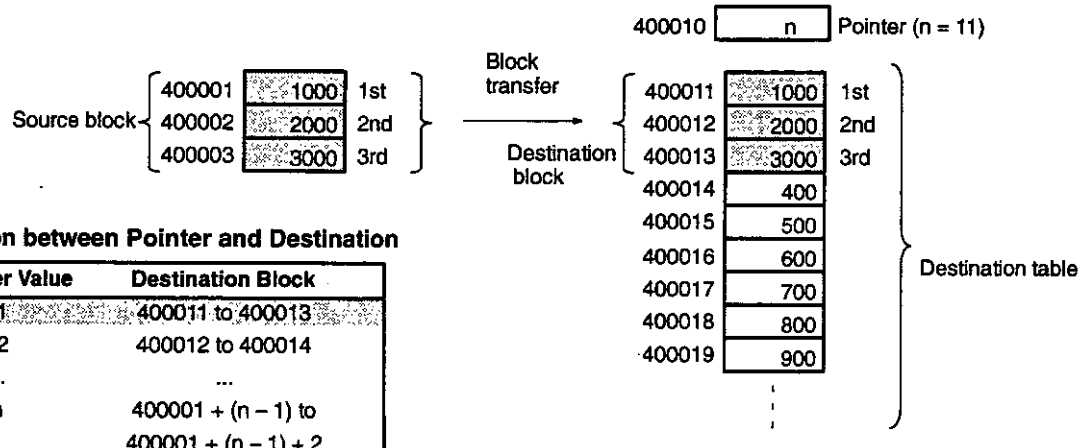


Relation between Pointer and Destination

Pointer Value	Destination Block
10001	100001 to 100032
10002	100017 to 100048
...	...
n	100001 + 16 (n – 10001) to 100001 + 16 (n – 10001) + 31

DESTINATION INDEXED BLOCK TRANSFER 2	DIBR	The contents of the source block (holding register table) is copied to the destination block specified by the pointer value (holding register table). The transfer is completed in one scan.	4-15

Example

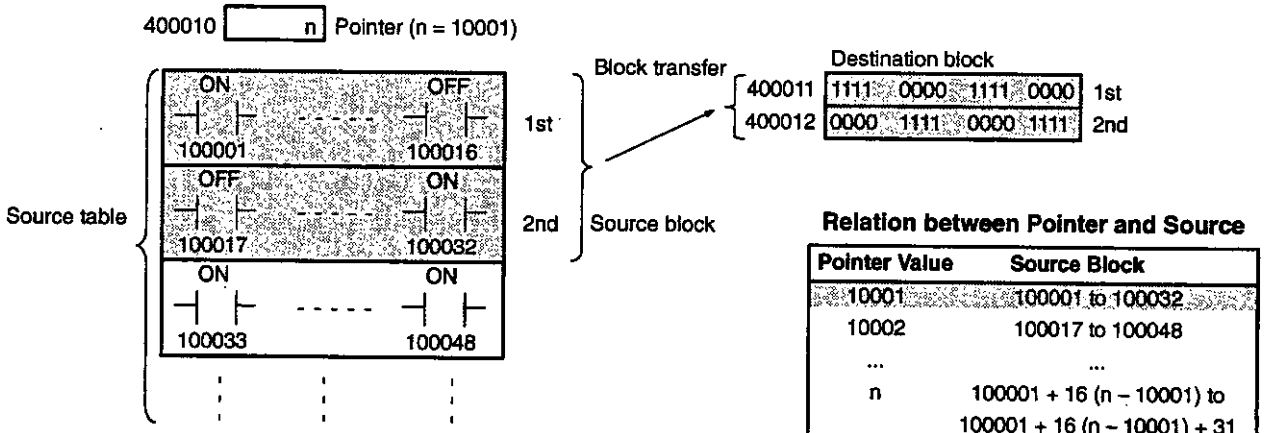


Relation between Pointer and Destination

Pointer Value	Destination Block
11	400011 to 400013
12	400012 to 400014
...	...
n	400001 + (n – 1) to 400001 + (n – 1) + 2

Name	Symbol	Function	Page
SOURCE INDEXED BLOCK TRANSFER 1	SIBT	The contents of the source block specified by the pointer value is copied to the destination block (holding register table) The transfer is completed in one scan.	4-22

Example

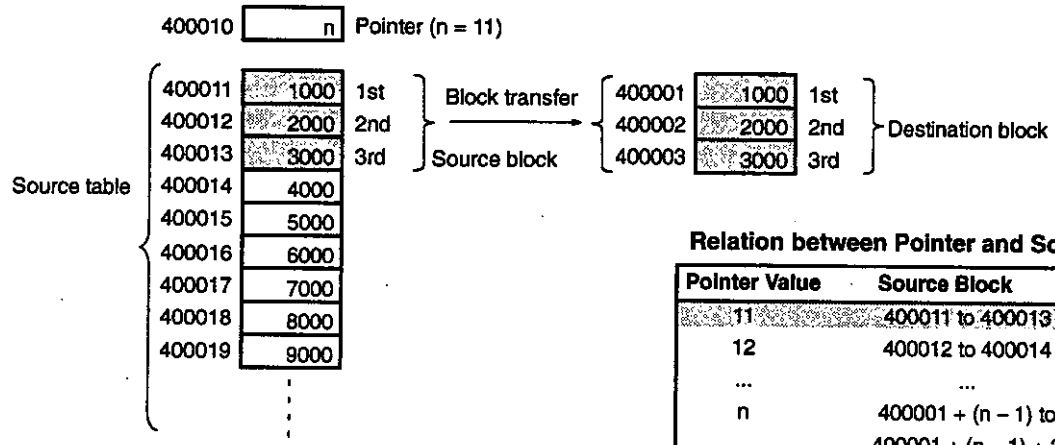


Relation between Pointer and Source

Pointer Value	Source Block
10001	100001 to 100032
10002	100017 to 100048
...	...
n	100001 + 16 (n – 10001) to 100001 + 16 (n – 10001) + 31

SOURCE INDEXED BLOCK TRANSFER 2	SIBR	The contents of the source block (holding register table) specified by the pointer value is copied to the destination block (holding register table). The transfer is completed in one scan.	4-38

Example



Relation between Pointer and Source

Pointer Value	Source Block
11	400011 to 400013
12	400012 to 400014
...	...
n	400001 + (n – 1) to 400001 + (n – 1) + 2

4.2 Indexed Block Transfer Instruction Terminology

This section explains the terms required to understand the operation of the indexed block transfer instructions.

4.2.1 Data Tables and Table Size

The following terms have the same meaning as for data transfer instructions. Refer to *3.2 Data Transfer Instruction Terminology* for definitions of these terms.

- Data table
- Register table
- Coil table
- Relay table
- Table size

4.2.2 Source and Destination

1) These terms also have the same meaning as for data transfer instructions. The origin of the data is called the source, and the data is transferred to a destination.

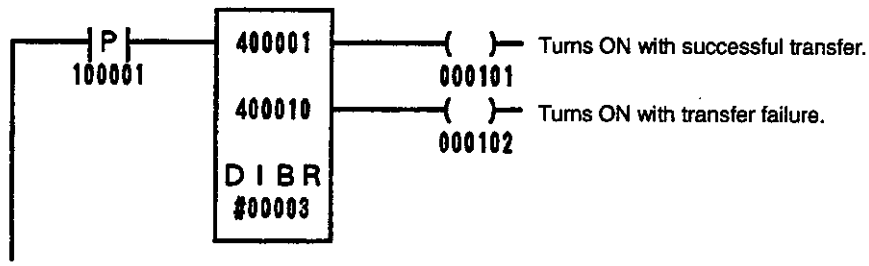
2) As shown in the following example, the source and destination data tables can differ in size for indexed block transfer instructions. Data is transferred, however, between blocks of the tables that are the same size. In such cases, the source data table is called the source block, and the destination data table is called the destination block.

4.2.3 Pointers

As shown below, a pointer in indexed block transfer instructions is used to specify data blocks in a source or destination tables.

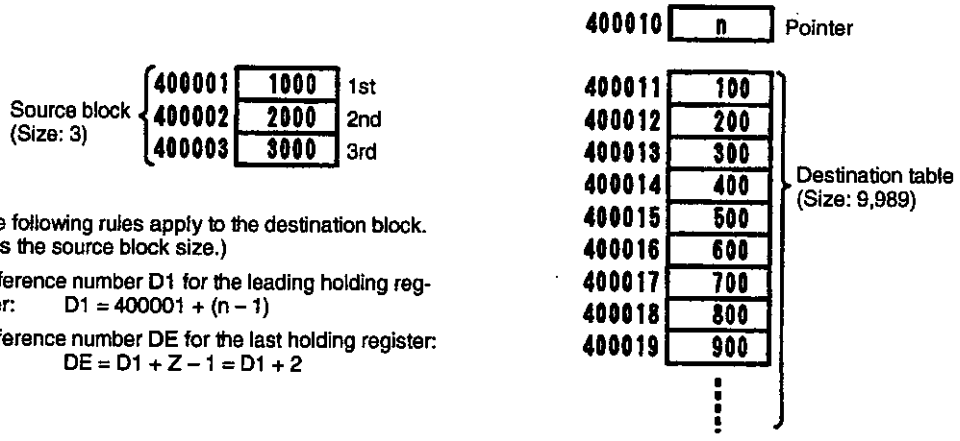
Example
Data transfer using DESTINATION INDEXED BLOCK TRANSFER INSTRUCTION 2

1) Ladder Programming



2) Transfer Contents

a) Status Before Execution

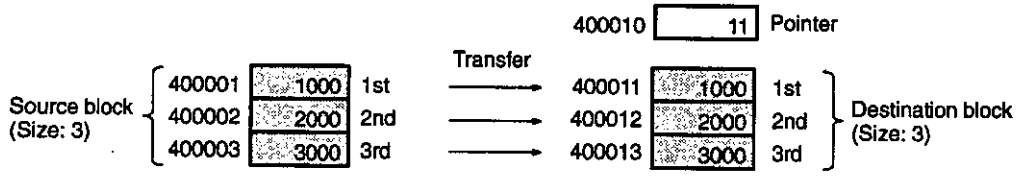


The following rules apply to the destination block.
(Z is the source block size.)

Reference number D1 for the leading holding register: D1 = 400001 + (n – 1)

Reference number DE for the last holding register:
DE = D1 + Z – 1 = D1 + 2

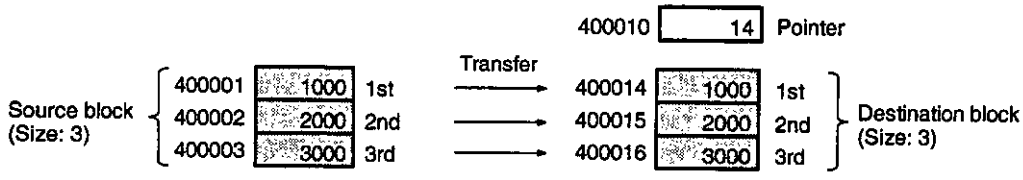
b) The following data transfer is executed when the pointer value (n) is 11 and input relay 100001 turns from OFF to ON. The transfer is completed in one scan.



(1) A holding register table (400011 to 400013) is selected as the destination block based on the pointer value (11) and the source block size (3).

(2) The contents of the source block is copied to the selected destination block.

c) The following data transfer is executed when the pointer value (n) is 14 and input relay 100001 turns from OFF to ON. The transfer is completed in one scan.



(1) The holding register table (400014 to 400016) is chosen as the destination block based on the pointer value (14) and the size of the source block size (3).

(2) The contents of the source block is copied to the selected destination block.

4.3 Details of Indexed Block Transfer Instructions

This section describes the function, structures and operation of each indexed block transfer instruction and provides simple examples of their application.

4.3.1 DESTINATION INDEXED BLOCK TRANSFER 1 (DIBT)

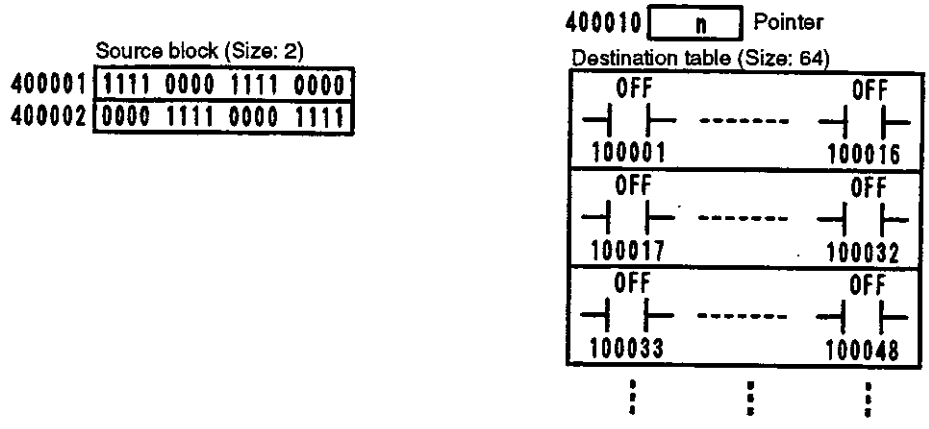
1. Function

1) Data is transferred using indexed blocks between source and destination data tables that differ in size. A pointer is used to determine the destination of the data.

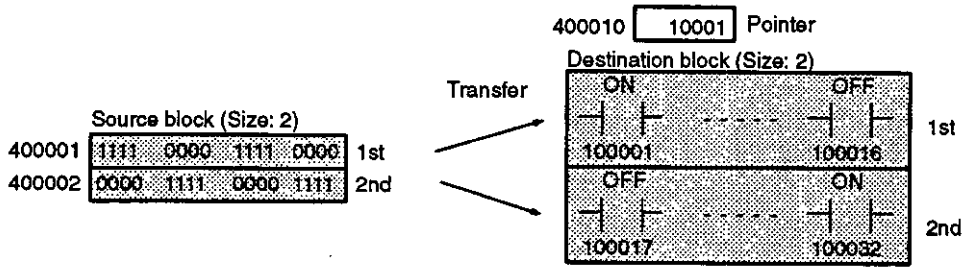
2) The source data table is called a source block. Coil tables, relay tables, and register tables can be used as source blocks. Only input relay tables can be used as the destination tables.

3) The contents of the source block are copied to the block in the destination table specified by the pointer (called the destination block). The transfer is completed in one scan.

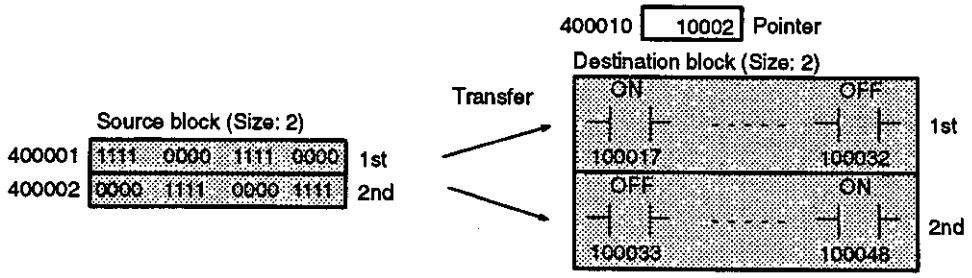
Example
The illustration shows the source block, destination table, and pointer.



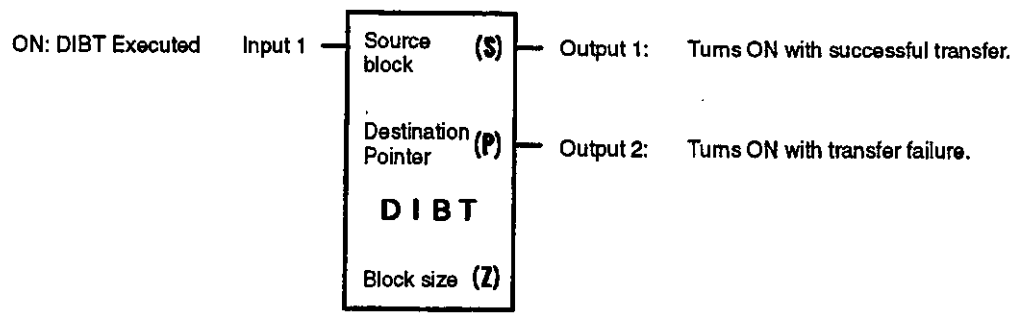
a) When executing the transfer instruction with the pointer value at 10001, the 32 bits of the source block are transferred to the 32 input relays in the destination block (100001 to 100032).



b) When executing the transfer instruction with the pointer value at 10002, the 32 bits of the source block are transferred to the 32 input relays in the destination block (100017 to 100048).



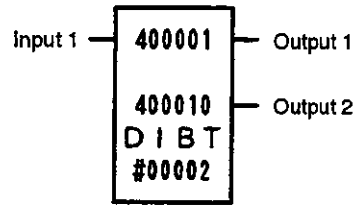
2. Structure



1) DIBT is the symbol for DESTINATION INDEXED BLOCK TRANSFER INSTRUCTION 1.

2) DIBT requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. Refer to *Table 4.2* for details on specifying constants or the reference numbers of coils, relays, or registers for these elements.

Example



400001: Reference number of the leading register in source block

400010: Reference number of pointer

#00002: Size of source block and destination blocks (2)

Table 4.2 DIBT Structural Elements

Element	Meaning	Possible settings	
Top (S)	Reference number of the leading register in the source block	Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (P)	Reference number of the pointer	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 or R20001 to R21024
Bottom (Z)	Size of the source block and destination blocks	Specify the constant. The maximum value of constant differs with specified reference type.	
		Coil:	#00001 to #00100
		Input relay:	#00001 to #00064
		Input register, holding register or constant register:	#00001 to #00100
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00100
		MC coil, MC relay or MC control relay:	#00001 to #00016
		MC control coil:	#00001 to #00010
		M code relay:	#00001 to #00006

Note When a coil or relay is specified, m must equal 16 n + 1 where m is the lower 5 digits of the reference number (and n = 0, 1, 2, etc.).

3) Destination Block

a) The destination block is an input relay table determined by the pointer value and the size of the source block. The following rules apply to input relay tables, where the pointer value is "n" and the size of the source block is "Z".

(1) Reference number for the first input relay D1: D1 = 100001 + 16 (n – 10001)

(2) Reference number for the last input relay DE: DE = D1 + 16Z – 1

b) The following table shows these rules.

Pointer Value	Input Relay Table Used as Destination Block	
	Leading Reference No. (D1)	Last Reference No. (DE)
10001	100001	100001 + 16Z – 1
10002	100017	100017 + 16Z – 1
—	—	—
n	100001 + 16 (n – 10001)	100001 + 16 (n – 10001) + 16Z – 1
—	—	—
10064	101009	101024

4) Effective Range of the Pointer Value

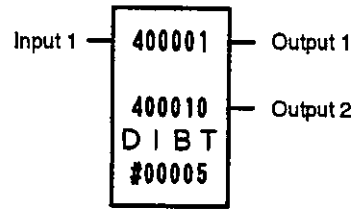
a) The pointer value (n) must satisfy both of the following conditions for the pointer to be defined as being in the effective range.

Condition 1 ..... $10001 \leq n \leq 10064$

Condition 2 ..... The input relay specified by n must actually exist, in other words, must satisfy the following two equations.
$100001 \leq 100001 + 16 (n - 10001) \leq 101009$
$100001 \leq 100001 + 16 (n - 10001) + 16Z - 1 \leq 101024$

b) Neither of the following examples satisfies the above conditions. Accordingly, the data will not be transferred if input 1 of the DIBT is ON. Output 1 will be OFF and output 2 will be ON.

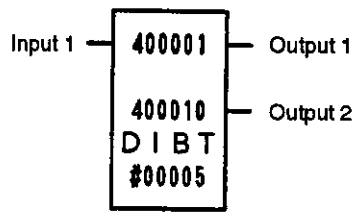
Example 1: Incorrect Application



400010 1000 Pointer

Does not satisfy condition 1.

Example 2: Incorrect Application

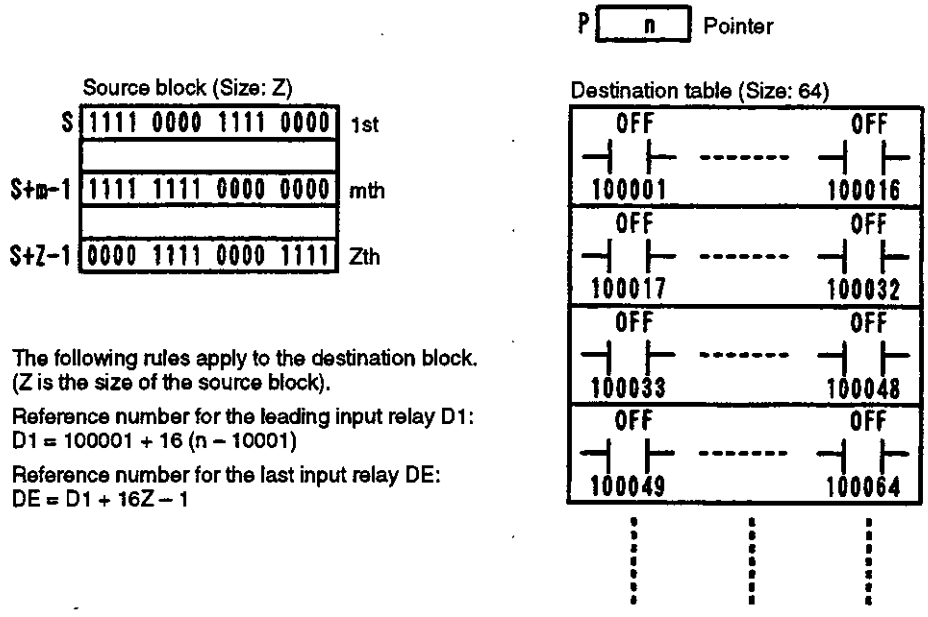


400010 10064 Pointer

Does not satisfy condition 2.

3. Operation

1) Status Before Execution

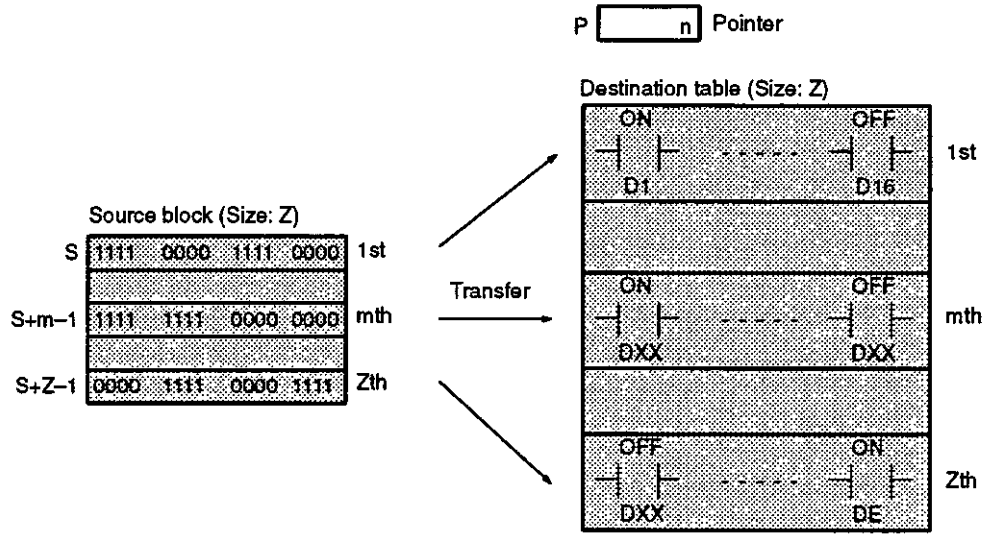


The following rules apply to the destination block. (Z is the size of the source block).

Reference number for the leading input relay D1:
D1 = 100001 + 16 (n – 10001)

Reference number for the last input relay DE:
DE = D1 + 16Z – 1

2) If the pointer value (n) is in the effective range, the following data will be transferred when input 1 turns ON. The transfer is completed in one scan.



a) The input relay table (D1 to DE) will be selected as the destination block in accordance with the rules, and based on the pointer value (n) and the size of the source block (Z).

b) The contents of the source block is copied to the selected destination block.

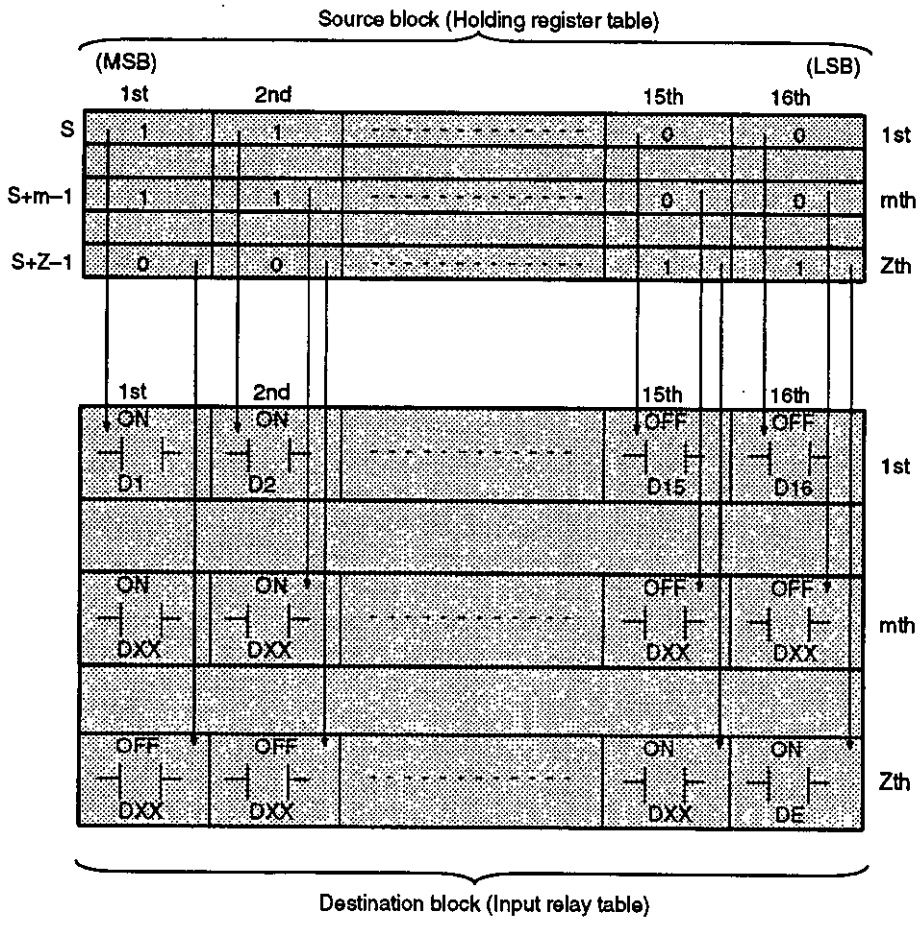
c) The relationship shown in the following illustration exists between the source block bits and the input relays in the destination block.

d) The pointer value and the contents of the source block are left unchanged.

e) Output 1 turns ON. Output 2 remains OFF.

f) Correspondence between Bits and Input Relays

The following illustration shows the correspondence between the bits of the source block and the input relays of the destination blocks.



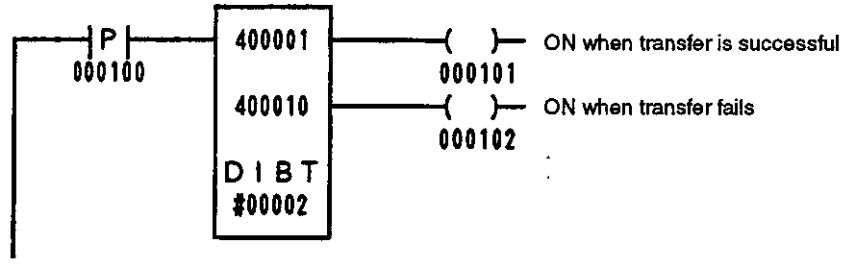
3) When the pointer value (n) is not in the effective range, no destination block exists. Therefore, if input 1 is ON, the following processing is performed.

a) The transfer is not executed.

b) Output 1 remains OFF. Output 2 is ON.

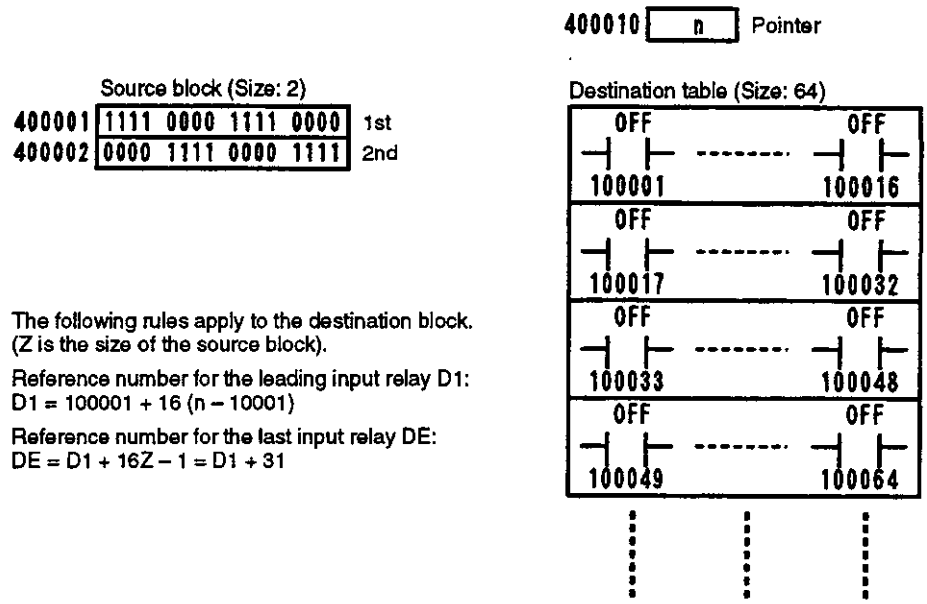
◀EXAMPLE▶ **4. Application Example**

1) Ladder Programming



2) Transfer Contents

a) Before Transfer

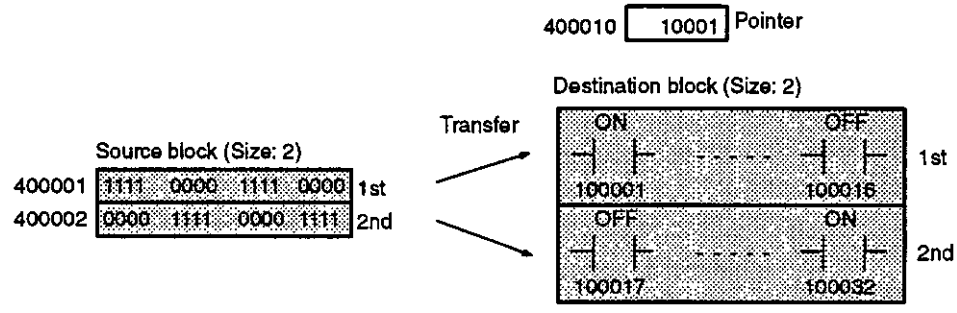


The following rules apply to the destination block. (Z is the size of the source block).

Reference number for the leading input relay D1:
D1 = 100001 + 16 (n – 10001)

Reference number for the last input relay DE:
DE = D1 + 16Z – 1 = D1 + 31

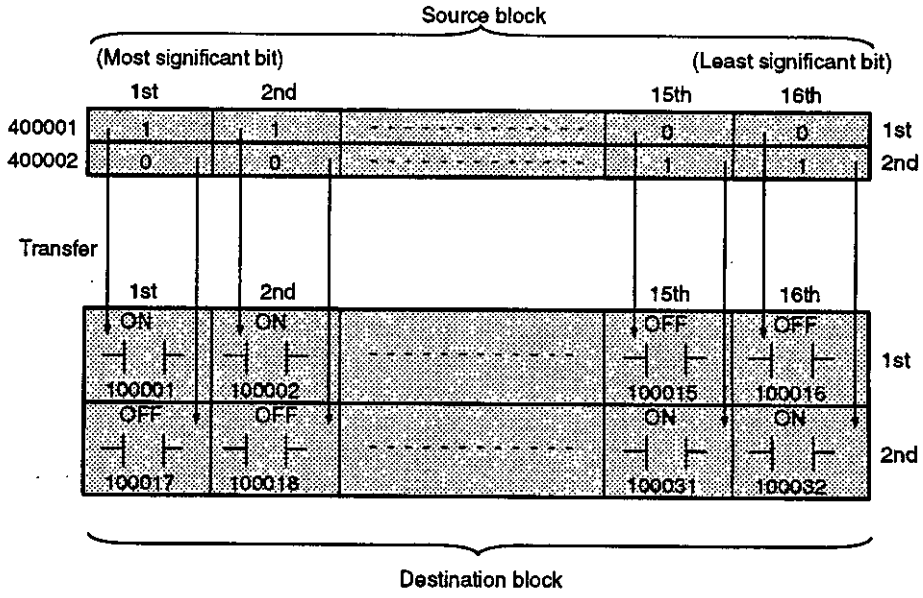
b) The following data transfer is executed when the pointer value (n) is 10001 and coil 000100 changes from OFF to ON. The transfer is completed in one scan.



(1) The input relay table (100001 to 100032) is selected as the destination block based on the pointer value (10001) and the size of the source block (2).

(2) The 32 bits of the source block are transferred to the 32 bits of the input relay in the selected destination block.

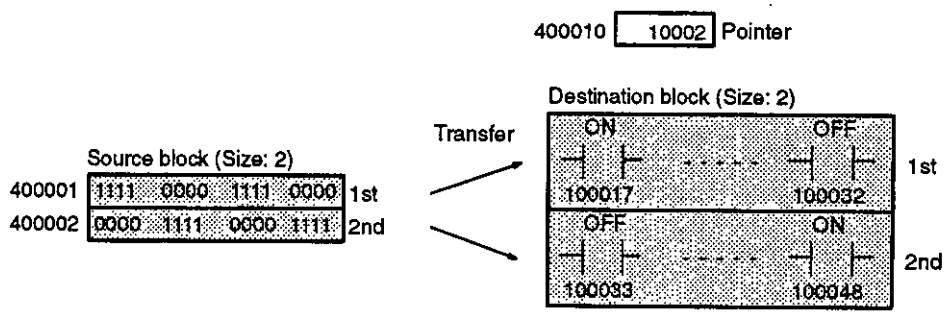
(3) The following illustration shows the relationship that exists between the source block bits and the input relays in the destination block in this instance.



(4) The pointer value and the contents of the source block remain unchanged.

(5) Coil 000101 turns ON only in scans where coil 000100 changes from OFF to ON. Coil 000102 remains OFF.

c) The following data transfer is executed when coil 000100 changes from OFF to ON and the pointer value (n) is 10002. The transfer is completed in one scan.

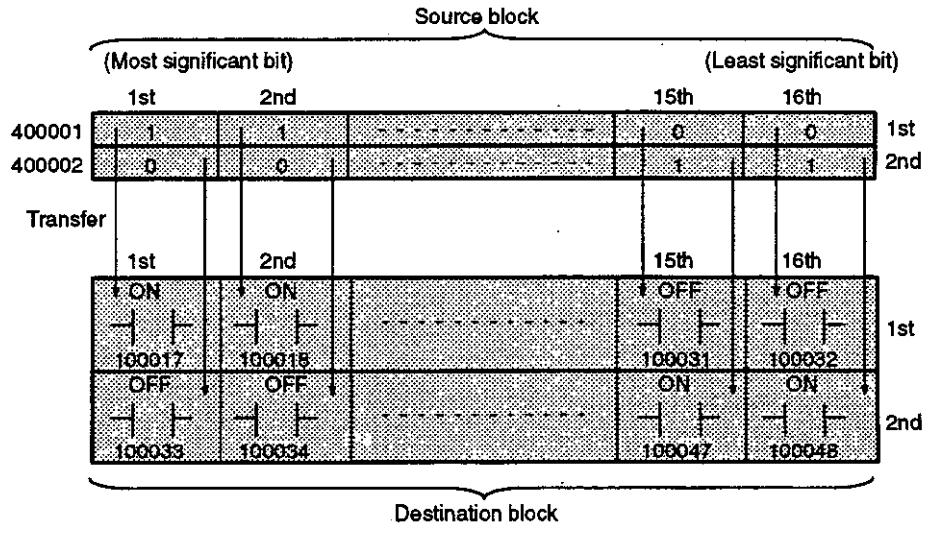


(1) The input relay table (100017 to 100048) is selected as the destination block based on the pointer value (10002) and the size of the source block (2).

(2) The 32 bits of the source block are transferred to the 32 bit input relay in the selected destination block.

(3) The following illustration shows the relationship that exists between the source block bits and the input relays in the destination block in this instance.

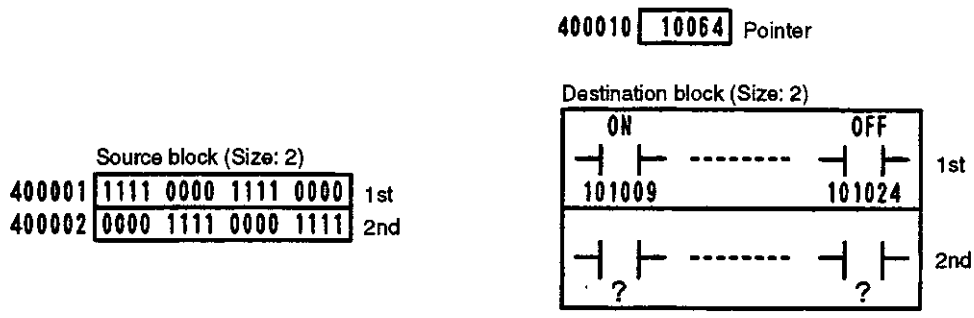
(4) Correspondence between Bits and Input Relays



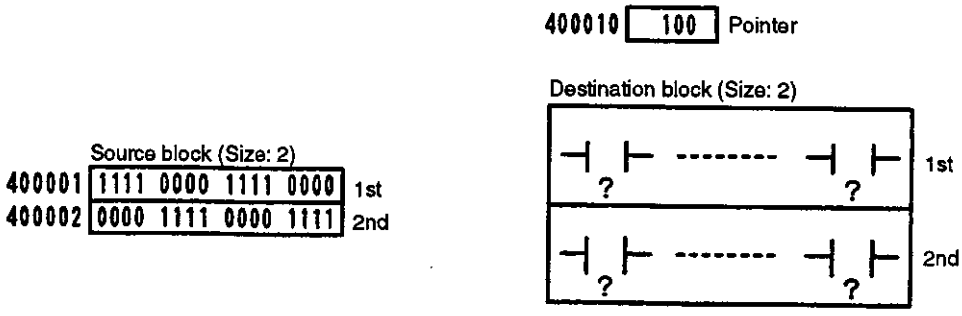
(5) The pointer value and the contents of the source block remain unchanged.

(6) Coil 000101 turns ON only in scans where coil 000100 changes from OFF to ON. Coil 000102 remains OFF.

d) As the illustration shows, no destination block exists when the pointer value (n) is 10064. The transfer of data will not be executed if coil 000100 changes from OFF to ON. Coil 000101 remains OFF. Coil 000102 turns ON only in scans where coil 000100 changes from OFF to ON.



e) As the illustration shows, no destination block exists when the pointer value is 100. Transfer will not be executed even when coil 000100 changes from OFF to ON. Coil 000101 remains OFF. Coil 000102 turns ON only in scans where coil 000100 changes from OFF to ON.



4.3.2 DESTINATION INDEXED BLOCK TRANSFER 2 (DIBR)

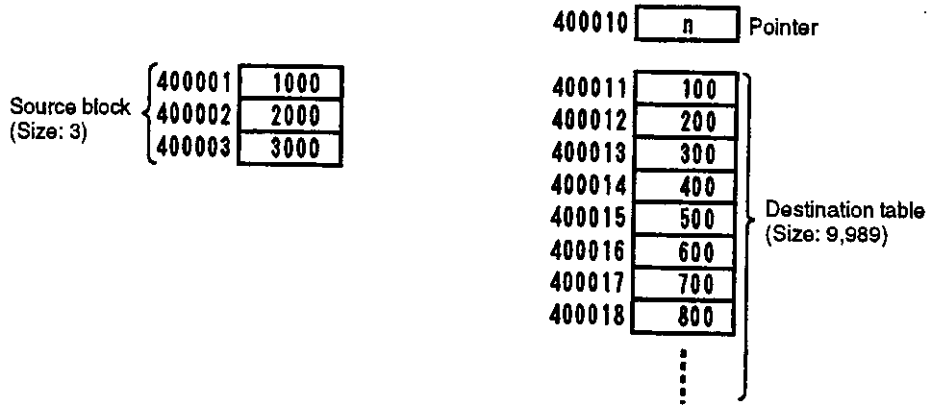
1. Function

1) Data is transferred using indexed blocks between source and destination data tables of different sizes. The pointer is on the destination side and determines the destination of the data.

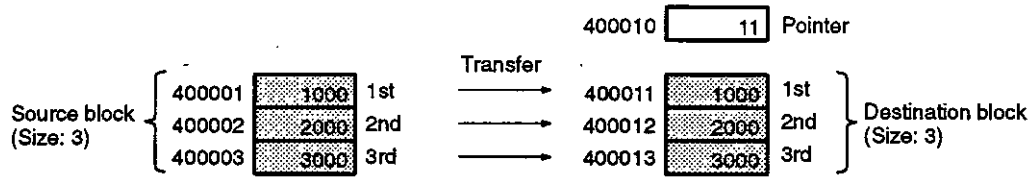
2) The source data table is called a source block. Coil tables, relay tables, and register tables can be used as the source block. Only holding register tables can be used as the destination table.

3) The contents of the source block is copied to the block in the destination table (called a destination block) specified by the pointer. The transfer is executed in one scan.

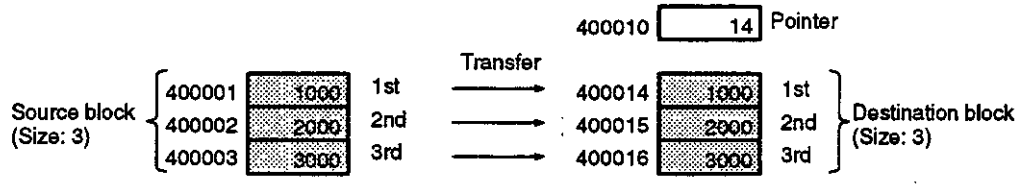
Example
The illustration shows the source block, destination table, and pointer.



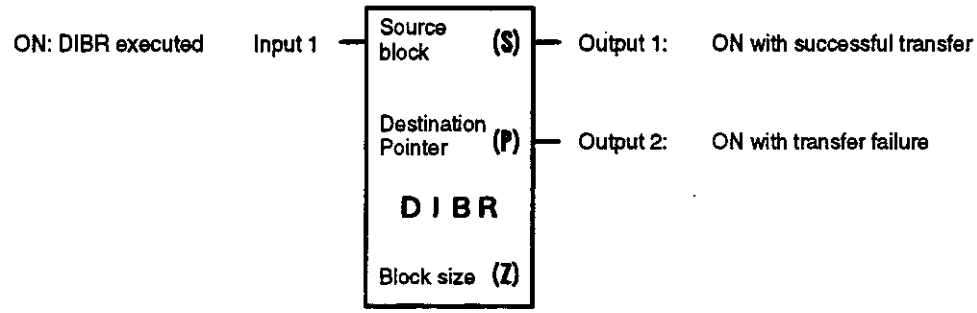
a) The contents of the source block is transferred to a destination block (400011 to 400013) when the transfer instruction is executed with a pointer value of 11.



b) The contents of the source block is transferred to a destination block (400014 to 400016) when the transfer instruction is executed with a pointer value of 14.



2. Structure



1) DIBR is the symbol for DESTINATION INDEXED BLOCK TRANSFER INSTRUCTION 2.

2) DIBR requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 4.3* lists the constants and coil, relay, and register reference numbers that can be specified.

Example

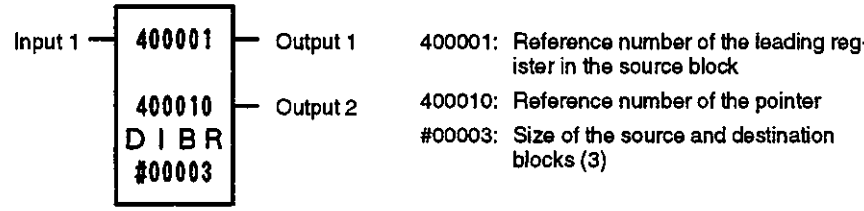


Table 4.3 DIBR Structural Elements

Element	Meaning	Possible settings	
Top (S)	Reference number of the leading register in the source block	Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (P)	Reference number of the pointer	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 or R20001 to R21024
Bottom (Z)	Size of the source and destination blocks	Specify the constant. The maximum value differs with specified reference type.	
		Coil:	#00001 to #00100
		Input relay:	#00001 to #00064
		Input register, holding register or constant register:	#00001 to #00100
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00100
		MC coil, MC relay or MC control relay:	#00001 to #00016
		MC control coil:	#00001 to #00010
		M code relay:	#00001 to #00006

Note When setting the reference numbers for coils and relays, m = 16n + 1, where m is the lower 5 digits of the reference number (and n = 0, 1, 2, etc.).

3) Destination Block

a) The destination block is a holding register table determined by the pointer value and the size of the source block. The following rules apply to holding register tables, where n is the pointer value and Z is the size of the source block.

(1) Reference number for the first holding register, D1: D1 = 400001 + (n – 1)

(2) Reference number for the last holding register, DE: DE = D1 + Z – 1

b) The following table shows these rules.

Pointer Value	Holding Register Table Used as Destination Block	
	Leading Reference No. (D1)	Last Reference No. (DE)
1	400001	400001 + Z – 1
2	400002	400002 + Z – 1
—	—	—
n	400001 + (n – 1)	400001 + (n – 1) + Z – 1
—	—	—
9999	409999	409999

4) Effective Range of the Pointer Value

a) The pointer value (n) must satisfy the following 3 conditions for the pointer to be defined as being in the effective range.

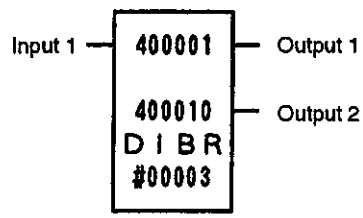
Condition 1: $1 \leq n \leq 9999$

Condition 2: The holding register table specified by n must actually exist, in other words, must satisfy the following two equations.
$400001 \leq 400001 + (n - 1) \leq 409999$
$400001 \leq 400001 + (n - 1) + Z - 1 \leq 409999$

Condition 3: The pointer must not be included in the destination block.

b) None the following examples satisfies the above conditions. Accordingly, the data will not be transferred if input 1 of the DIBR is ON. Output 1 will be OFF and output 2 will be ON.

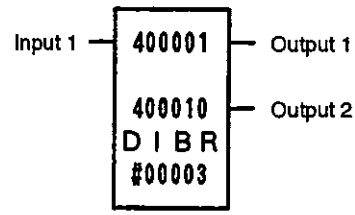
Example 1: Incorrect Application



400010 [20000] Pointer

Does not satisfy condition 1.

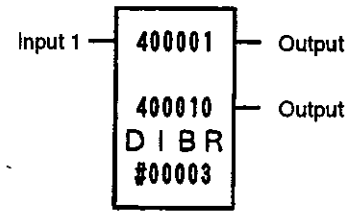
Example 2: Incorrect Application



400010 [9999] Pointer

Does not satisfy condition 2.

Example 3: Incorrect Application

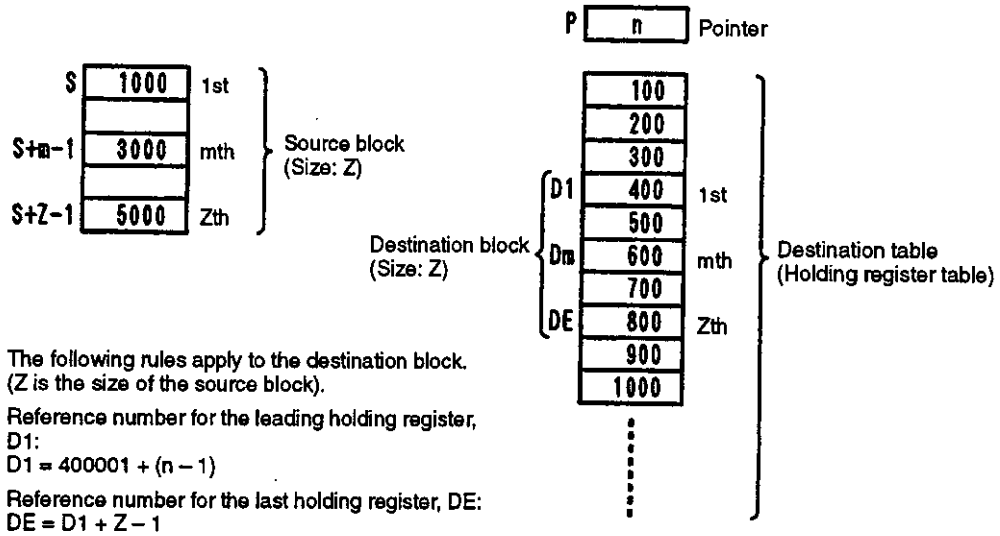


400010 [10] Pointer

Does not satisfy condition 3.

3. Operation

1) Status Before Execution

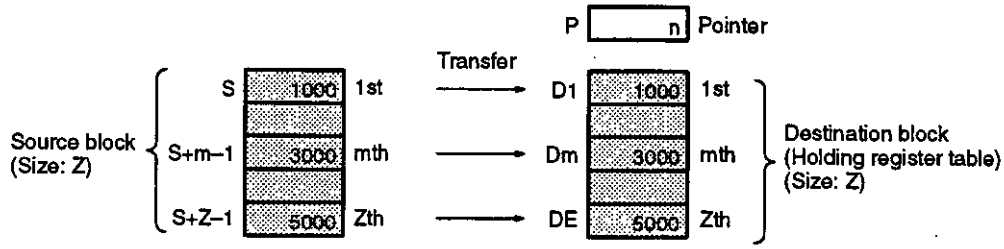


The following rules apply to the destination block.
(Z is the size of the source block).

Reference number for the leading holding register, D1:
D1 = 400001 + (n – 1)

Reference number for the last holding register, DE:
DE = D1 + Z – 1

2) If the pointer value n is in the effective range, the following data will be transferred when input 1 turns ON. The transfer is completed in one scan.



a) The holding register table (D1 to DE) will be selected as the destination block in accordance with the rules, and based on the pointer value (n) and the size of the source block (Z).

b) The contents of the source block is transferred to the selected holding register table. The contents of the m^{th} (m = 1 to Z) word in the source table is copied to the m^{th} (m = 1 to Z) word in the destination block.

c) The pointer value and the contents of the source block remain unchanged.

d) Output 1 turns ON. Output 2 remains OFF.

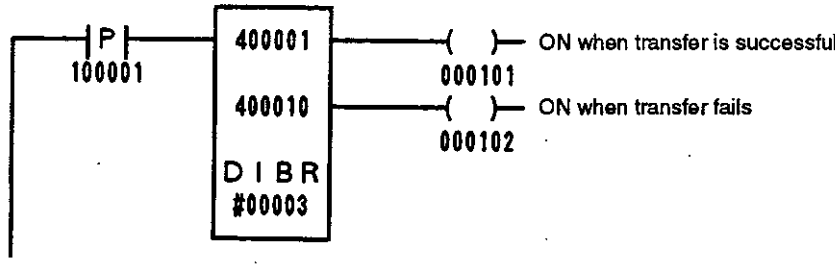
3) When the pointer value (n) is not in the effective range, no destination block exists. Therefore, if input 1 is ON, the following occurs:

a) The transfer is not executed.

b) Output 1 remains OFF. Output 2 turns ON.

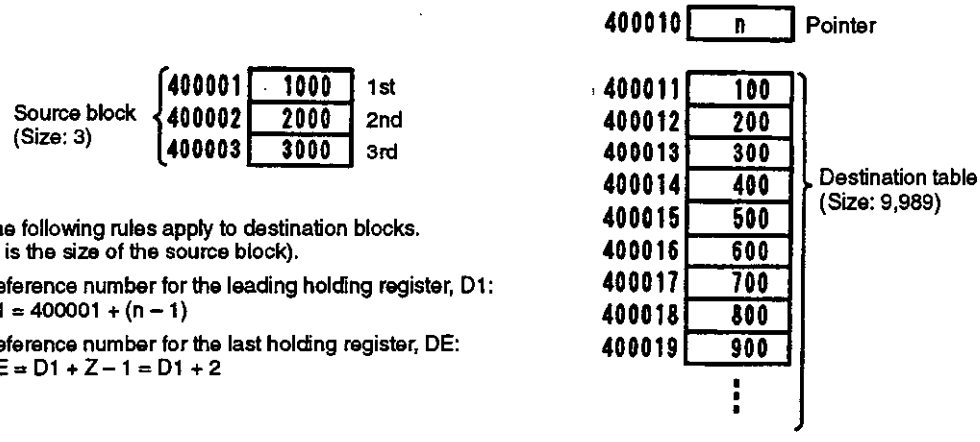
◀EXAMPLE▶ **4. Application Example**

1) Ladder Programming



2) Transfer Contents

a) Status Before Execution

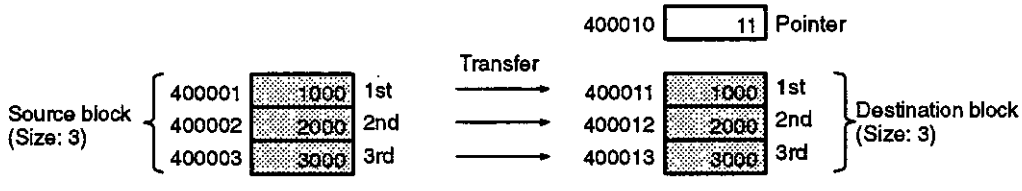


The following rules apply to destination blocks.
(Z is the size of the source block).

Reference number for the leading holding register, D1:
D1 = 400001 + (n – 1)

Reference number for the last holding register, DE:
DE = D1 + Z – 1 = D1 + 2

b) If the pointer value (n) is 11, the following data will be transferred when input relay 100001 changes from OFF to ON. The transfer is completed in one scan.



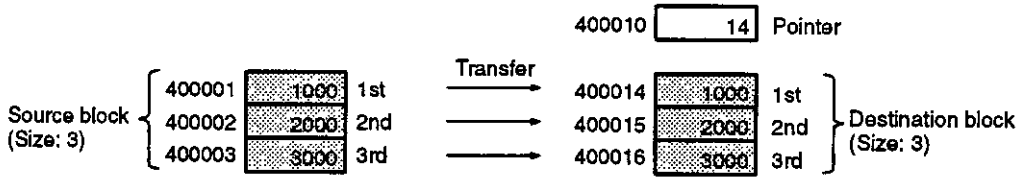
(1) A holding register table (400011 to 400013) is selected as the destination block, based on the pointer value (11) and the size of the source block (3).

(2) The contents of the source block is transferred to the selected destination block.

(3) The pointer and the contents of the source block remain unchanged.

(4) Coil 000101 turns ON only in scans where input relay 100001 changes from OFF to ON. Coil 000102 remains OFF.

c) The following data transfer is executed when the pointer value n is 14 and input relay 100001 is turned from OFF to ON. The transfer is completed in one scan.



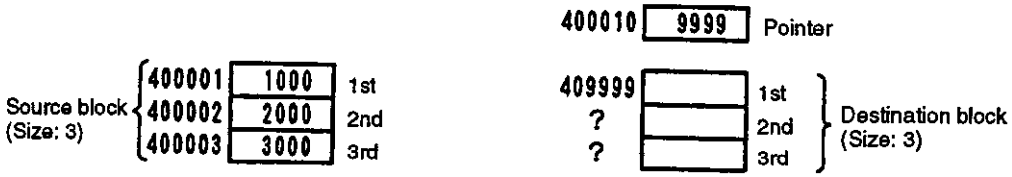
(1) A holding register table (400014 to 400016) is selected as the destination block based on the pointer value (14) and the size of the source block (3).

(2) The contents of the source block is transferred to the selected destination block.

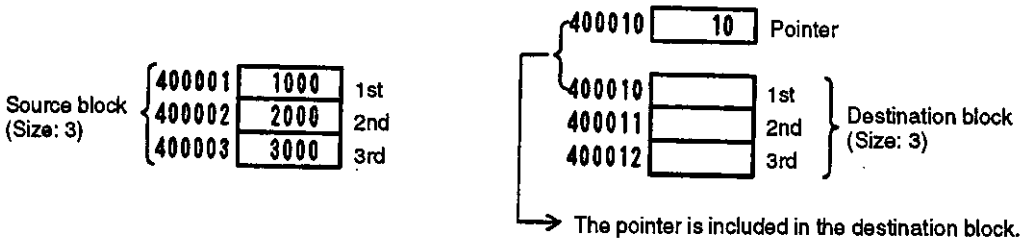
(3) The pointer value and the contents of the source block remain unchanged.

(4) Coil 000101 turns ON only in scans where input relay 100001 changes from OFF to ON. Coil 000102 remains OFF.

d) As the illustration shows, no destination block exists when the pointer value (n) is 9999. Accordingly, transfer is not executed if input relay 100001 is turned from OFF to ON. Coil 000101 remains OFF. Coil 000102 turns ON only in scans where input relay 100001 changes from OFF to ON.



e) As the illustration shows, the pointer (400010) is included in the destination block when the pointer value is 10. Accordingly, transfer is not executed even when input relay 100001 is turned from OFF to ON. Coil 000101 remains OFF. Coil 000102 turns ON only in scans where input relay 100001 changes from OFF to ON.



4.3.3 SOURCE INDEXED BLOCK TRANSFER 1 (SIBT)

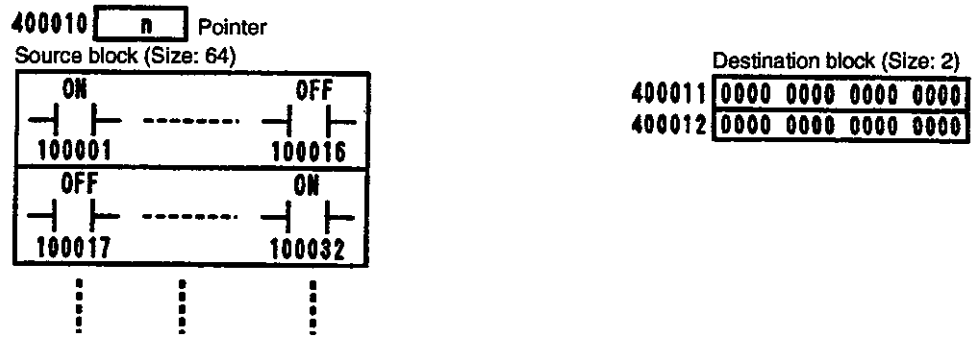
1. Function

1) Data is transferred using indexed blocks between source and destination data tables of different sizes. A pointer is used to determine the destination of the data in the destination table.

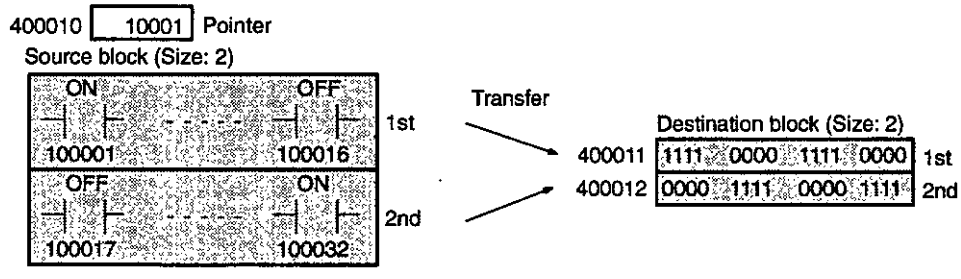
2) Coil tables, input relay tables, and input register tables can be used as the source table. Only holding register tables can be used as the destination table (called destination blocks).

3) The contents of the source block specified by the pointer (called the source block) are copied to the block in the destination table. The transfer is completed in one scan.

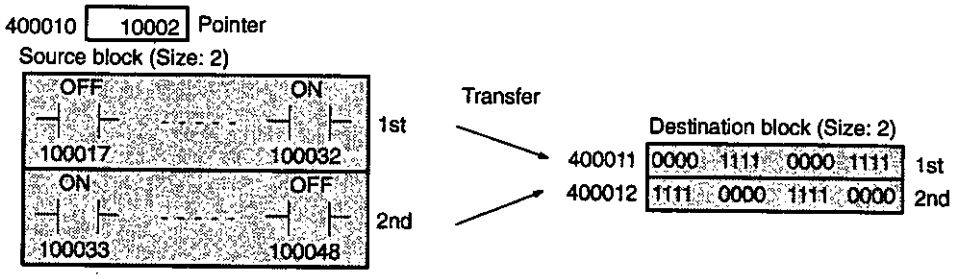
Example
The illustration shows the source table, destination block, and pointer.



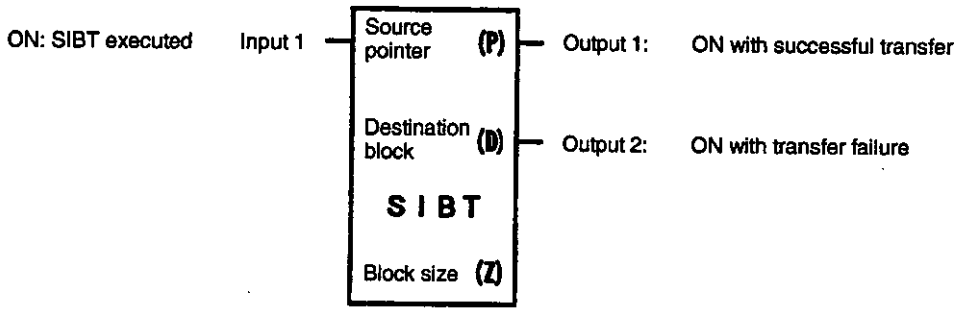
a) When executing the transfer instruction with the pointer value at 10001, the 32 input relays of the source block (100001 to 100032) are transferred to the 32 bits of the destination block.



b) When executing the transfer instruction with the pointer value at 10002, the 32 input relays of the source block (100017 to 100048) are transferred to the 32 bits of the destination block.



2. Structure



1) SIBT is the symbol for SOURCE INDEXED BLOCK TRANSFER INSTRUCTION 1.

2) SIBT requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 4.4* lists the constants and register reference numbers that can be specified.

Example

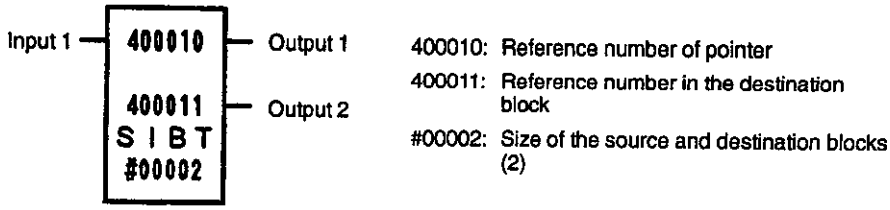


Table 4.4 SIBT Structural Elements

Element	Meaning	Possible settings
Top (P)	Reference number of the pointer	Holding register: 400001 to 409999 (W00001 to W09999) Constant register: 700001 to 704096 (K00001 to K04096) Link register: R10001 to R11024 or R20001 to R21024
Middle (D)	Leading reference number in the destination block	Holding register: 400001 to 409999 (W00001 to W09999) Link register: R10001 to R11024 or R20001 to R21024
Bottom (Z)	Size of the source and destination blocks	Constant: #00001 to #00100

3) Source Block

a) The source block is one of the following data tables as determined by the pointer value.

Pointer Value	Source Block
1 to 512	Coil table
10001 to 10064	Input relay table
30001 to 30512	Input register table

b) Source Block is a Coil Table

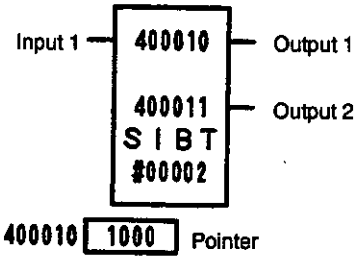
(1) The following rules apply to coil tables, where the pointer value is (n) and the size of the destination block is Z.
Reference number of the leading coil, S1: S1 = 000001 + 16 (n – 1)
Reference number of the last coil, SE: SE = S1 + 16Z – 1

(2) The following table outlines these rules.

Pointer Value	Coil Tables Used as Source Block	
	Leading Reference No. (S1)	Last Reference No. (SE)
1	000001	000001 + 16Z – 1
2	000017	000017 + 16Z – 1
—	—	—
n	000001 + 16 (n – 1)	000001 + 16 (n – 1) + 16Z – 1
—	—	—
512	008177	008192

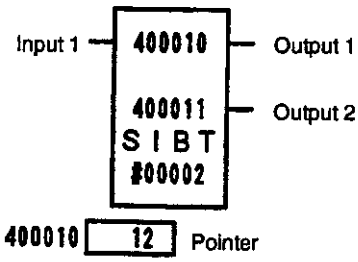
(3) The pointer value (n) must satisfy the following 3 conditions to be in the effective range.
Condition 1: $1 \leq n \leq 512$
Condition 2: The coil table specified by n actually must exist, i.e., must satisfy the following two equations:
$1 \leq 1 + 16 (n - 1) \leq 8177$
$1 \leq 1 + 16 (n - 1) + 16Z - 1 \leq 8192$
Condition 3: The pointer must not be included in the destination block.

Example 1: Incorrect Application



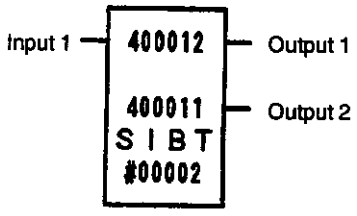
Does not satisfy condition 1.

Example 2: Incorrect Application



Does not satisfy condition 2.

Example 3: Incorrect Application



Does not satisfy condition 3.

c) Source Block is an Input Relay Table

(1) The following rules apply to input relays, where the pointer value is n and the size of the destination block is Z.
Reference number of the leading input relay, S1: S1 = 100001 + 16 (n – 10001)
Reference number of the last input relay, SE: SE = S1 + 16Z – 1

(2) The following table outlines these rules.

Pointer Value	Input Relay Table Used as Source Block	
	Leading Reference No. (S1)	Last Reference No. (SE)
10001	100001	100001 + 16Z – 1
10002	100017	100017 + 16Z – 1
—	—	—
n	100001 + 16 (n – 10001)	100001 + 16 (n – 10001) + 16Z – 1
—	—	—
10064	101009	101024

(3) The pointer value (n) must satisfy the following 3 conditions to be in the effective range.

Condition 1 ..... $10001 \leq n \leq 10064$

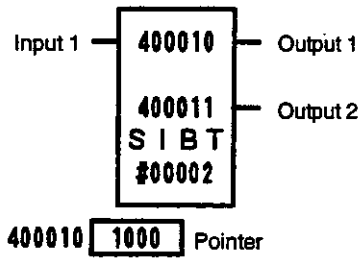
Condition 2 ..... The input relay table specified by n actually exists, i.e., satisfies the following 2 equations.

$100001 \leq 100001 + 16\ (n - 10001) \leq 101009$

$100001 \leq 100001 + 16\ (n - 10001) + 16Z - 1 \leq 101024$

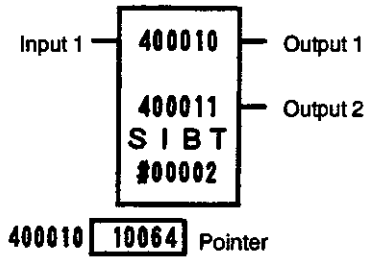
Condition 3 ..... The pointer is not included in the destination block.

Example 1



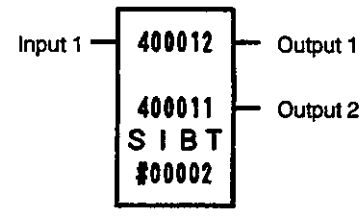
Does not satisfy condition 1.

Example 2



Does not satisfy condition 2.

Example 3



Does not satisfy condition 3.

d) Source Block is an Input Register Table

(1) The following rules apply to input register tables, where the pointer value is n and the size of the destination block is Z.

Reference number of the leading input register, S1: $S1 = 300001 + (n - 30001)$

Reference number of the last input register, SE: $SE = S1 + Z - 1$

(2) The rules are outlined on the following table.

Pointer Value	Input Register Table Used as Source Block	
	Leading Reference No. (S1)	Last Reference No. (SE)
30001	300001	300001 + Z – 1
30002	300002	300002 + Z – 1
—	—	—
n	300001 + (n – 30001)	300001 + (n – 30001) + Z – 1
—	—	—
30512	300512	300512

(3) The pointer value (n) must satisfy the following 3 conditions to be in the effective range.

Condition 1 ..... $30001 \leq n \leq 30512$

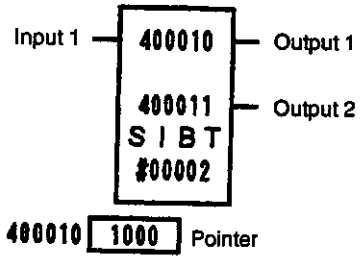
Condition 2 ..... The input register table specified by n actually exists, i.e., satisfies the following 2 equations.

$300001 \leq 300001 + (n - 30001) \leq 300512$

$300001 \leq 300001 + (n - 30001) + Z - 1 \leq 300512$

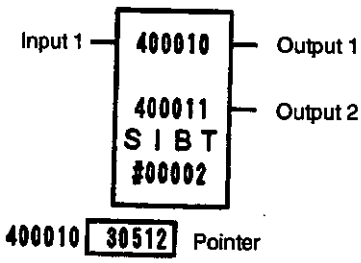
Condition 3 ..... The pointer is not included in the destination block.

Example 1



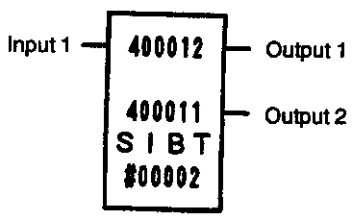
Does not satisfy condition 1.

Example 2



Does not satisfy condition 2.

Example 3



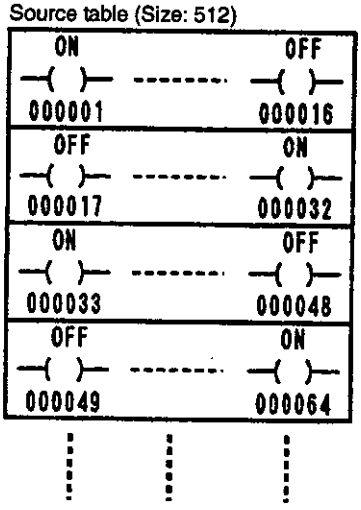
Does not satisfy condition 3.

3. Operation

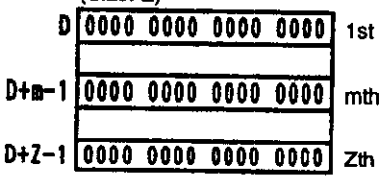
1) Source Block is a Coil Table

a) Status Before Execution

Source table (Size: 512)



Destination block (Holding register table) (Size: Z)

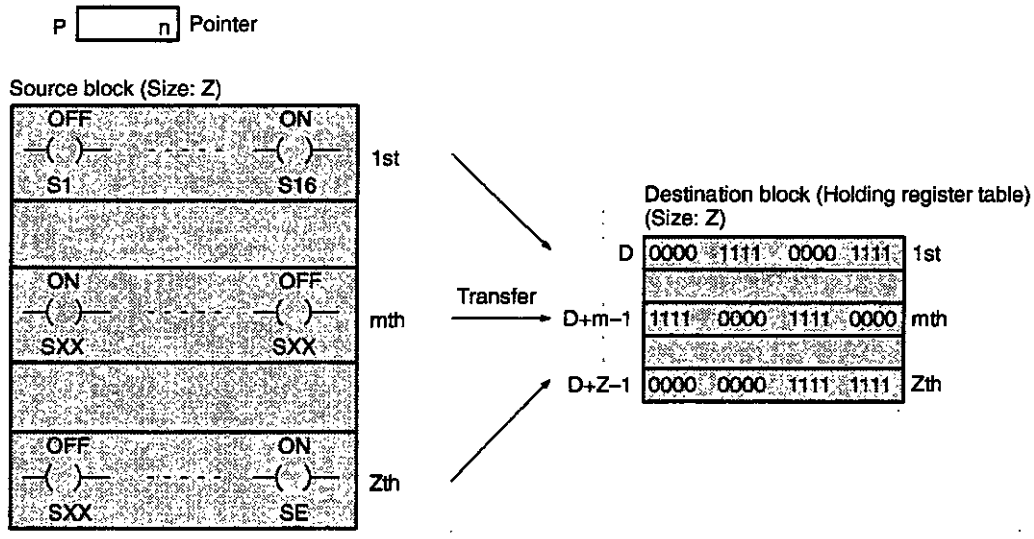


The following rules apply to the source block. (Z is the size of the destination block).

Reference number of the leading coil, S1:
S1 = 000001 + 16 (n – 1)

Reference number of the last coil, SE:
SE = S1 + 16Z – 1

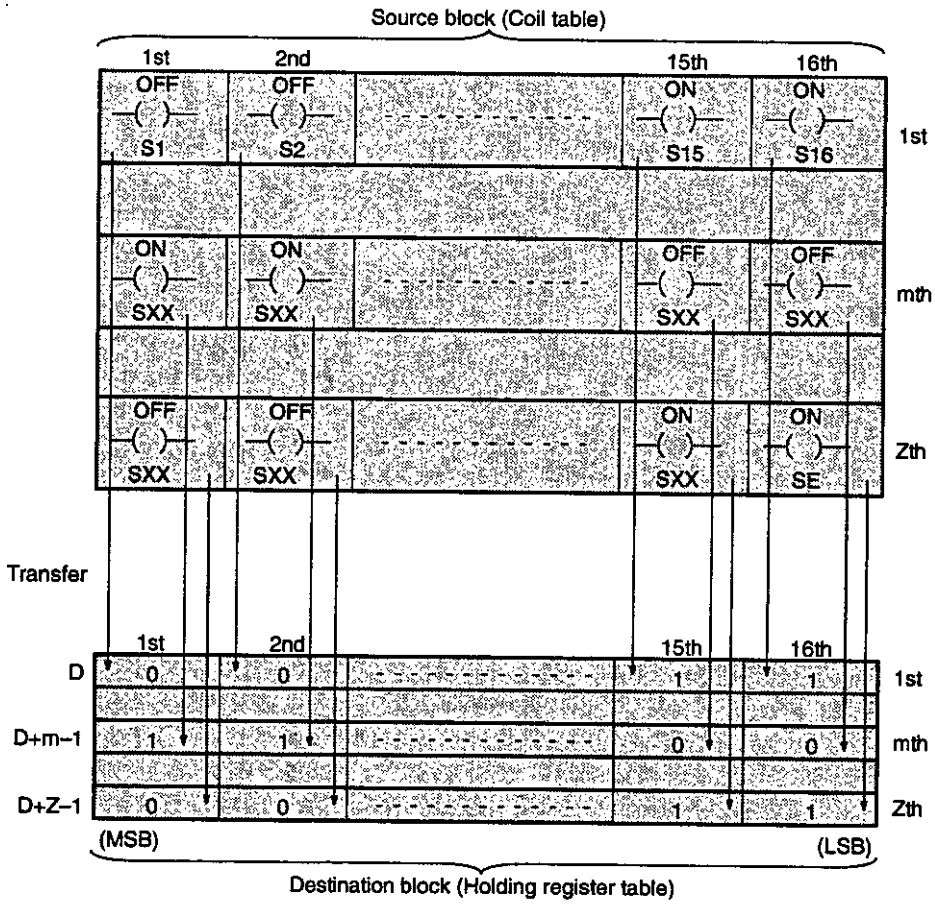
b) The following data transfer is executed when the pointer value n is in the effective range and input 1 is turned ON. The transfer is completed in one scan.



(1) A coil table (S1 to SE) is selected as the source block, in accordance with the rules and based on the pointer value (n) and the size of the destination block (Z).

(2) The contents of the selected coil table is copied to the destination block.

(3) The following chart shows the relationship that exists between the coil of the source block and the destination block bits.



(4) The pointer value and the content of the source block remain unchanged.

(5) Output 1 turns ON. Output 2 remains OFF.

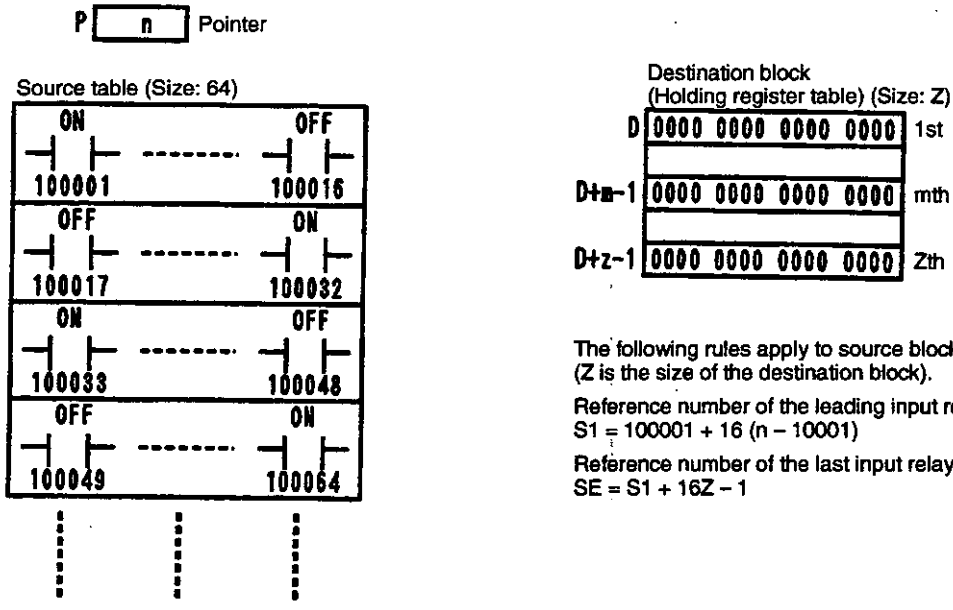
c) No source block exists when the pointer value (n) is outside the effective range. Therefore, even if input 1 is turned ON the following occurs.

(1) The data transfer is not executed.

(2) Output 1 remains OFF. Output 2 turns ON.

2) Source Block is an Input Relay Table

a) Status Before Execution

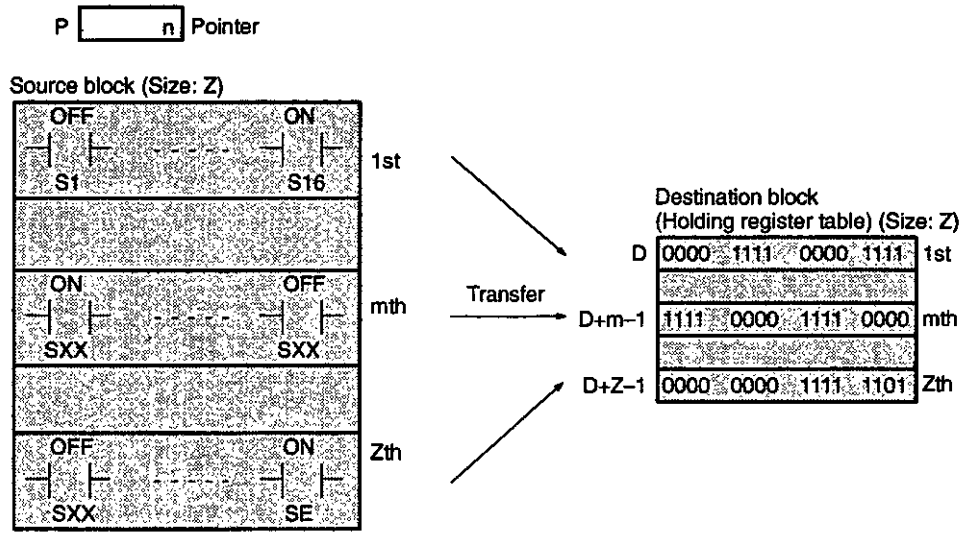


The following rules apply to source blocks.
(Z is the size of the destination block).

Reference number of the leading input relay, S1:
S1 = 100001 + 16 (n – 10001)

Reference number of the last input relay, SE:
SE = S1 + 16Z – 1

b) The following data transfer is executed when the pointer value (n) is in the effective range and input 1 is turned ON. The transfer is executed in one scan.



(1) An input relay table (S1 to SE) is selected as the source block in accordance with the rules and based on the pointer value (n) and the size of the destination block (Z).

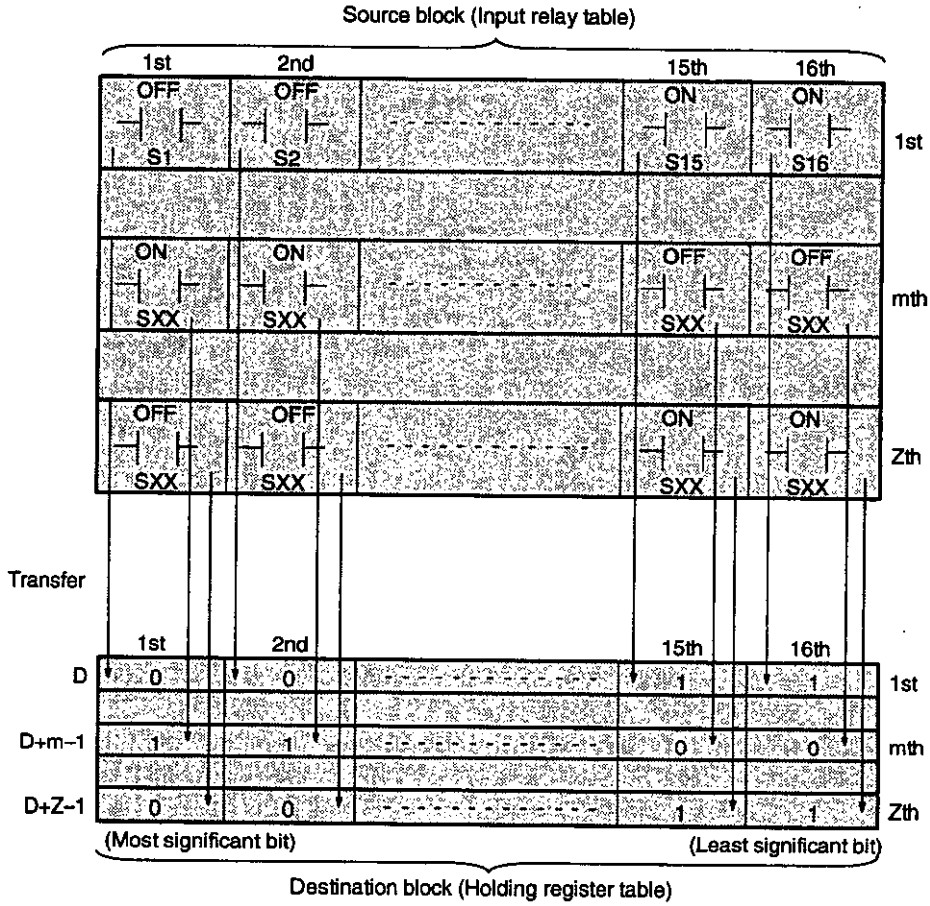
(2) The contents of the selected input relay table is transferred to the destination block.

(3) The relationship between the input relay of the source block and the destination block bits is shown in the figure associated with (6) below.

(4) The pointer value and the content of the source block remain unchanged.

(5) Output 1 turns ON. Output 2 remains OFF.

(6) **Correspondence between Input Relays and Bits**
The correspondence between the input relay of the source block and the destination block bits is shown in the following figure.



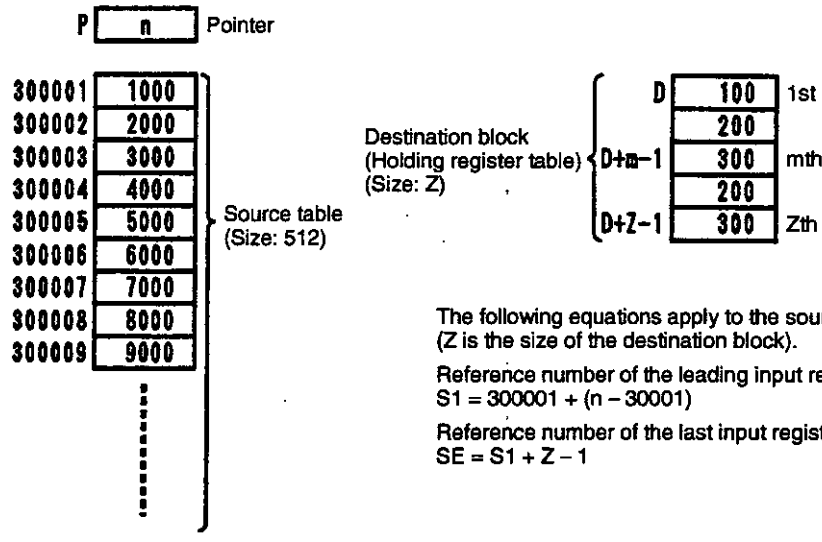
c) No source block exists when the pointer value n is not in the effective range and therefore, even if input 1 is turned ON, the following occurs.

(1) The data transfer is not executed.

(2) Output 1 remains OFF. Output 2 turns ON.

3) Source Blocks is an Input Register Table

a) Status Before Execution

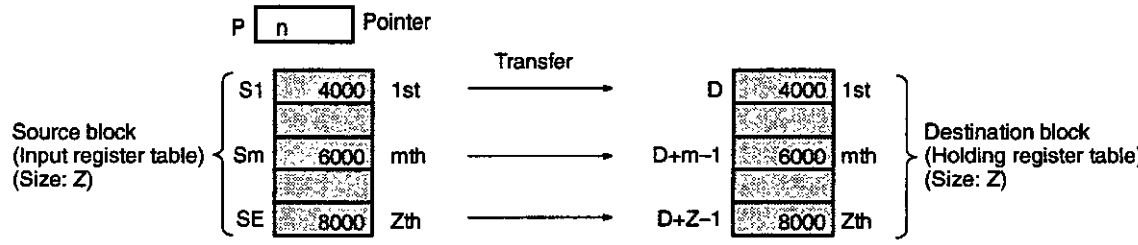


The following equations apply to the source block.
(Z is the size of the destination block).

Reference number of the leading input register, S1:
S1 = 300001 + (n – 30001)

Reference number of the last input register, SE:
SE = S1 + Z – 1

b) The following data transfer is executed when the pointer value (n) is in the effective range and input 1 is turned ON. The transfer is completed in one scan.



(1) An input register table (S1 to SE) is selected as the source block in accordance with the rules and based on the pointer value (n) and the size of the destination block (Z).

(2) The contents of the selected input register table are copied to the destination block. The m^{th} word (m = 1 to Z) in the source block is copied to the m^{th} word (m = 1 to Z) in the destination block.

(3) The pointer value and the content of the source block are unchanged.

(4) Output 1 turns ON. Output 2 remains OFF.

c) No source block exists when the pointer value (n) is not in the effective range, and therefore even if input 1 is turned ON, the following occurs.

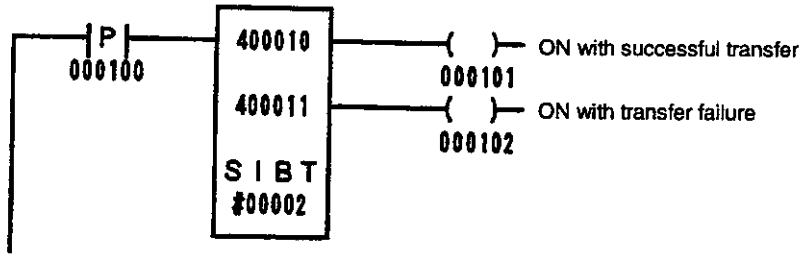
(1) The data transfer is not executed.

(2) Output 1 remains OFF. Output 2 turns ON.

4. Application Examples

◀EXAMPLE▶ **Example 1**

1) Ladder Programming

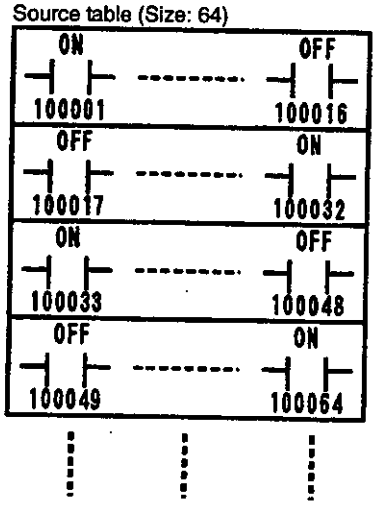


2) Transfer Contents

a) Status Before Execution

400010 [n] Pointer

Source table (Size: 64)



Destination block (Size: 2)

400011	1111 0000 1111 0000	1st
400012	0000 1111 0000 1111	2nd

The following rules apply to the source block.
(Z is the size of the destination block).

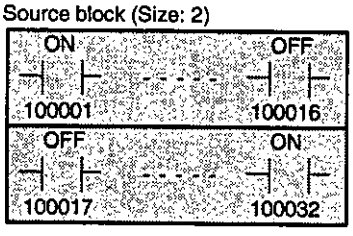
Reference number of the leading input relay, S1:
S1 = 100001 + 16 (n – 10001)

Reference number of the last input relay, SE:
SE = S1 + 16Z – 1 = S1 + 31

b) The following data transfer is executed when the pointer value (n) is 10001 and coil 000100 is turned from OFF to ON. The transfer is completed in one scan.

400010 [10001] Pointer

Source block (Size: 2)



Transfer

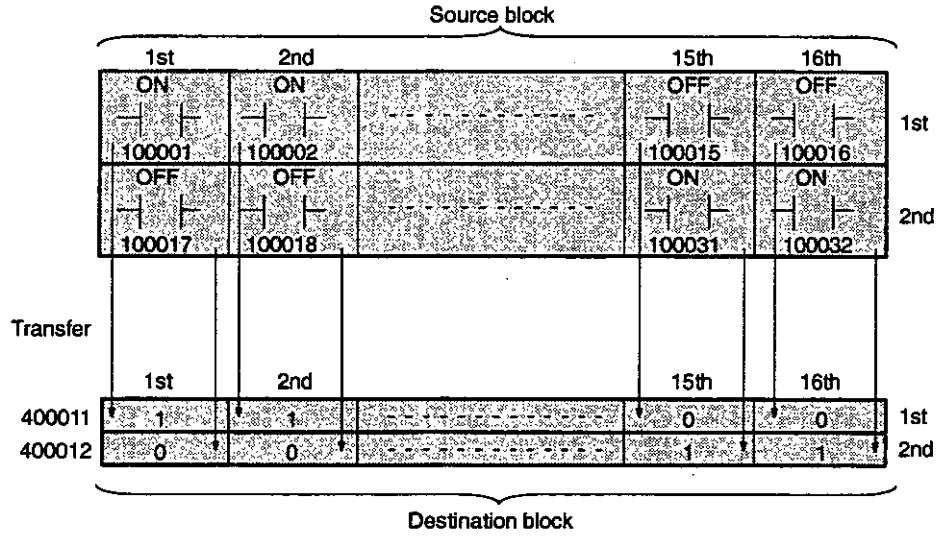
Destination block (Size: 2)

400011	1111 0000 1111 0000	1st
400012	0000 1111 0000 1111	2nd

(1) An input relay table (100001 to 100032) is selected as the source block based on the pointer value (10001) and the size of the destination block (2).

(2) The 32 input relays of the selected source block are transferred to the 32 destination block bits.

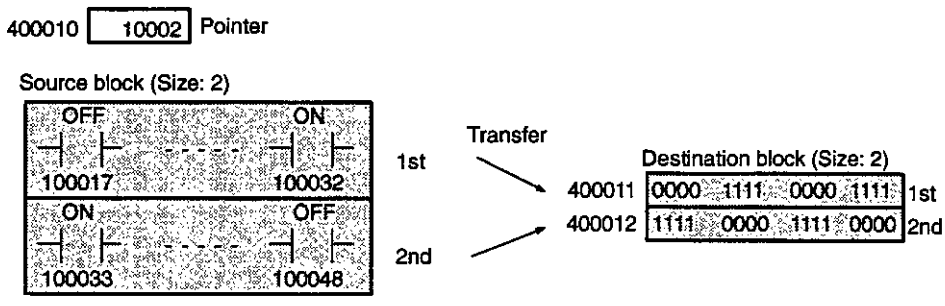
(3) The relationship between the input relay of the source block and the destination block bits is shown in the following figure.



(4) The pointer value and the content of the source block remain unchanged.

(5) Coil 000101: Turns ON only in scans where coil 000100 turns from OFF to ON.
Coil 000102: Remains OFF.

c) The following data transfer is executed when the pointer value (n) is 10002 and coil 000100 is turned from OFF to ON. The transfer is completed in one scan.

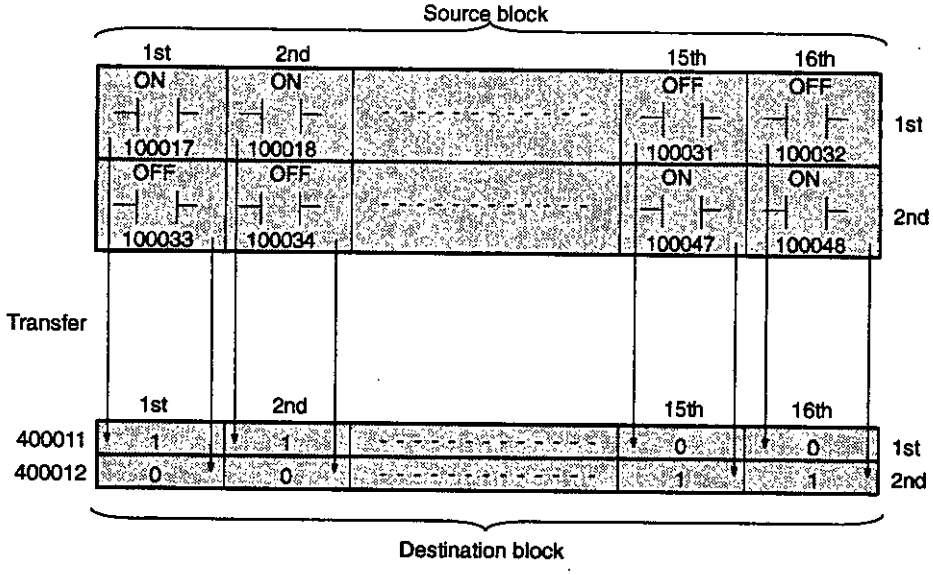


(1) An input relay table (100017 to 100048) is selected as the source block based on the pointer value (10002) and the size of the destination block (2).

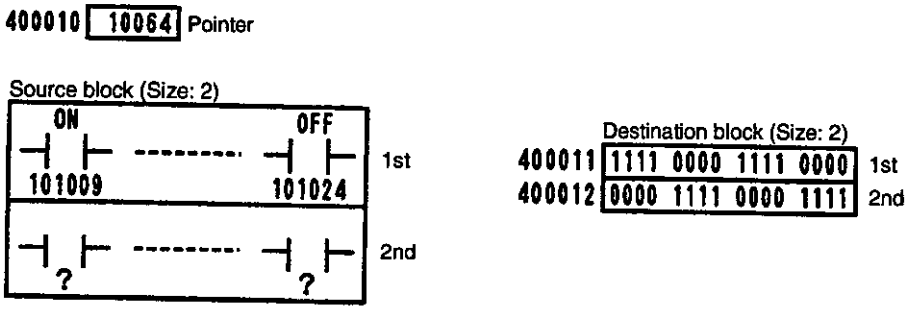
(2) The 32 input relays of the selected source block are transferred to the 32 destination block bits.

(3) The relationship between the source block input relays and the destination block bits is shown in the figure associated with (4) below.

(4) Correspondence between Input Relays and Bits

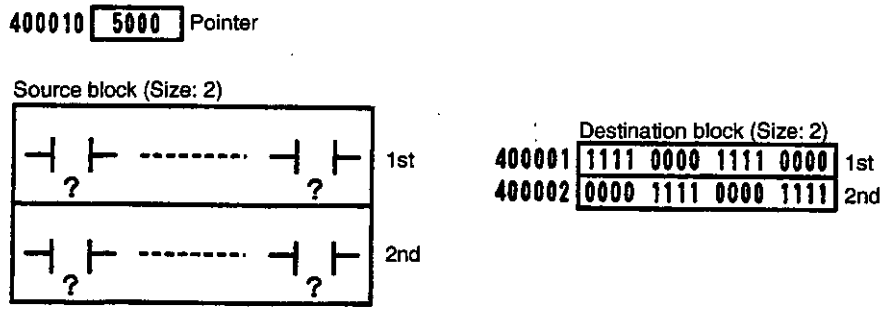


d) As shown in the figure below, no source block exists when the pointer value (n) is 10064. Accordingly, transfer is not executed even if coil 000100 is turned from OFF to ON. Coil 000101 remains OFF. Coil 000102 turns ON only in scans where coil 000100 turns from OFF to ON.



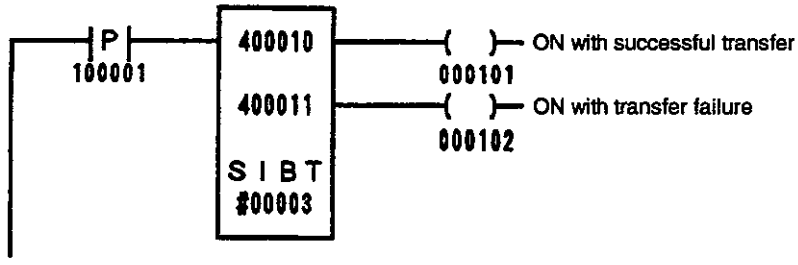
e) As shown in the figure below, no destination block exists when the pointer value (n) is 5000. Accordingly, the transfer is not executed even if coil 000100 is turned from OFF

to ON. Coil 000101 remains OFF. Coil 000102 turns ON only in scans where coil 000100 turns from OFF to ON.



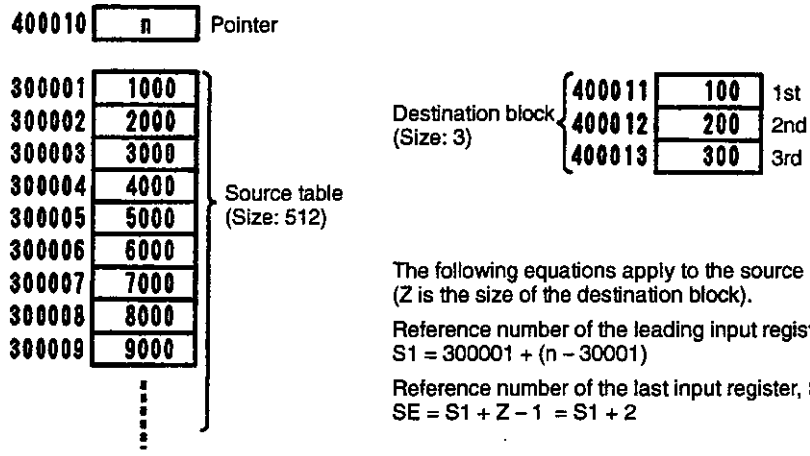
◀EXAMPLE▶ **Example 2**

1) Ladder Programming

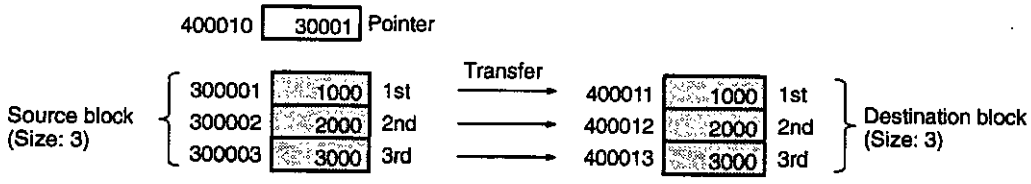


2) Transfer Contents

a) Status Before Execution



b) The following data transfer is executed when the pointer value (n) is 30001 and input relay 100001 is turned from OFF to ON. The transfer is completed in one scan.



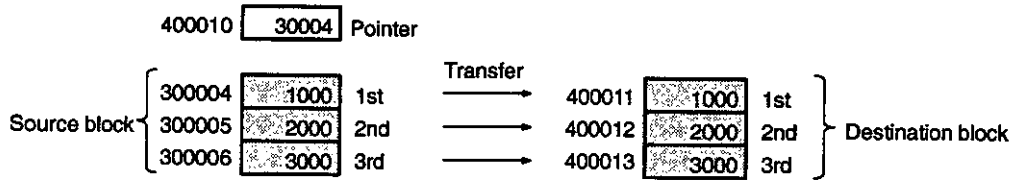
(1) An input register table (300001 to 300003) is selected as the source block based on the pointer value (30001) and the size of the destination block (3).

(2) The contents of the selected source block are transferred to the destination block.

(3) The pointer and the contents of the source block remain unchanged.

(4) Coil 000101 turns ON only in scans where input relay 100001 turns from OFF to ON. Coil 000102 remains OFF.

c) The following data transfer is executed when the pointer value (n) is 30004 and input relay 100001 is turned from OFF to ON. The transfer is completed in one scan.



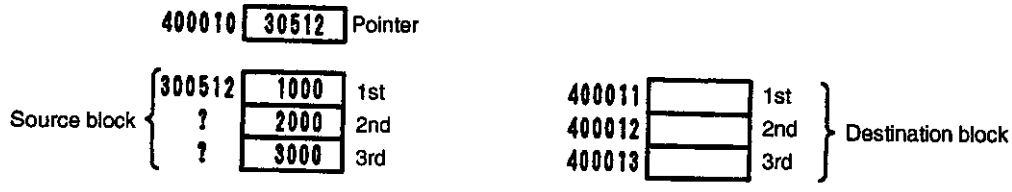
(1) An input register table (300004 to 300006) is selected as the source block based on the pointer value (30004) and the size of the destination block (3).

(2) The contents of the selected source block are transferred to the destination block.

(3) The pointer and the contents of the source block are unchanged.

(4) Coil 000101 turns ON only in scans where input relay 100001 turns from OFF to ON. Coil 000102 remains OFF.

d) As the figure below shows, no destination block exists when the pointer value (n) is 30512. Accordingly, the transfer will not be executed even if input relay 100001 is turned from OFF to ON. Coil 000101 remains OFF. Coil 000102 turns ON only in scans where coil 000100 turns from OFF to ON.



4.3.4 SOURCE INDEXED BLOCK TRANSFER 2 (SIBR)

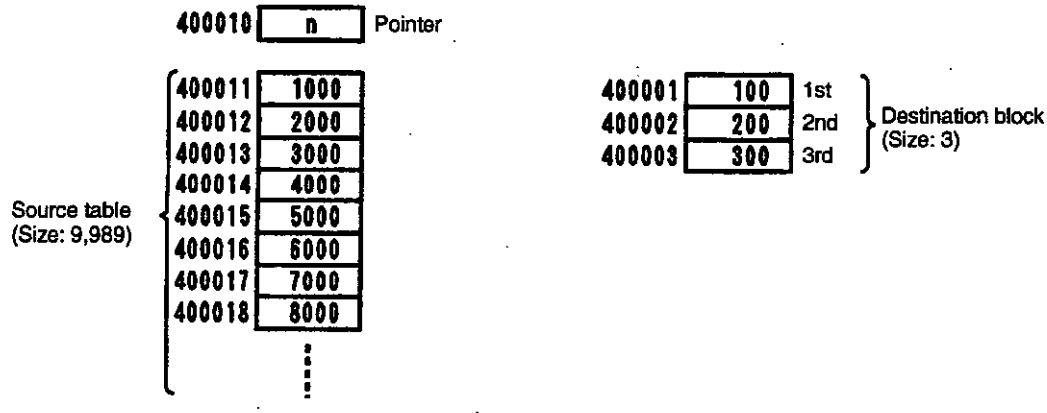
1. Function

1) Source indexed block transfer 2 executes the transfer of data between source and destination data tables of different size. A pointer is used to determine the origin of the data in the source table.

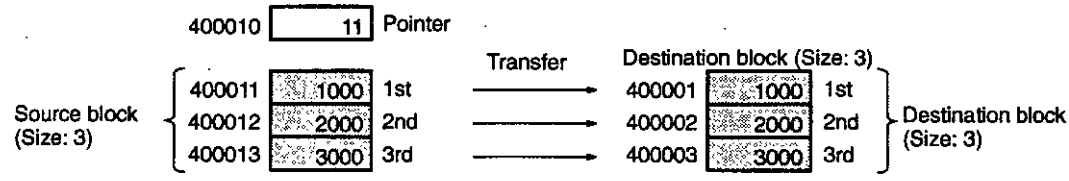
2) Source tables and destination data tables (called destination blocks) must be holding register tables.

3) The contents of the block in the source table specified by the pointer (called the source block) are copied to the destination block. The transfer is completed in one scan.

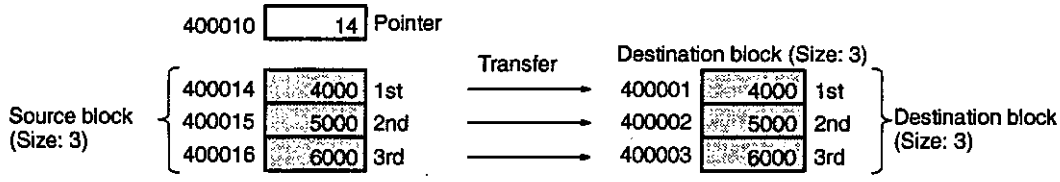
Example
The figure below shows source tables, destination blocks and pointers.



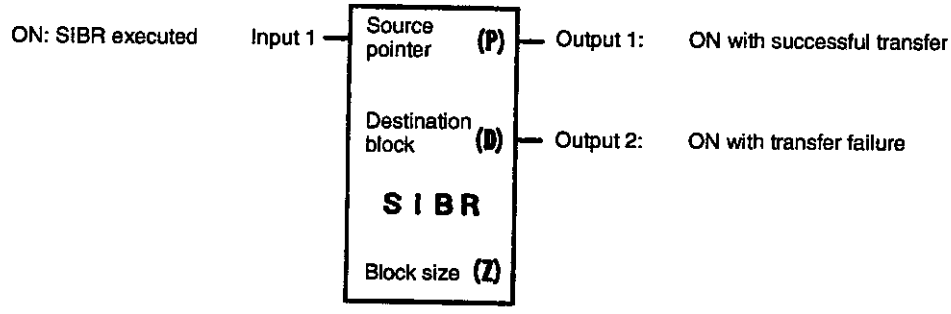
a) When a transfer instruction with a pointer value of 11 is executed, the content of the source block (400011 to 400013) is transferred to the destination block.



b) When a transfer instruction with a pointer value of 14 is executed, the content of the source block (400014 to 400016) is transferred to the destination block.



2. Structure



1) SIBR is the symbol for SOURCE INDEXED BLOCK TRANSFER INSTRUCTION 2.

2) SIBR requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 4.5* lists the constants and register reference numbers that can be specified.

Example

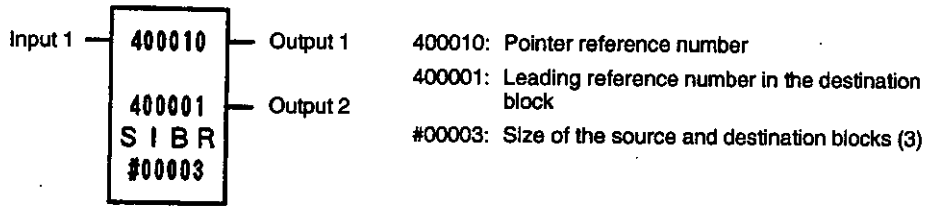


Table 4.5 SIBR Structural Elements

Element	Meaning	Possible settings
Top (P)	Pointer reference number	Holding register: 400001 to 409999 (W00001 to W09999) Constant register: 700001 to 704096 (K00001 to K04096) Link register: R10001 to R11024, R20001 to R21024
Middle (D)	Leading reference number in the destination block	Holding register: 400001 to 409999 (W00001 to W09999) Link register: R10001 to R11024, R20001 to R21024
Bottom (Z)	Size of the source and destination blocks	Constant: #00001 to #00100

3) Source Blocks

a) Source blocks are holding register tables determined by the pointer value and the size of the destination block. The following rules apply to holding register tables, where the pointer value is n and the size of the destination block is Z.

(1) Reference number for the leading holding register, S1: S1 = 400001 + (n – 1)

(2) Reference number for the last holding register, SE: SE = S1 + Z – 1

b) The following table shows these rules.

Pointer Value	Holding Register Table Used as Source Block	
	Leading Reference No. (S1)	Last Reference No. (SE)
1	400001	400001 + Z – 1
2	400002	400002 + Z – 1
—	—	—
n	400001 + (n – 1)	400001 + (n – 1) + Z – 1
—	—	—
9999	409999	409999

4) Effective Range of the Pointer Value

a) The pointer value (n) must satisfy the following 3 conditions to be in the effective range.

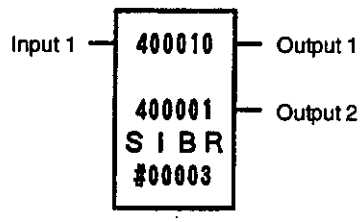
Condition 1 ..... $1 \leq n \leq 9999$

Condition 2 ..... The holding register table specified by n actually exists, in other words, satisfies the following two equations.
$400001 \leq 400001 + (n - 1) \leq 409999$
$400001 \leq 400001 + (n - 1) + Z - 1 \leq 409999$

Condition 3 ..... The pointer is not included in the destination block.

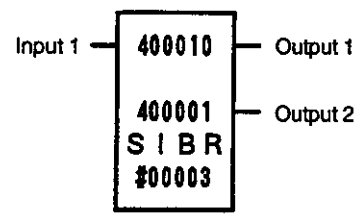
b) None the following examples satisfies the above conditions. Accordingly, the data will not be transferred even if input 1 of the SIBR is ON. Output 1 will be OFF and output 2 will be ON.

Example 1



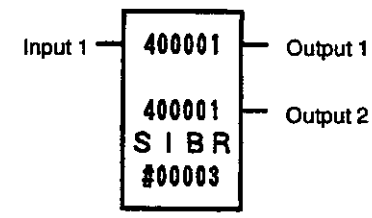
400010 [20000] Pointer
Does not satisfy condition 1

Example 2



400010 [9999] Pointer
Does not satisfy condition 2

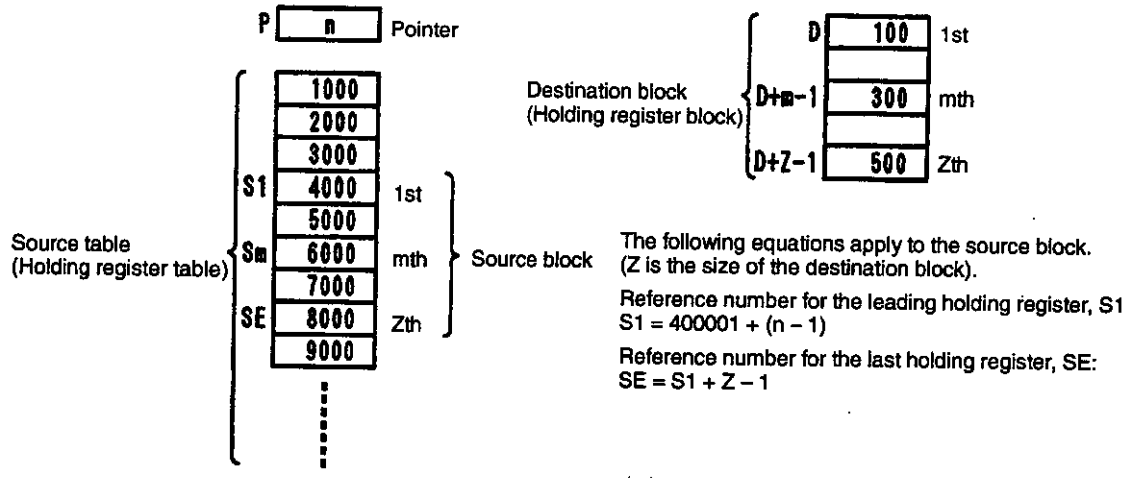
Example 3



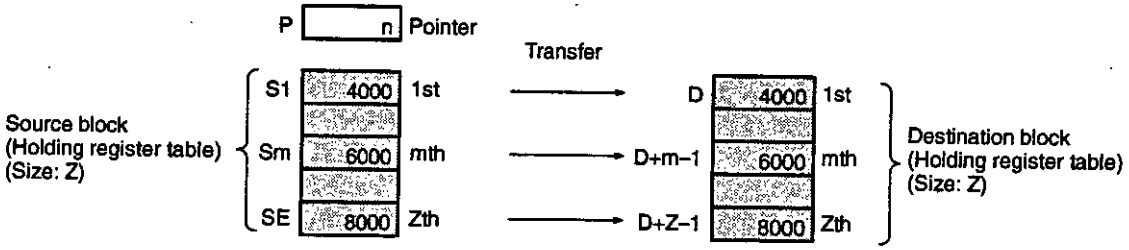
400001 [10] Pointer
Does not satisfy condition 3

3. Operation

1) Status Before Execution



2) If the pointer value (n) is in the effective range, the following data will be transferred when input 1 is turned ON. The transfer is completed in one scan.



a) A holding register table (S1 to SE) will be selected as the source block in accordance with the rules, and based on the pointer value (n) and the size of the source block (Z).

b) The content of the selected holding register table is transferred to the destination block. The contents of the m^{th} (m = 1 to Z) word in the source block are copied to the m^{th} (m = 1 to Z) word in the destination block.

c) The pointer value and the content of the source block remain unchanged.

d) Output 1 turns ON. Output 2 remains OFF.

3) When the pointer value (n) is not in the effective range, no source block exists. Therefore, even if input 1 is ON, the following occurs:

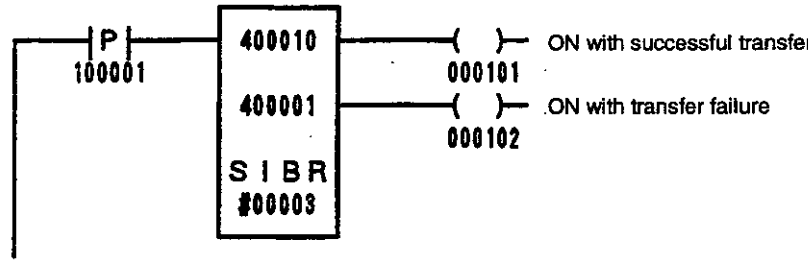
a) The transfer is not executed.

b) Output 1 remains OFF. Output 2 turns ON.

◀EXAMPLE▶

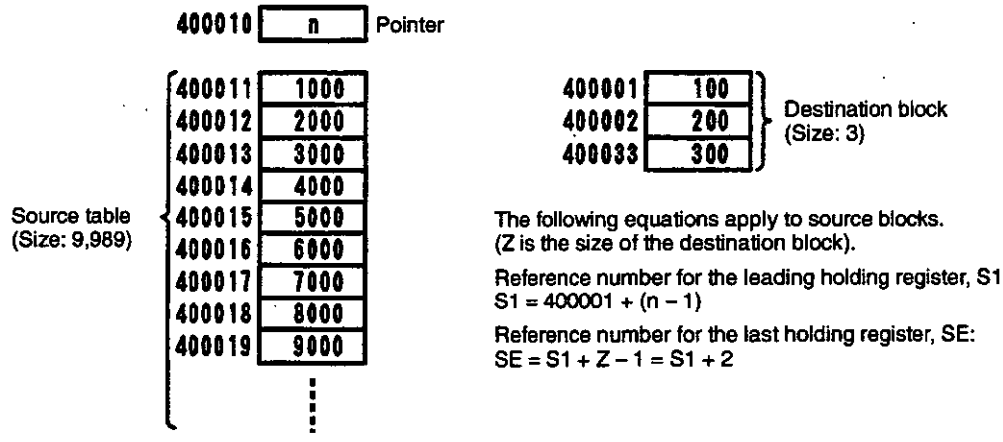
4. Application Example

1) Ladder Programming

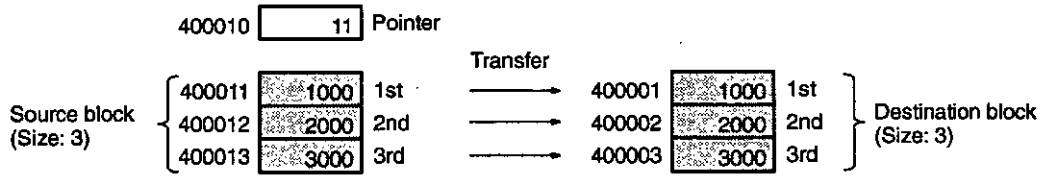


2) Transfer Contents

a) Status Before Execution



b) If the pointer value (n) is 11, the following data will be transferred when input relay 100001 turns from OFF to ON. The transfer is completed in one scan.



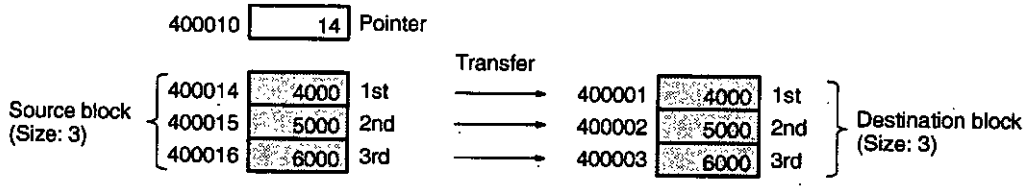
(1) A holding register table (400011 to 400013) is selected as the source block, based on the pointer value (11) and the size of the destination block (3).

(2) The contents of the selected source block is transferred to the destination block.

(3) The pointer and the content of the destination block remain unchanged.

(4) Coil 000101 turns ON only in scans where input relay 100001 turns from OFF to ON. Coil 000102 remains OFF.

c) The following data transfer is executed when the pointer value (n) is 14 and input relay 100001 is turned from OFF to ON. The transfer is completed in one scan.



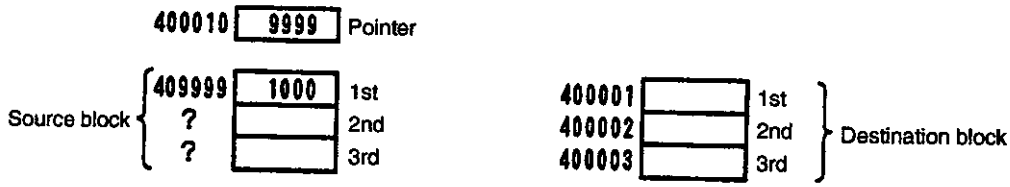
(1) A holding register table (400014 to 400016) is selected as the source block based on the pointer value (14) and the size of the destination block (3).

(2) The contents of the selected source block is transferred to the destination block.

(3) The pointer value and the content of the source block remain unchanged.

(4) Coil 000101 turns ON only in scans where input relay 100001 turns from OFF to ON. Coil 000102 remains OFF.

d) As the figure shows, no source block exists when the pointer value (n) is 9999. Accordingly, transfer is not executed even if input relay 100001 is turned from OFF to ON. Coil 000101 remains OFF. Coil 000102 turns ON only in scans where input relay 100001 turns from OFF to ON.



4.4 Building Programs

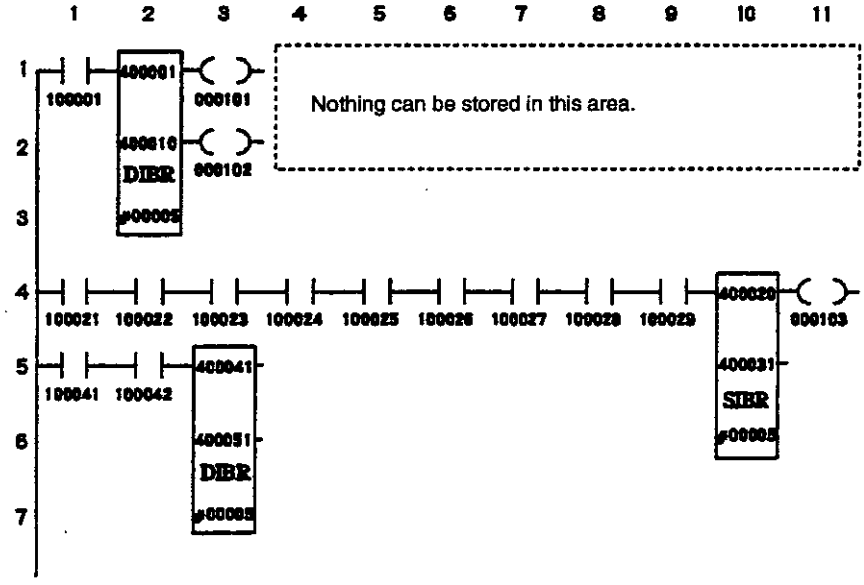
This section describes precautions that should be taken when designing programs that contain data transfer instructions.

4.4.1 Storage Locations on Networks

All indexed data transfer instructions require three elements (top, middle, and bottom) located vertically on the network, so they can be stored anywhere on a 5-row by 10-column matrix (rows 1 through 5 and columns 1 through 10).

Note Indexed data transfer instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

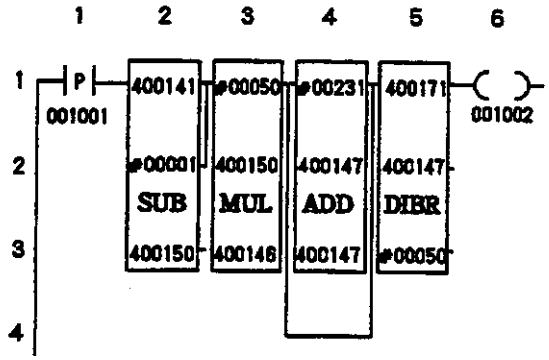
Example



4.4.2 Inputs

Inputs to indexed data transfer can be connected to relay elements (except coils) and outputs from timers, counters, math instructions, data transfer instructions, other instructions, etc.

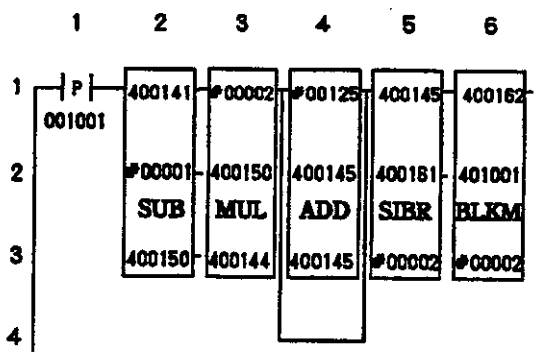
Example



4.4.3 Outputs

Outputs from indexed data transfer can be connected to any of the following: coils, contacts, inputs to instructions, inputs to data transfer instructions, etc.

Example



Note The input relays used in the destination for INDEXED BLOCK TRANSFER 1 (DIBT) are turned ON/OFF in accordance with the results of solving DIBT. That status is maintained until the completion of the scan cycle. The status at the beginning of the next scan cycle depends on whether the relays have been allocated to I/O and whether they are in enabled or disabled mode. The following table outlines the differences.

Table 6.6 Status at Beginning of Input Relay Scan Cycle

I/O Allocation	Enable/Disable	Input Relay Status at Beginning of Scan Cycle
No I/O allocation	Enabled	Status determined by solving DIBT remains.
	Disabled ON	
	Disabled OFF	
I/O allocation	Enabled	In accordance with the input signal
	Disabled ON	Turns ON.
	Disabled OFF	Turns OFF.

Note Do not use the positive or negative transitional contacts for the destination input relay for DESTINATION INDEXED BLOCK TRANSFER 1 (DIBT). The original operation, where the status is turn for only one scan depending on the ON/OFF status of the corresponding relay, will not be performed.

Matrix Instructions

5

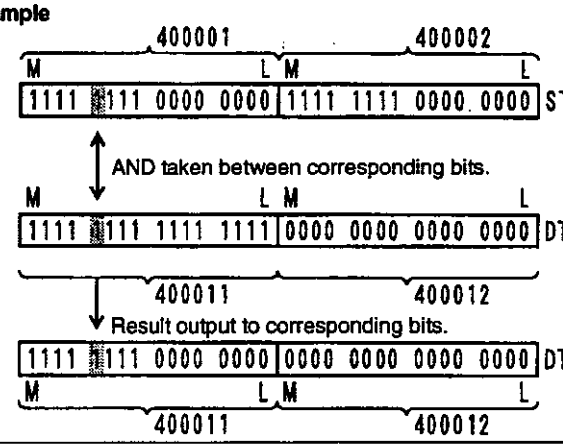
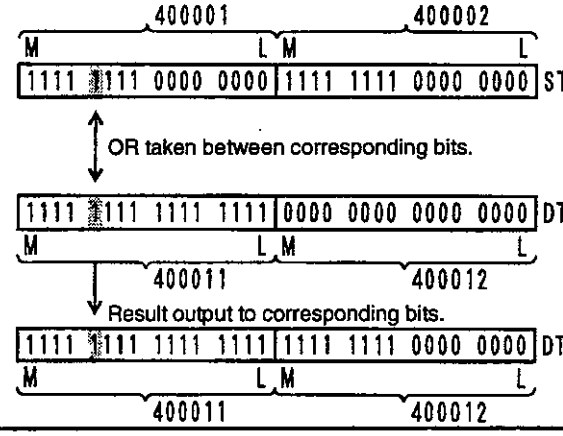
This chapter covers the instructions that perform various operations on matrixes.

5.1 Matrix Instructions

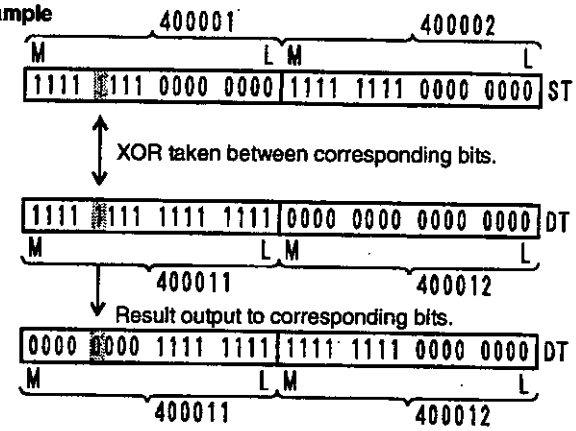
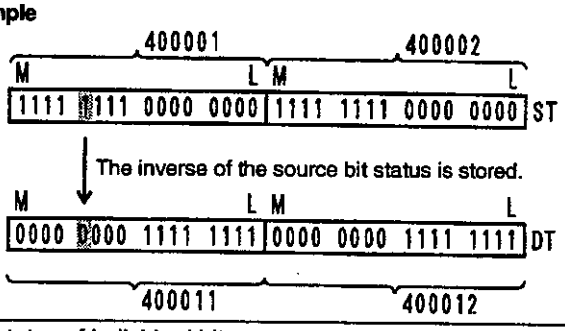
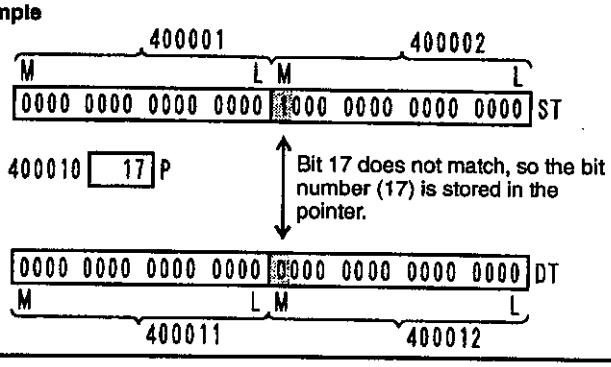
This section introduces the matrix instructions. Matrix instructions perform AND, OR, and other bit operations on data stored in data memory (including holding registers, input relays, coils, etc.). Matrix instructions can be used for complex, advanced data processing operations that are difficult to perform with the basic instructions, math instructions, or other instructions.

- The ten matrix instructions are outlined in the following table.

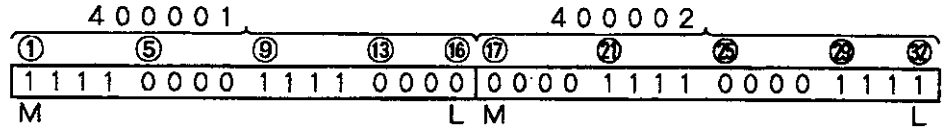
Table 5.1 Matrix Instructions

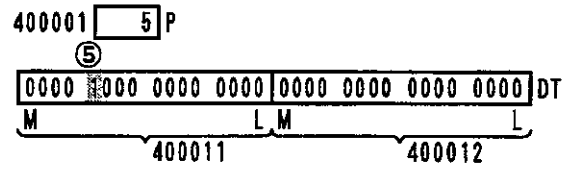
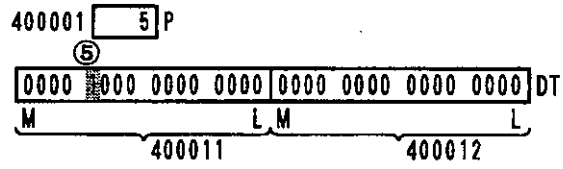
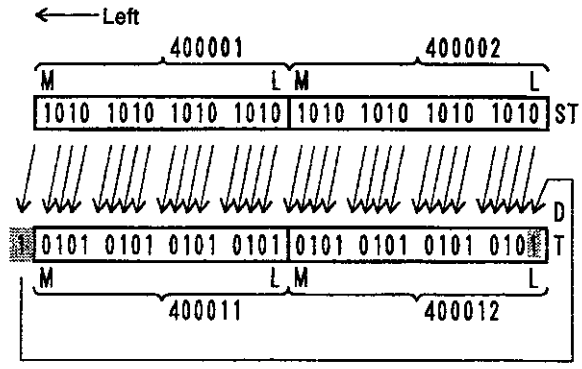
Name	Symbol	Function	Page
LOGICAL AND	AND	An AND is performed between corresponding bits of the source table (ST) and the destination table (DT) and the result is stored in the destination table. **Example** 	5-11
LOGICAL OR	OR	An OR is performed between corresponding bits of the source table (ST) and the destination table (DT) and the result is stored in the destination table. **Example** 	5-16

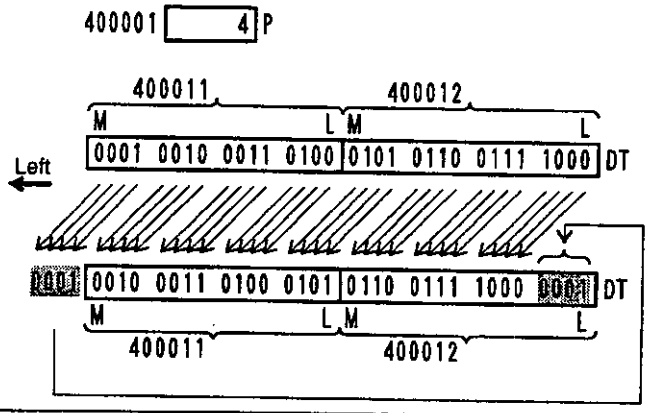
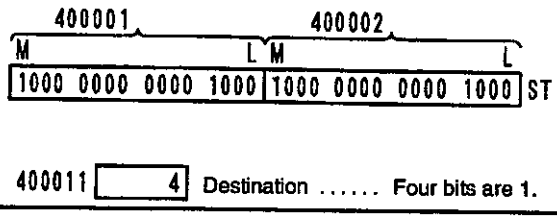
Abbreviations ST: Source table
DT: Destination table
P: Pointer
M: Most significant bit
L: Least significant bit

Name	Symbol	Function	Page
LOGICAL EXCLUSIVE OR	XOR	An exclusive OR is perform between corresponding bits of the source table (ST) and the destination table (DT) and the result is stored in the destination table. **Example** 	5-20
LOGICAL COMPLEMENT	COMP	The status of the bits in the source table (ST) are inverted and stored in the destination table (DT). **Example** 	5-24
LOGICAL COMPARE	CMPR	The status of individual bits are compared between the source table (ST) and the destination table (DT) and the bit numbers of the bits with different status is output using a pointer (P). **Example** 	5-29

Abbreviations Bit numbers are used to identify the position of a bit and are defined as shown in the following diagram.



Name	Symbol	Function	Page
LOGICAL BIT MODIFY	MBIT	A pointer (P) is used to force the status of an arbitrary bit in the destination table (DT) to either 1 or 0. **Example:** If the pointer is set to 5 and inputs 1 and 2 to the LOGICAL BIT MODIFY instruction are both ON, bit #5 will be force-set to 1. 	5-38
LOGICAL SENSE	SENS	A pointer (P) is used to detect the status of an arbitrary bit in the destination table (DT). **Example:** If the pointer is set to 5 and input 1 to the LOGICAL SENSE instruction is ON, output 2 will turn ON to indicate that the status of bit #5 is 1. 	5-44
LOGICAL BIT ROTATE	BROT	The bit pattern in the source table (ST) is shifted one bit to the left or to the right and stored in the destination table (DT). **Example:** Shift to the Left 	5-51

Name	Symbol	Function	Page
LOGICAL MULTI-BIT ROTATE	MROT	The bit pattern in the destination table (ST) is shifted a specified number of bits (1 to 15) to the left or to the right. **Example**: Pattern shifted four bits to the left. 	5-58
LOGICAL BIT COUNT	BCNT	The number of bits that are 1 or 0 in the source table (ST) are counted. **Example:** Counting the number of 1 bits 	5-65

5.2 Basic Information on Matrix Instructions

This section describes basic information required to use the matrix instructions.

5.2.1 Data Tables and Table Size

The following terms have the same meaning for matrix instructions as they do with data transfer instructions. Refer to *3.2 Data Transfer Instruction Terminology* for definitions of these terms.

- Data table
- Register table
- Coil table
- Relay table
- Table size

5.2.2 Bit Numbers

1. Data Processing Units

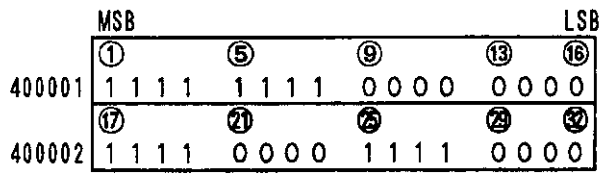
Matrix instructions process data by the smallest unit of a data table, i.e., by individual bits, coils, or relays.

2. Bit Numbers

1) Because matrix instruction process data by the smallest unit of a data table (i.e., by individual bits, coils, or relays) individual bits, coils, and relays are identified by bit numbers that indicate the position of the bit, coil, or relay within the table.

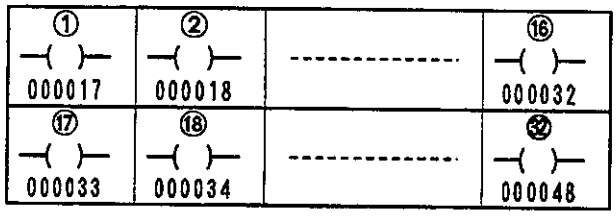
2) Bit numbers are allocated consecutively from the leading bit in the first register of the table starting from 1, i.e., 1, 2, 3, etc.

Example 1



MSB: Most significant bit
LSB: Least significant bit
① to ㉜: Bit number

Example 2



5.2.3 Source Tables and Destination Tables

Source table and destination table are defined as below.

1) Source Table

a) One of the data tables that undergoes a matrix operation is called a source table. If the table size of the source table is 1, then the source table is simply called the source.

b) Except when used as a pointer, the data in the source table does not change even when an operation is executed.

c) The following instructions use the source table as a pointer.

(1) LOGICAL BIT MODIFY (MBIT)

(2) LOGICAL SENSE (SENS)

(3) LOGICAL MULTI-BIT ROTATE (MROT)

2) Destination Table

a) The other data table that is used in matrix operations is called a destination table. If the table size of the destination table is 1, then the destination table is simply called the destination.

b) The result from instruction execution is stored in the destination table, except for the following instructions.

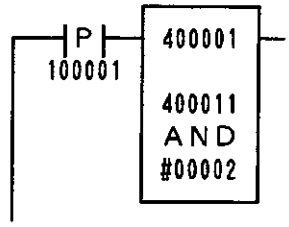
(1) LOGICAL COMPARE (CMPR)

(2) LOGICAL SENSE (SENS)

3) An example of the source and destination tables for the LOGICAL AND (AND) instruction is given next.

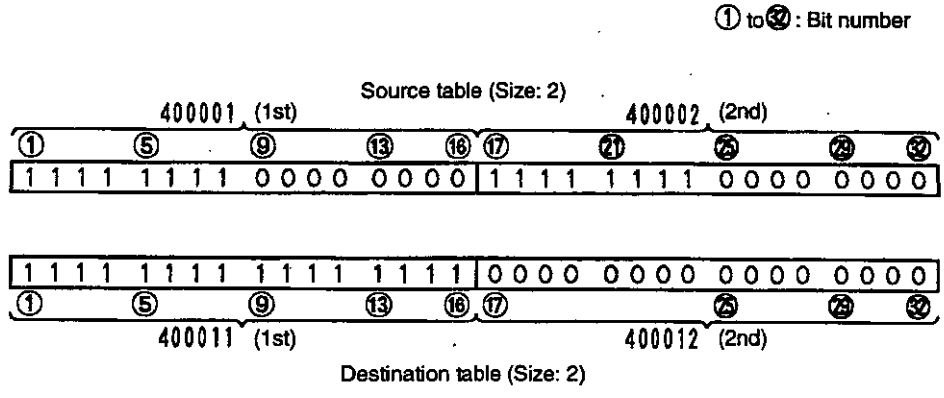
Example

1) Ladder Programming

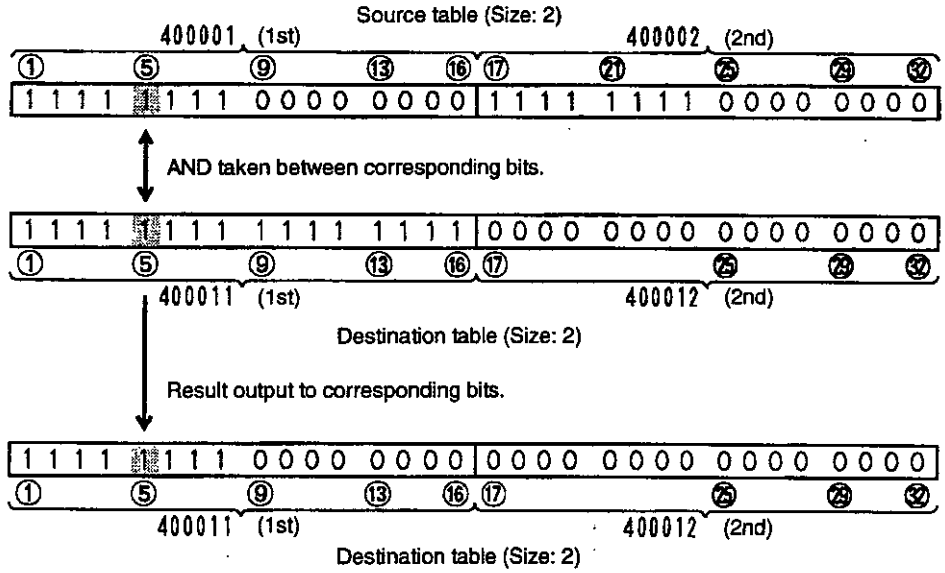


2) Operation

a) Before Execution



b) When input relay changes from OFF to ON, the following operation will be performed. The operation is completed in one scan.



(1) An AND is performed between corresponding bits of the source table and the destination table and the result is stored in the corresponding bits of the destination table.

(2) The data in the source table is not changed.

5.2.4 Pointers

1) Pointers are used for the following purposes in matrix instructions.

a) Storing Bit Numbers

In the following instructions, the pointer is used to store a bit number.

(1) LOGICAL COMPARE (CMPR)

(2) LOGICAL BIT MODIFY (MBIT)

(3) LOGICAL SENSE (SENS)

b) Storing the Number of Bits to Shift

In the LOGICAL MULTI-BIT ROTATE instruction (MROT), the pointer is used to store the number of bits to shift the destination table.

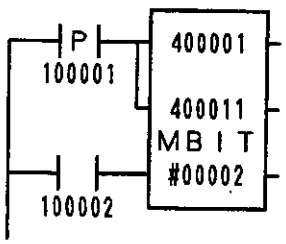
c) Storing the Number of Bits

In the LOGICAL BIT COUNT (BCNT) instruction, the pointer is used to store the number of bits that are 1 or 0.

2) The following example shows the use of the pointer in the LOGICAL BIT MODIFY (MBIT) instruction.

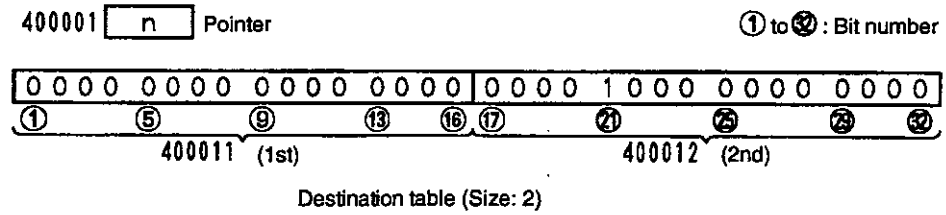
Example

1) Ladder Programming



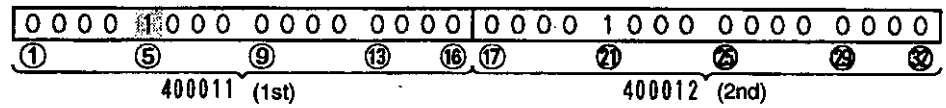
2) Operation

a) Before Execution



Destination table (Size: 2)

b) If the pointer value (n) is 5 and input relay 100001 changes from OFF to ON with the bit status shown above, the following operation will be performed. The operation is completed in one scan.



Destination table (Size: 2)

(1) The status of bit #5 in the destination table will be force-set to 1.

(2) The pointer value will be have as follows according to the status of input relay 100002:
If 100002 is ON, the pointer value will change to 6.
If 100002 is OFF, the pointer value will remain at 5.

(3) The status of any bit in the destination table can be force-set to 1 by changing the pointer value (n) to a value between 1 and 32.

5.3 Matrix Instructions

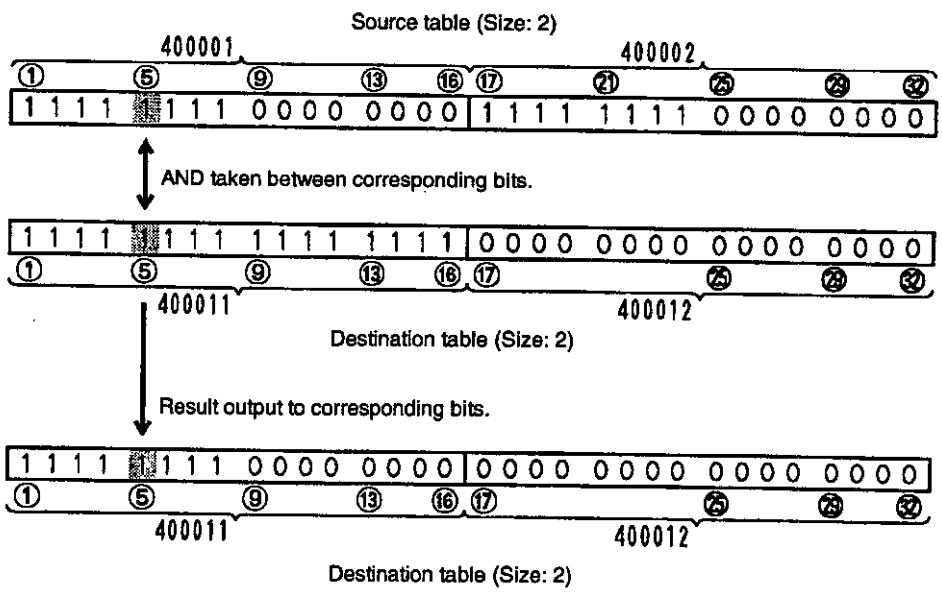
This section describes the functions, structures, and operation of the matrix instructions and provides simple examples of their application.

5.3.1 LOGICAL AND (AND)

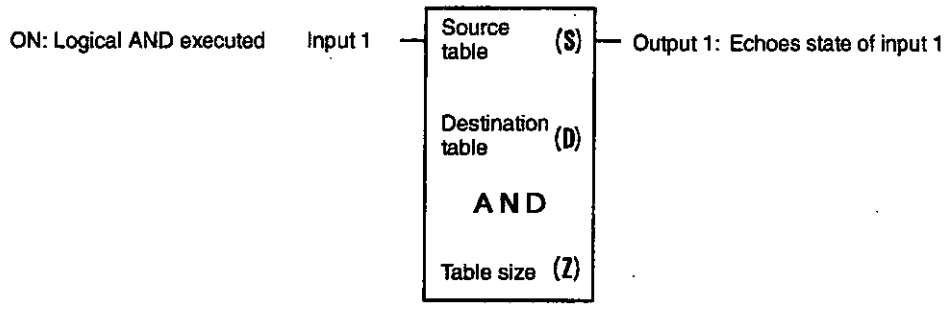
1. Function

A logical AND operation is performed between a source table and a destination table of the same size and the result is stored in the destination table. The operation is completed in one scan.

Example



2. Structure



1) AND is the symbol of LOGICAL AND.

2) AND requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 5.2* lists the register reference numbers and constants that can be specified.

Example

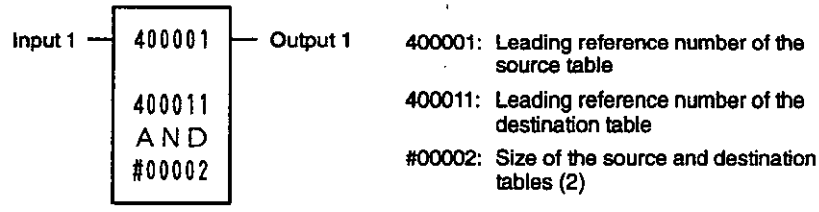


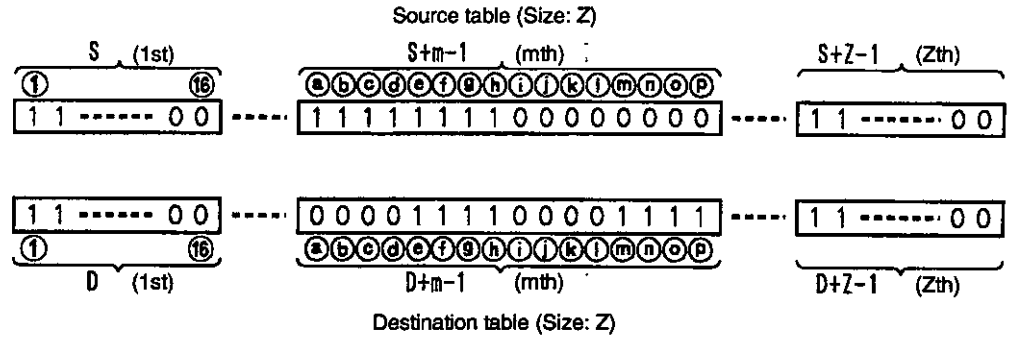
Table 5.2 Structural Elements of AND

Element	Meaning	Possible settings	
Top (S)	Leading reference number of the source table	Constant:	#00000 to #65535
		Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (D)	Leading reference number of the destination table	Coil:	000001 to 008161 (O00001 to O08161)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
Bottom (Z)	Size of the source and destination tables	Specify the constant. The maximum value differs with specified reference type.	
		Coil:	#00001 to #00100
		Input relay:	#00001 to #00064
		Constant, input register, holding register or constant register:	#00001 to #00100
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00100
		MC coil, MC relay or MC control relay:	#00001 to #00016
		MC control coil:	#00001 to #00010
		M code relay:	#00001 to #00006

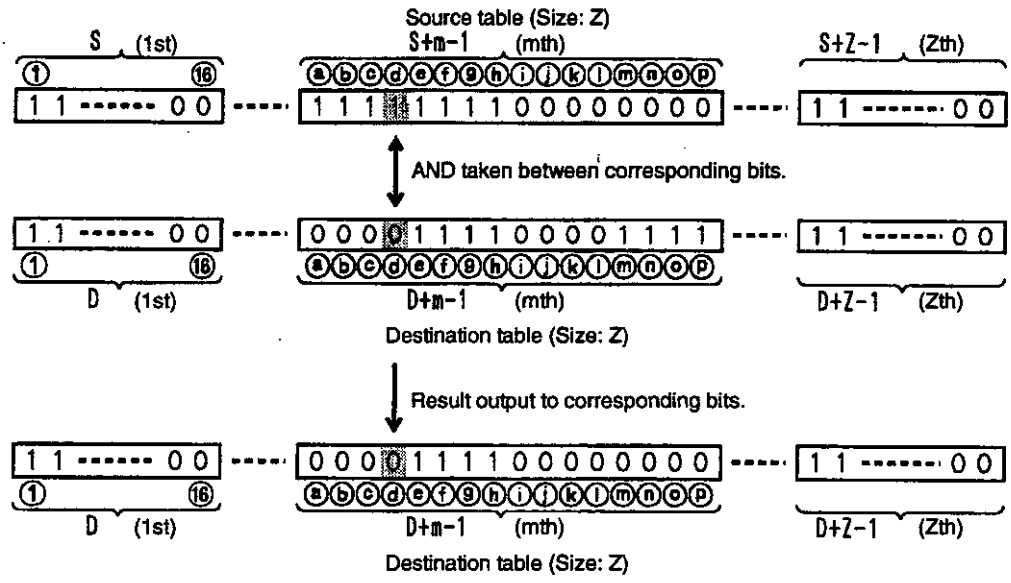
Note When specifying reference numbers for coils or relays, the following equation must be satisfied ("m" represents the lower 5 digits of the reference number):
m = 16n + 1, where n = 0, 1, 2, etc.

3. Operation

1) Before Execution



2) The following operation will be performed when input 1 is ON. The operation will be completed in one scan.



a) A logical AND is performed between corresponding bits of the source table and the destination table and the result is stored in the corresponding bits of the destination table.

b) The following truth table is used for the AND operation.

AND Truth Table

Before Execution		Result
Source Bit	**Destination Bit**	**Destination Bit**
1	0	0
1	1	1
0	1	0
0	0	0

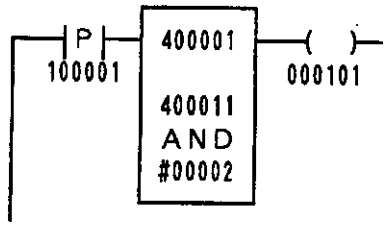
c) The data in the source table does not change.

d) Output 1 remains ON as long as input 1 is ON.

◀EXAMPLE▶

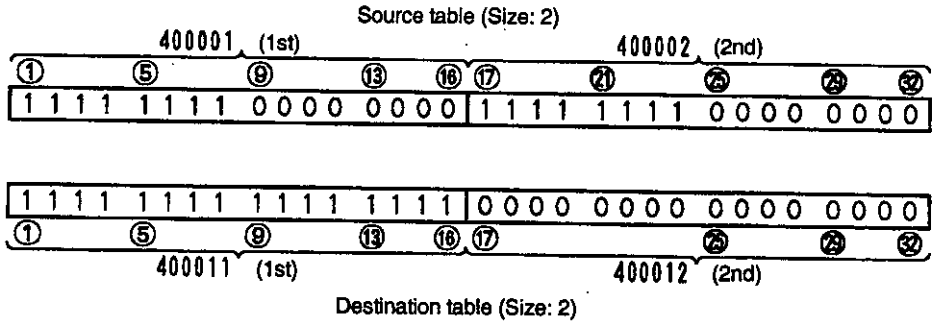
4. Application Example

1) Ladder Programming

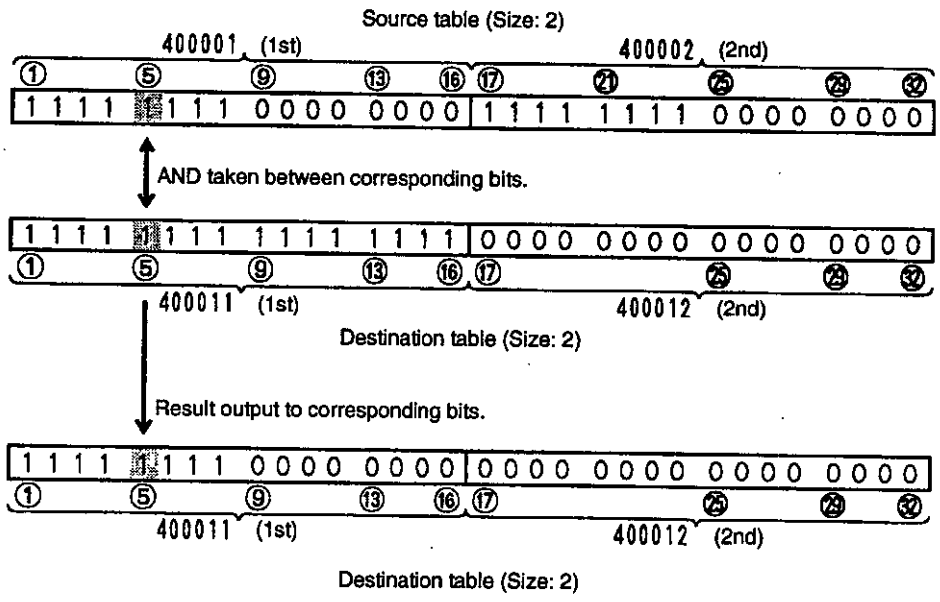


2) Operation

a) Before Execution



b) The following operation will be performed when input relay 100001 changes from OFF to ON. The operation will be completed in one scan.



(1) A logical AND is performed between corresponding bits of the source table and the destination table and the result is stored in the corresponding bits of the destination table.

(2) The data in the source table is not changed.

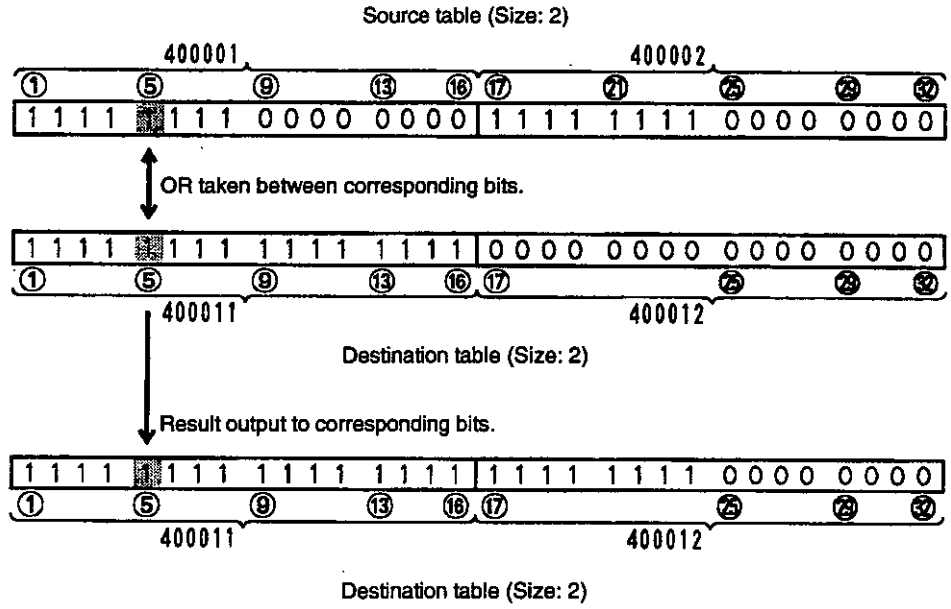
(3) Coil 000101 will be ON only in the scan in which input relay 100001 changed from OFF to ON.

5.3.2 LOGICAL OR (OR)

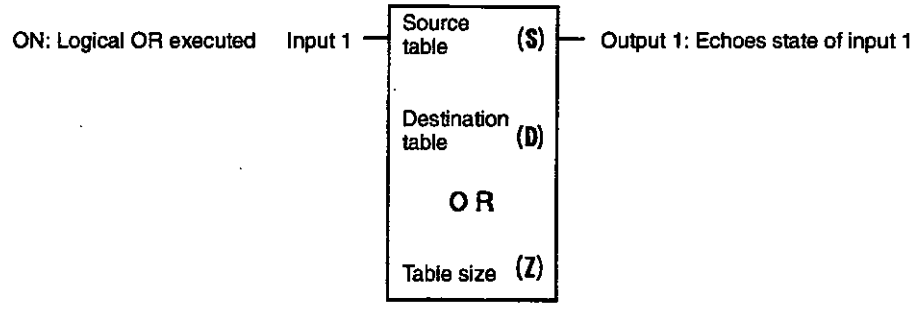
1. Function

A logical OR operation is performed between a source table and a destination table of the same size and the result is stored in the destination table. The operation is completed in one scan.

Example



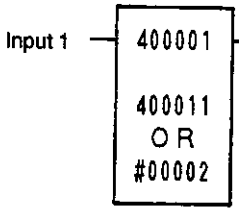
2. Structure



1) OR is the symbol of LOGICAL OR.

2) OR requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 5.3* lists the register reference numbers and constants that can be specified.

Example



400001: Leading reference number of the source table

400011: Leading reference number of the destination table

#00002: Size of the source and destination tables (2)

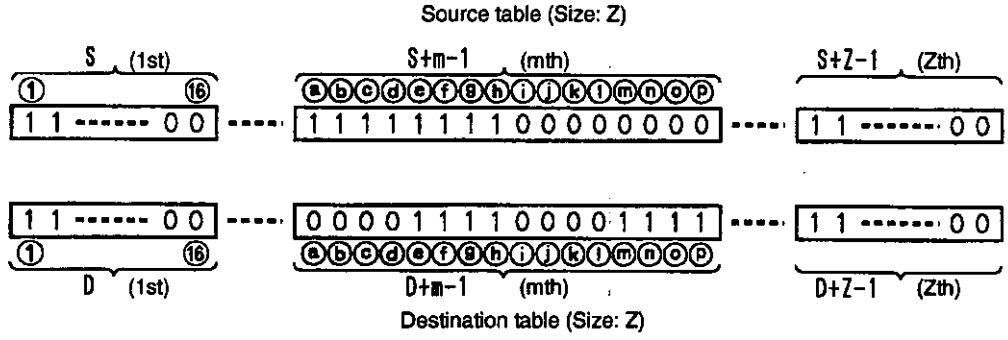
Table 5.3 Structural Elements of OR

Element	Meaning	Possible settings	
Top (S)	Leading reference number of the source table	Constant:	#00000 to #65535
		Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (D)	Leading reference number of the destination table	Coil:	000001 to 008161 (O00001 to O08161)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
Bottom (Z)	Size of the source and destination tables	Specify the constant. The maximum value differs with specified reference type.	
		Coil:	#00001 to #00100
		Input relay:	#00001 to #00064
		Constant, input register, holding register or constant register:	#00001 to #00100
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00100
		MC coil, MC relay or MC control relay:	#00001 to #00016
		MC control coil:	#00001 to #00010
		M code relay:	#00001 to #00006

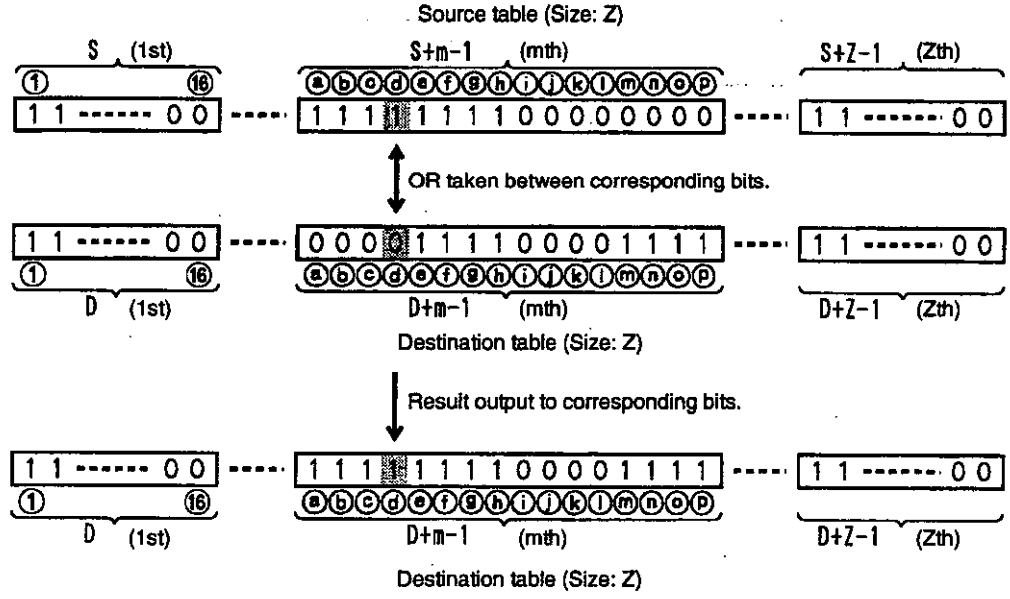
Note When specifying reference numbers for coils or relays, the following equation must be satisfied ("m" represents the lower 5 digits of the reference number):
m = 16n + 1, where n = 0, 1, 2, etc.

3. Operation

1) Before Execution



2) The following operation will be performed when input 1 is ON. The operation will be completed in one scan.



a) A logical OR is performed between corresponding bits of the source table and the destination table and the result is stored in the corresponding bits of the destination table.

b) The following truth table is used for the OR operation.

OR Truth Table

Before Execution		Result
Source Bit	**Destination Bit**	**Destination Bit**
1	0	1
1	1	1
0	1	1
0	0	0

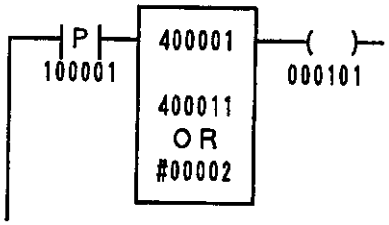
c) The data in the source table does not change.

d) Output 1 remains ON as long as input 1 is ON.

◀EXAMPLE▶

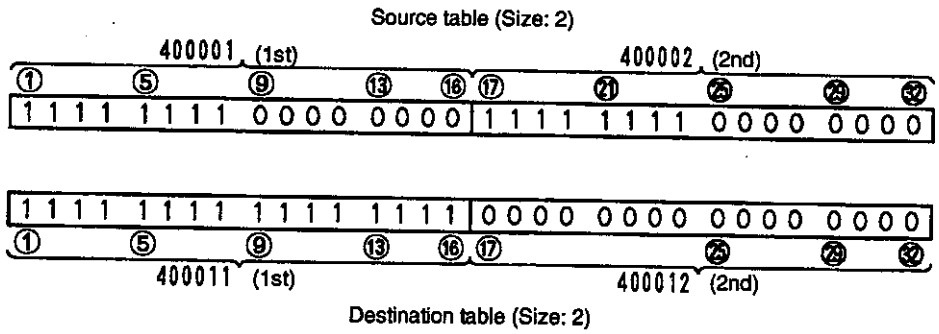
4. Application Example

1) Ladder Programming

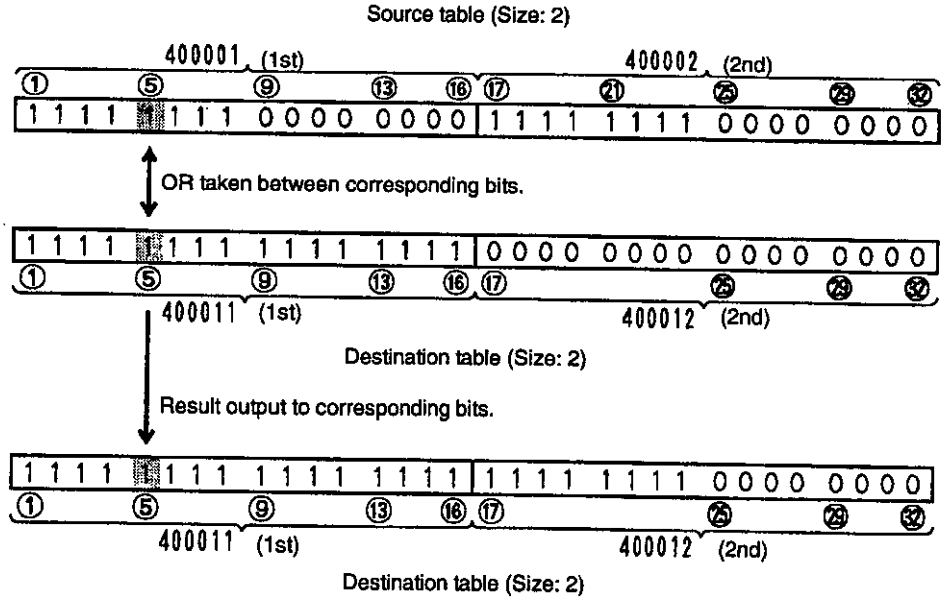


2) Operation

a) Before Execution



b) The following operation will be performed when input relay 100001 changes from OFF to ON. The operation will be completed in one scan.



(1) A logical OR is performed between corresponding bits of the source table and the destination table and the result is stored in the corresponding bits of the destination table.

(2) The data in the source table is not changed.

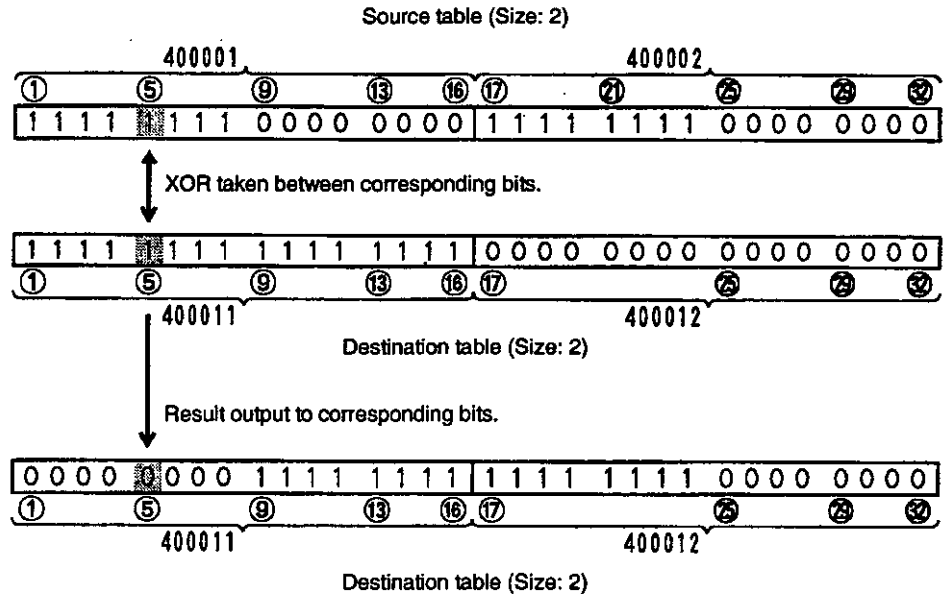
(3) Coil 000101 will be ON only in the scan in which input relay 100001 changed from OFF to ON.

5.3.3 LOGICAL EXCLUSIVE OR (XOR)

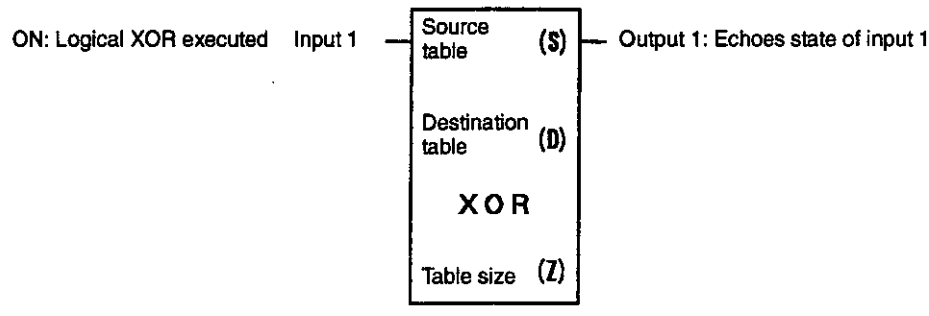
1. Function

A LOGICAL EXCLUSIVE OR operation is performed between a source table and a destination table of the same size and the result is stored in the destination table. The operation is completed in one scan.

Example



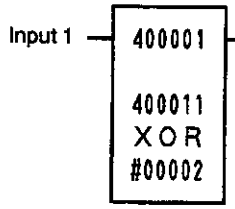
2. Structure



1) XOR is the symbol of LOGICAL EXCLUSIVE OR.

2) XOR requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 5.4* lists the register reference numbers and constants that can be specified.

Example



400001: Leading reference number of the source table

400011: Leading reference number of the destination table

#00002: Size of the source and destination tables (2)

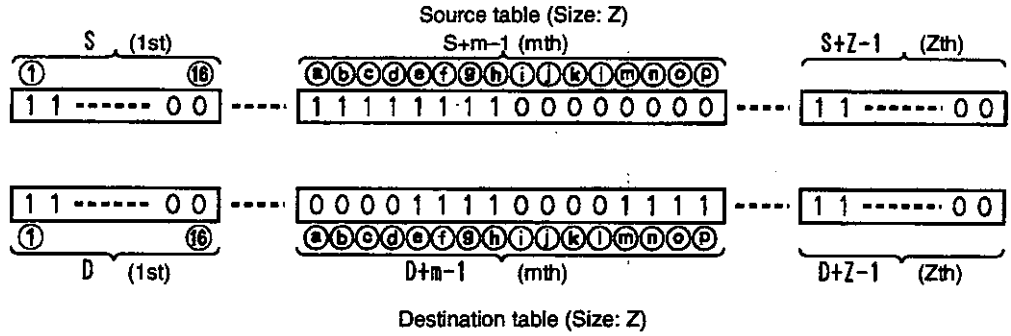
Table 5.4 Structural Elements of XOR

Element	Meaning	Possible settings	
Top (S)	Leading reference number of the source table	Constant:	#00000 to #65535
		Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (D)	Leading reference number of the destination table	Coil:	000001 to 008161 (O00001 to O08161)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
Bottom (Z)	Size of the source and destination tables	Specify the constant. The maximum value of the constant differs with specified reference type.	
		Coil:	#00001 to #00100
		Input relay:	#00001 to #00064
		Constant, input register, holding register or constant register:	#00001 to #00100
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00100
		MC coil, MC relay or MC control relay:	#00001 to #00016
		MC control coil:	#00001 to #00010
		M code relay:	#00001 to #00006

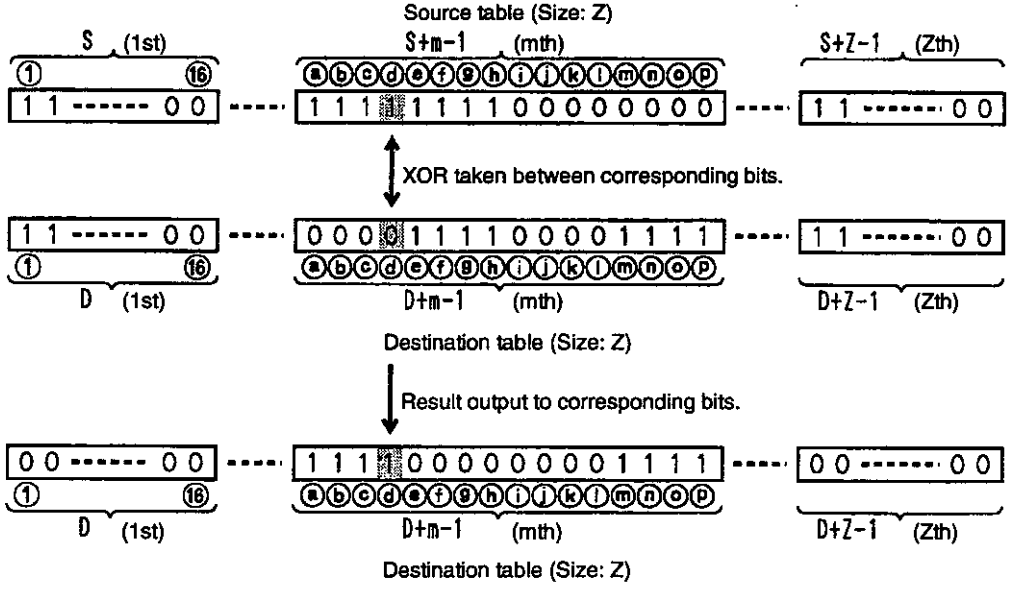
Note When specifying reference numbers for coils or relays, the following equation must be satisfied ("m" represents the lower 5 digits of the reference number):
m = 16n + 1, where n = 0, 1, 2, etc.

3. Operation

1) Before Execution



2) The following operation will be performed when input 1 is ON. The operation will be completed in one scan.



a) A LOGICAL EXCLUSIVE OR is performed between corresponding bits of the source table and the destination table and the result is stored in the corresponding bits of the destination table.

b) The following truth table is used for the XOR operation.

XOR Truth Table

Before Execution		Result
Source Bit	Destination Bit	Destination Bit
1	0	1
1	1	0
0	1	1
0	0	0

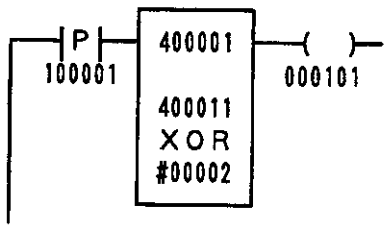
c) The data in the source table does not change.

d) Output 1 remains ON as long as input 1 is ON.

◀EXAMPLE▶

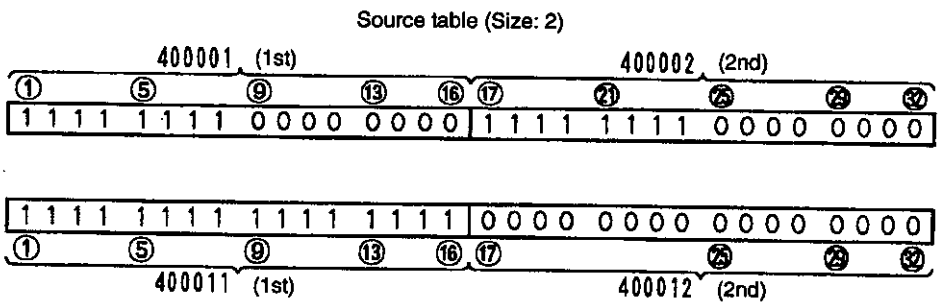
4. Application Example

1) Ladder Programming

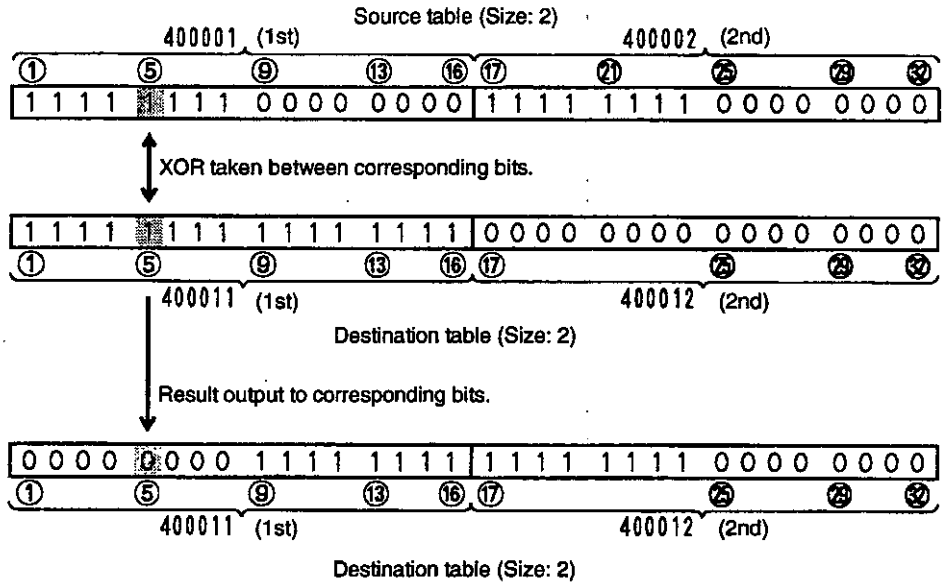


2) Operation

a) Before Execution



b) The following operation will be performed when input relay 100001 changes from OFF to ON. The operation will be completed in one scan.



(1) A LOGICAL EXCLUSIVE OR is performed between corresponding bits of the source table and the destination table and the result is stored in the corresponding bits of the destination table.

(2) The data in the source table is not changed.

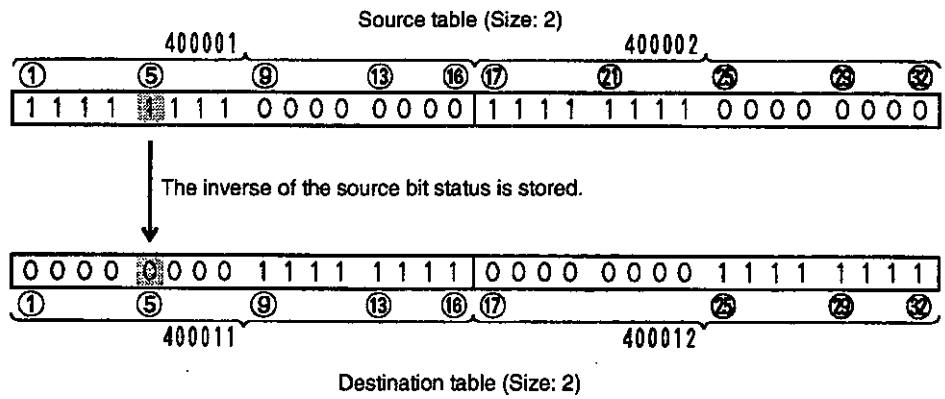
(3) Coil 000101 will be ON only in the scan in which input relay 100001 changed from OFF to ON.

5.3.4 LOGICAL COMPLEMENT (COMP)

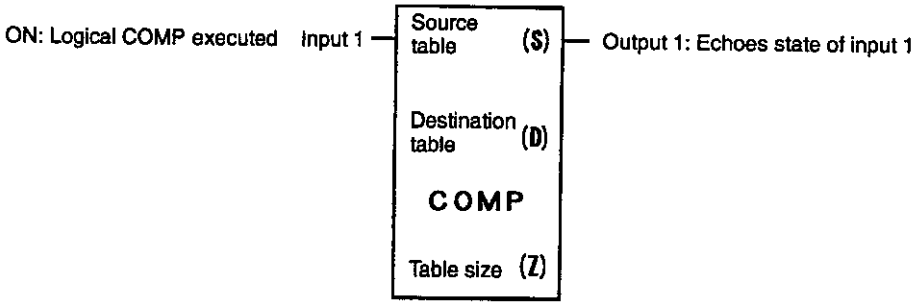
1. Function

The inverse of the status of each bit of the source table is stored in a destination table of the same size. The operation is completed in one scan.

Example



2. Structure



1) COMP is the symbol of LOGICAL COMPLEMENT.

2) COMP requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 5.5* lists the register reference numbers and constants that can be specified.

Example

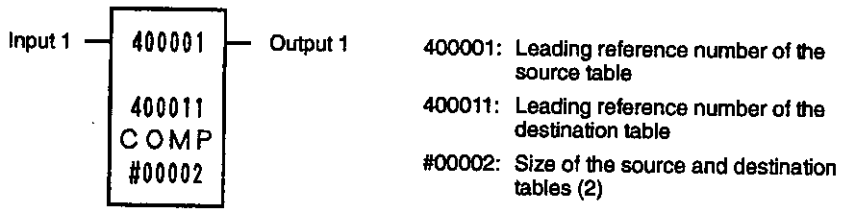


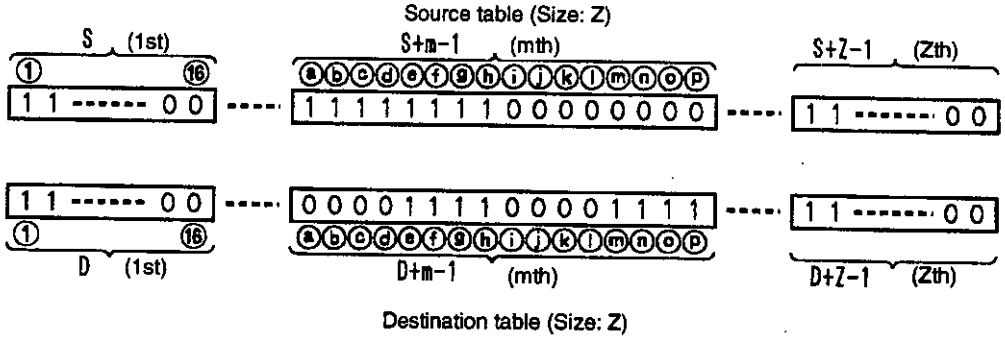
Table 5.5 Structural Elements of COMP

Element	Meaning	Possible settings	
Top (S)	Leading reference number of the source table	Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (D)	Leading reference number of the destination table	Coil:	000001 to 008161 (O00001 to O08161)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
Bottom (Z)	Size of the source and destination tables	Specify the constant. The maximum value of the constant differs with specified reference type.	
		Coil:	#00001 to #00100
		Input relay:	#00001 to #00064
		Input register, holding register or constant register:	#00001 to #00100
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00100
		MC coil, MC relay or MC control relay:	#00001 to #00016
		MC control coil:	#00001 to #00010
		M code relay:	#00001 to #00006

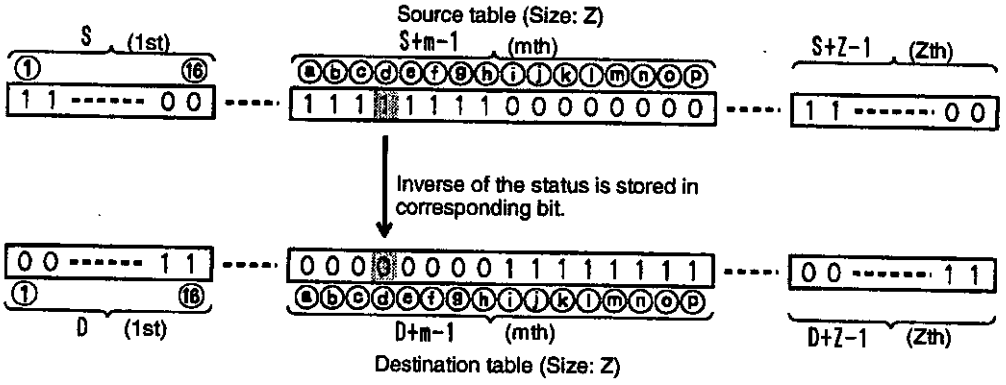
Note When specifying reference numbers for coils or relays, the following equation must be satisfied ("m" represents the lower 5 digits of the reference number):
m = 16n + 1, where n = 0, 1, 2, etc.

3. Operation

1) Before Execution



2) The following operation will be performed when input 1 is ON. The operation will be completed in one scan.



a) The inverse of the status of each bit of the source table is stored in the destination table.

b) The following table shows the operation of COMP.

Table 5.6 Operation of COMP

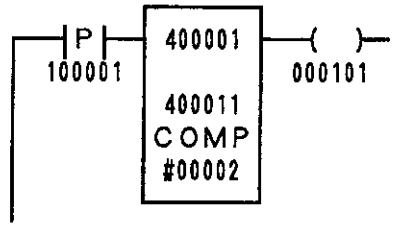
Before Execution		Result
Source Bit	Destination Bit	Destination Bit
1	Any	0
0	Any	1

c) The data in the source table does not change.

d) Output 1 remains ON as long as input 1 is ON.

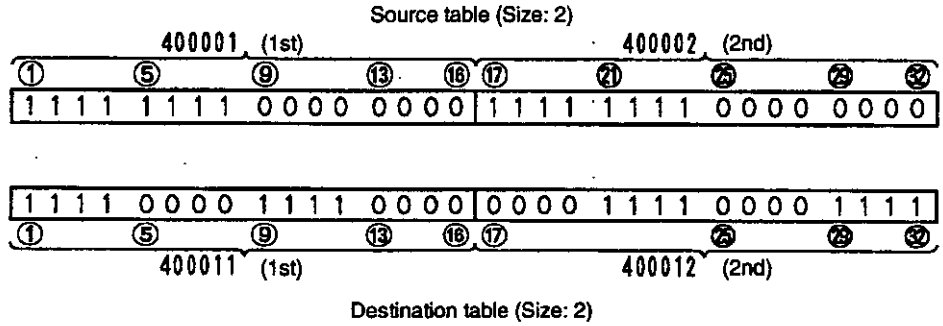
◀EXAMPLE▶ **4. Application Example**

1) Ladder Programming

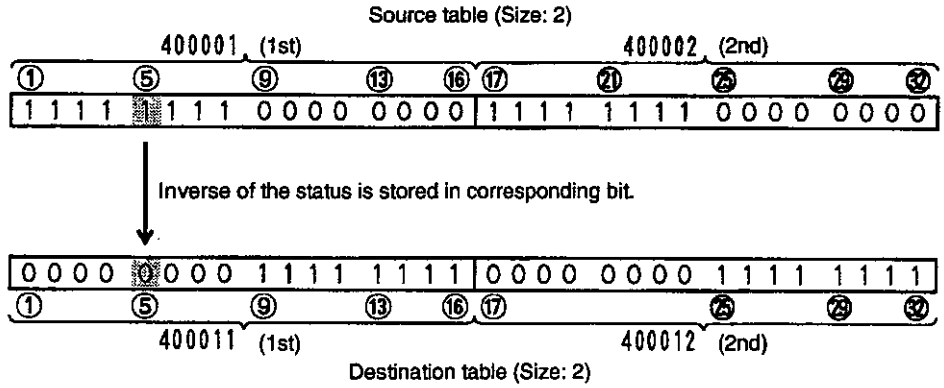


2) Operation

a) Before Execution



b) The following operation will be performed when input relay 100001 changes from OFF to ON. The operation will be completed in one scan.



(1) The inverse of the status of each bit of the source table is stored in the destination table.

(2) The data in the source table is not changed.

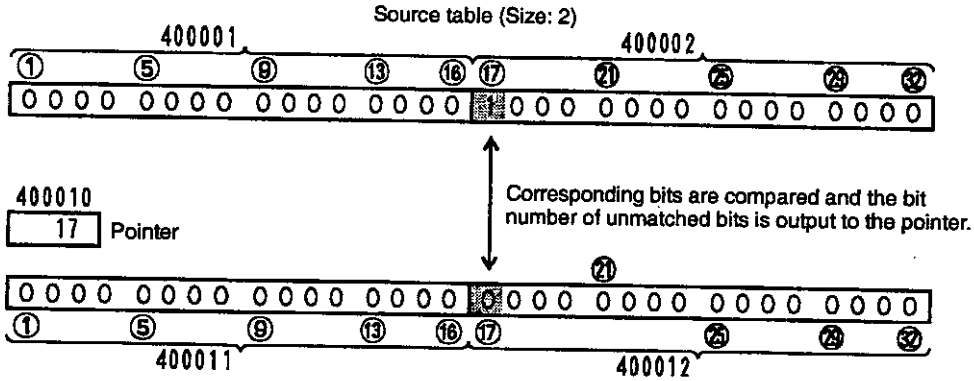
(3) Coil 000101 will be ON only in the scan in which input relay 100001 changed from OFF to ON.

5.3.5 LOGICAL COMPARE (CMPR)

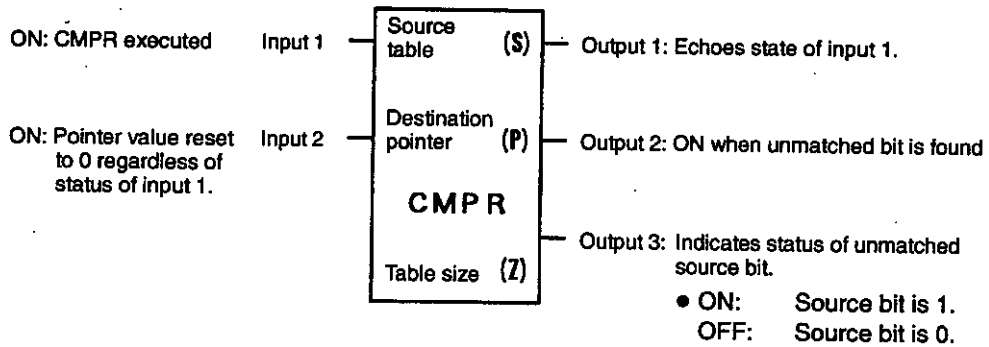
1. Function

The bit patterns of a source table and a destination table of the same size are compared one bit at a time.

Example



2. Structure



1) CMPR is the symbol of LOGICAL COMPARE.

2) CMPR requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 5.7* lists the register reference numbers and constants that can be specified. The next register after the destination pointer is the leading register of the destination table.

Example

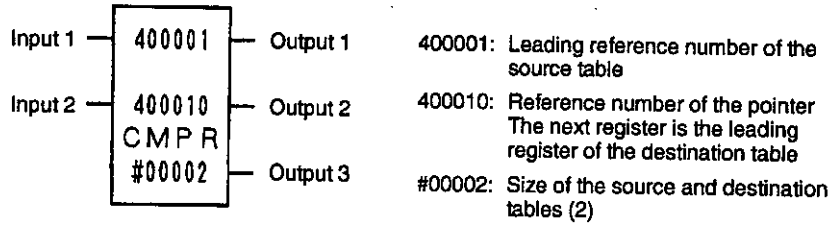


Table 5.7 Structural Elements of CMPR

Element	Meaning	Possible settings	
Top (S)	Leading reference number of the source table	Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (P)	Reference number of pointer	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 or R20001 to R21023
Bottom (Z)	Size of the source and destination tables	Specify the constant. The maximum value of the constant differs with specified reference type.	
		Coil:	#00001 to #00100
		Input relay:	#00001 to #00064
		Input register, holding register or constant register:	#00001 to #00100
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00100
		MC coil, MC relay or MC control relay:	#00001 to #00016
		MC control coil:	#00001 to #00010
		M code relay:	#00001 to #00006

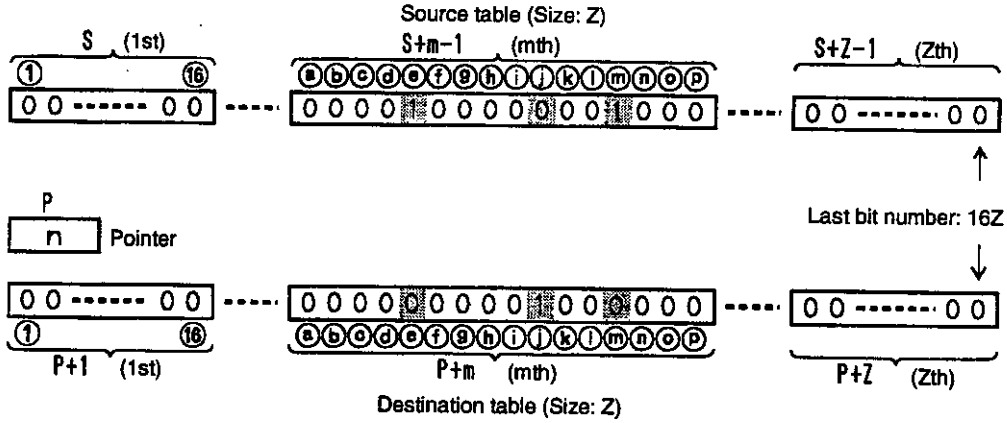
Note (1) When specifying reference numbers for coils or relays, the following equation must be satisfied ("m" represents the lower 5 digits of the reference number):
$m = 16n + 1$, where $n = 0, 1, 2$, etc.

(2) The next register after the pointer is the leading register of the destination table.

3. Operation

1) Before Execution

The following description assumes that the register contents are as shown below, i.e., there are unmatched bits at the positions labeled e, j, and m.



2) If input 2 is OFF, the following processing will take place.

When input 1 changes from OFF to ON, bit comparison starts from the bit position one higher (n+1) than the current pointer value (n).

a) In the first scan that input 1 changes from OFF to ON, the pointer value (n) will be 0 and the following processing will take place. This processing will be completed in one scan.

(1) The bit patterns of the source table and destination table are compared one bit at a time. The pointer value is 0, so comparison starts at bit #1.

(2) An unmatched bit is found at the position labeled e, so the bit number of this position is stored in the pointer and bit comparison ends.

(3) The data in the source and destination tables does not change.

(4) Outputs 1 and 2 will turn ON. Output 3 will also turn ON to indicate that the status of the unmatched source bit is 1.

b) In the second scan that input 1 changes from OFF to ON, the following processing will take place. This processing will be completed in one scan.

(1) The bit patterns of the source table and destination table are compared one bit at a time. The pointer value is at the bit number of the position labeled e, so comparison starts at the bit labeled f.

(2) An unmatched bit is found at the position labeled j, so the bit number of this position is stored in the pointer and bit comparison ends.

(3) The data in the source and destination tables does not change.

(4) Outputs 1 and 2 will turn ON. Output 3 will turn OFF to indicate that the status of the unmatched source bit is 0.

c) In the third scan that input 1 changes from OFF to ON, the following processing will take place. This processing will be completed in one scan.

(1) The bit patterns of the source table and destination table are compared one bit at a time. The pointer value is at the bit number of the position labeled j, so comparison starts at the bit labeled k.

(2) An unmatched bit is found at the position labeled m, so the bit number of this position is stored in the pointer and bit comparison ends.

(3) The data in the source and destination tables does not change.

(4) Outputs 1 and 2 will turn ON. Output 3 will turn OFF regardless of the 1 status of the unmatched source for the following reason.

Note Output 3 will not turn ON when the status of the source bit is 1 and the status of the destination bit is 0 if unmatched bits have already been found for which the status of the source bit was 0 and the status of the destination bit was 1.

d) In the fourth scan that input 1 changes from OFF to ON, the following processing will take place. This processing will be completed in one scan.

(1) The bit patterns of the source table and destination table are compared one bit at a time. The pointer value is at the bit number of the position labeled m, so comparison starts at the bit labeled n.

(2) Comparison continues to the last bit in the tables (16Z) and no more unmatched bits are found, so the pointer is reset to 0 and comparison ends.

(3) The data in the source and destination tables does not change.

(4) Output 1 will turn ON, and outputs 2 and 3 will turn OFF.

e) If input 1 turns ON again, the above processing from step a) to step d) will be repeated.

3) If input 2 is ON, the following processing will take place.

When input 1 changes from OFF to ON, bit comparison will always begin from bit #1. Processing will be as follows:

a) The pointer value (n) is set to 0.

b) The bit pattern of the source table and destination table are compared one bit at a time. The pointer value is 0, so comparison starts at bit #1.

c) An unmatched bit is found at the position labeled e, so the bit number of this position is stored in the pointer and bit comparison ends.

d) The data in the source and destination tables does not change.

e) Outputs 1 and 2 will turn ON. Output 3 will also turn ON to indicate that the status of the unmatched source bit is 1.

4) An overview of the operation of CMPR is given in the following table. n is the pointer value, Z is the size of the source and destination tables.

Table 5.8 Operation of CMPR

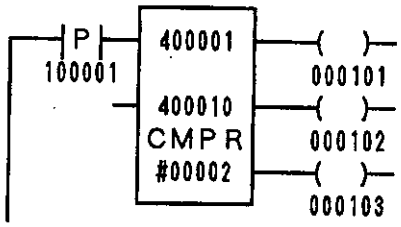
Inputs		Condition of n	Operation	Outputs		
1	2			1	2	3
ON	OFF	0≤n≤16Z–1	1) Bit patterns in source and destination table compared one bit at a time. 2) Bit comparison starts at n+1. 3) If unmatched bit is found, bit number is stored in pointer and bit comparison ends for current scan. 4) If no unmatched bits are found before 16Z, pointer is reset to 0 and bit comparison ends for current scan. 5) All unmatched bits can be found one at a time.	**ON**	**ON** when un-matched bit found	Status of un-matched source bit. **ON** when source bit is 1.
		n<0, n≥16Z	1) Bit patterns in source and destination table compared one bit at a time. 2) Bit comparison always starts from bit #1. 3) If unmatched bit is found, bit number is stored in pointer and bit comparison ends for current scan. 4) If no unmatched bits are found before 16Z, pointer is reset to 0 and bit comparison ends for current scan. 5) Only the first unmatched bit can be found even if more exist.			
	ON	None				
OFF	OFF	0≤n≤16Z–1	Bits are not compared.	OFF	OFF	OFF
		n<0, n≥16Z	1) Pointer is reset to 0 2) Bits are not compared.			
	ON	None				

Note Output 3 will not turn ON when the status of the source bit is 1 and the status of the destination bit is 0 if unmatched bits have already been found for which the status of the source bit was 0 and the status of the destination bit was 1.

4. Application Examples

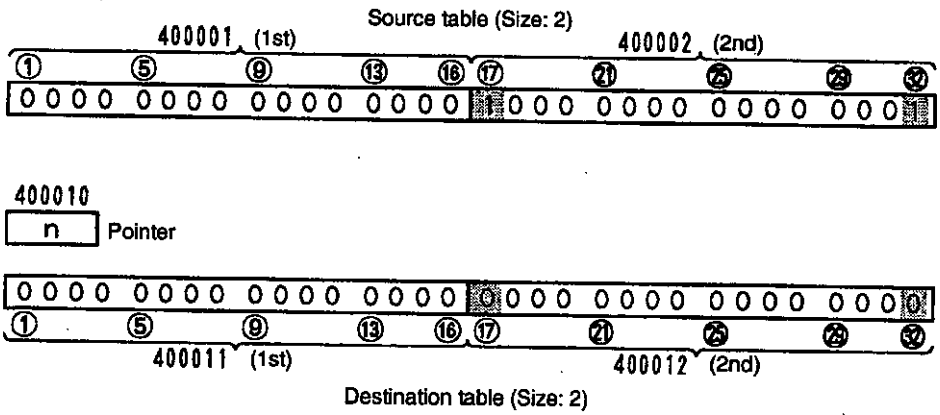
◀EXAMPLE▶ **Example 1**

1) Ladder Programming



2) Operation

a) Before Execution



b) The pointer value (n) is initially 0. When input relay 100001 changes from OFF to ON the following processing will take place. This processing will be completed in one scan.

(1) The bit patterns of the source table and destination table are compared one bit at a time. The pointer value is 0, so comparison starts at bit #1.

(2) An unmatched bit is found at the bit #17, so 17 is stored in the pointer and bit comparison ends.

(3) The data in the source and destination tables does not change.

(4) Coils 000101, 000102, and 000103 all turn ON.

c) When input relay 100001 changes from OFF to ON again, the following processing will take place. This processing will be completed in one scan.

(1) The bit patterns of the source table and destination table are compared one bit at a time. The pointer value is 17, so comparison starts at bit #18.

(2) An unmatched bit is found at bit #32, so 32 is stored in the pointer and bit comparison ends.

(3) The data in the source and destination tables does not change.

(4) Coils 000101, 000102, and 000103 all turn ON.

d) When input relay 100001 changes from OFF to ON again, the following processing will take place. This processing will be completed in one scan.

(1) The bit patterns of the source table and destination table are compared one bit at a time. The pointer value is 32, so the pointer is reset to 0 and comparison starts at bit #1.

(2) An unmatched bit is found at bit #17, so 17 is stored in the pointer and bit comparison ends.

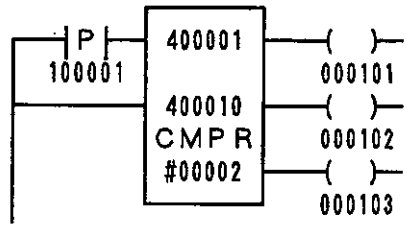
(3) The data in the source and destination tables does not change.

(4) Coils 000101, 000102, and 000103 all turn ON.

e) When input relay 100001 continues to change from OFF to ON, the above processing is repeated.

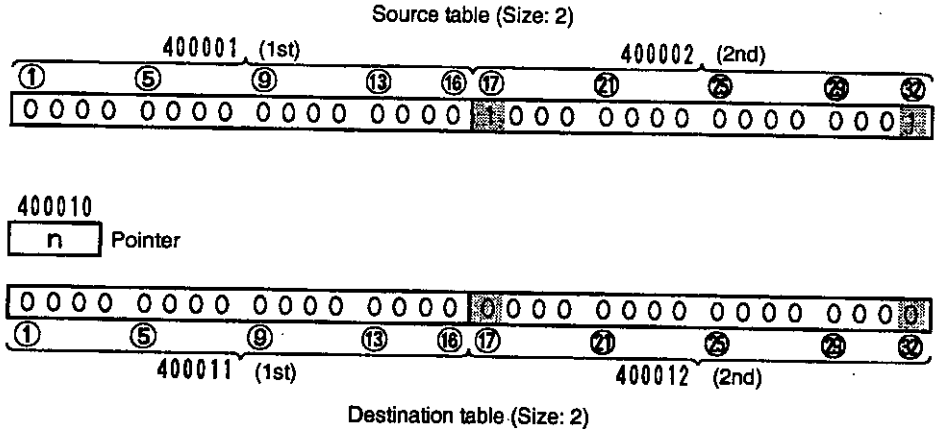
◀EXAMPLE▶ **Example 2**

1) Ladder Programming



2) Operation

a) Before Execution



b) When input relay 100001 changes from OFF to ON, the following processing will take place. This processing will be completed in one scan.

(1) The pointer value is reset to 0.

(2) The bit patterns of the source table and destination table are compared one bit at a time. The pointer value is 0, so comparison starts at bit #1.

(3) An unmatched bit is found at the bit #17, so 17 is stored in the pointer and bit comparison ends.

(4) The data in the source and destination tables does not change.

(5) Coils 000101, 000102, and 000103 all turn ON.

c) As shown in the example, input 2 is always ON, so the pointer value is always reset to 0 and comparison will always start from bit #1. Therefore, bit #17 will always be found as the unmatched bit, unless the contents of the registers change.

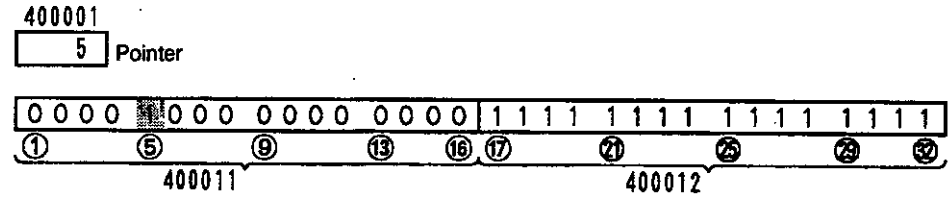
5.3.6 LOGICAL BIT MODIFY (MBIT)

1. Function

A pointer (P) is used to force the status of an arbitrary bit in the destination table (DT) to either 1 or 0. The operation is completed in one scan.

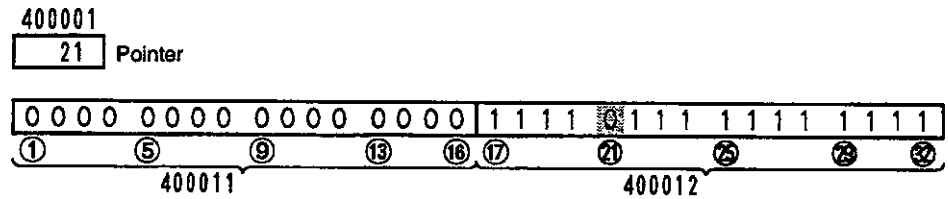
Example

1) If the pointer value is 5 and inputs 1 and 2 are both ON when LOGICAL BIT MODIFY is executed, then bit #5 of the destination table will be set to 1.



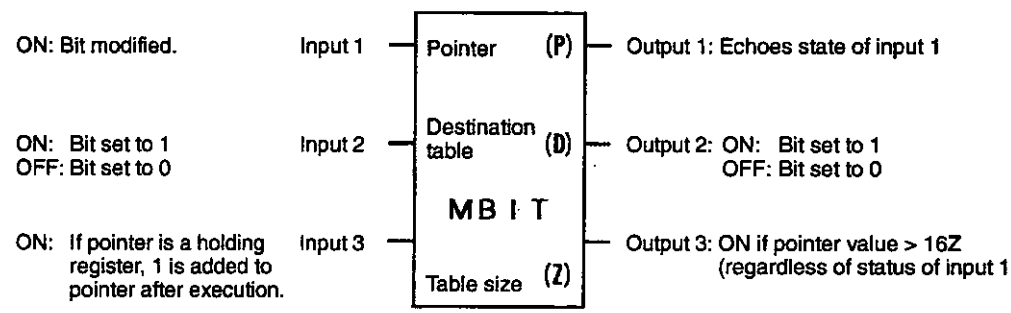
Destination table (Size: 2)

2) If the pointer value is 21 and only input 1 is ON when LOGICAL BIT MODIFY is executed, then bit #21 of the destination table will be set to 0.



Destination table (Size: 2)

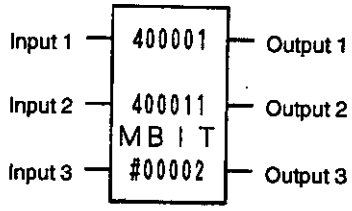
2. Structure



1) MBIT is the symbol of LOGICAL BIT MODIFY.

2) MBIT requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 5.9* lists the register reference numbers and constants that can be specified.

Example



400001: Reference number of pointer
400011: Leading reference number of the destination table
#00002: Size of destination table (2)

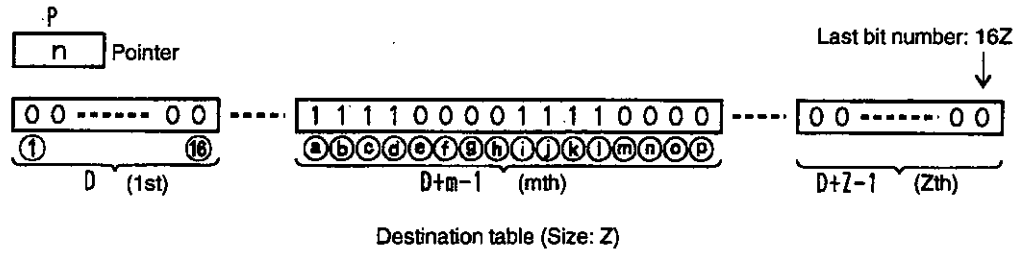
Table 5.9 Structural Elements of MBIT

Element	Meaning	Possible settings	
Top (P)	Reference number of pointer	Constant:	#00001 to #09600
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 or R20001 to R21024
Middle (D)	Leading reference number of the destination table	Coil:	000001 to 008161 (O00001 to O08161)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
Bottom (Z)	Size of destination table	Specify the constant. The maximum value of the constant differs with reference type specified in middle element.	
		Coil:	#00001 to #00512
		Holding register:	#00001 to #00600
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00600
		MC coil:	#00001 to #00016
		MC control coil:	#00001 to #00010

Note When specifying reference numbers for coils or relays, the following equation must be satisfied ("m" represents the lower 5 digits of the reference number):
m = 16n + 1, where n = 0, 1, 2, etc.

3. Operation

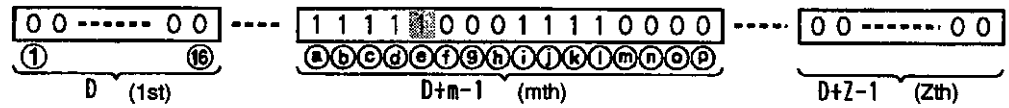
1) Before Execution



2) If input 2 is ON, the following processing will take place.

The following operation is performed when input 1 is ON. This operation is completed in one scan.

a) If the pointer value (n) is e, the the bit labeled e in the destination table is set to 1.



b) The pointer value (e) is incremented if the following two conditions are met.

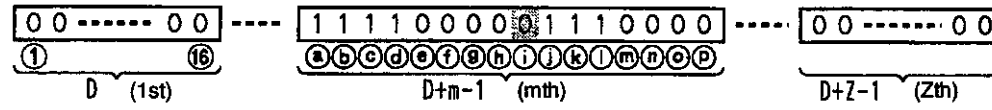
- The pointer is a holding or link register.
- Input 3 is ON.

c) Outputs 1 and 2 turn ON and output 3 remains OFF.

3) If input 2 is OFF, the following processing will take place.

The following operation is performed when input 1 is ON. This operation is completed in one scan.

a) If the pointer value (n) is i, the the bit labeled i in the destination table is set to 0.



b) The pointer value (i) is incremented if the following two conditions are met.

- The pointer is a holding register.
- Input 3 is ON.

c) Output 1 turns ON and outputs 2 and 3 remain OFF.

4) If the pointer value (n) is less then 0 or greater than 16Z (highest bit number), the the following operation is performed regardless of the status of input 1.

a) The instruction is not executed.

b) The pointer value and the data in the destination table do not change.

c) Outputs 1 and 3 turn ON and output 2 remains OFF.

5) An overview of the operation of MBIT is shown in the following table. Pointer value is n and the size of the destination table is Z.

Table 5.10 Operation of MBIT

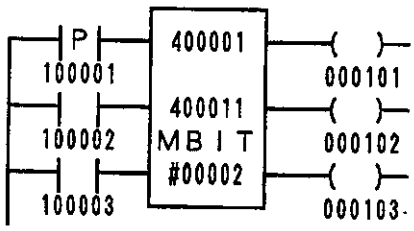
Inputs			Condition of n	Operation	Outputs		
1	2	3			1	2	3
ON	ON	OFF	1 ≤ n ≤ 16Z	1) Bit #n in destination table set to 1. 2) Value of n does not change.	ON	ON	OFF
		ON		1) Bit #n in destination table set to 1. 2) Pointer incremented to n+1 if it is a register.			See note.
	OFF	OFF		1) Bit #n in destination table set to 0. 2) Value of n does not change.		OFF	OFF
		ON		1) Bit #n in destination table set to 0. 2) Pointer incremented to n+1 if it is a register.			See note.
	Any	Any	n = 0	1) Nothing is done. 2) Pointer value and data in destination table do not change.			OFF
			n < 0, n > 16Z				ON
OFF	Any	Any	1 ≤ n ≤ 16Z		OFF		OFF
			n ≤ 0, n > 16Z				ON

Note Output 3 turns ON if the pointer is incremented to 16Z+1.

◀EXAMPLE▶

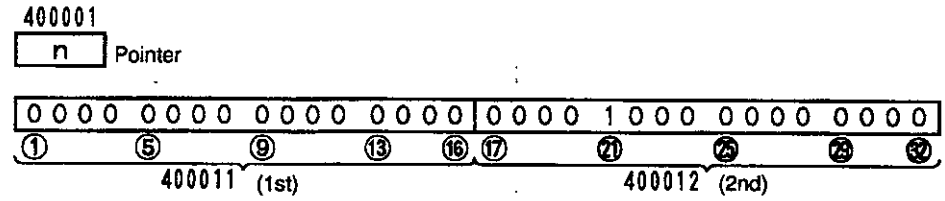
4. Application Example

1) Ladder Programming



2) Operation

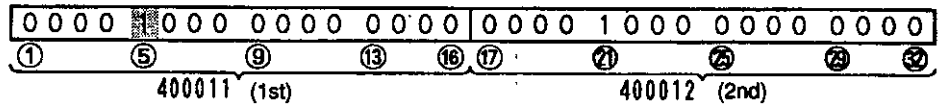
a) Before Execution



Destination table (Size: 2)

b) If input relay 100002 is ON, the following processing will take place.

If the pointer value is 5 when input 1 changes from OFF to ON, the following operation is performed. This operation will be completed in one scan.



Destination table (Size: 2)

(1) Bit #5 of the destination table is set to 1.

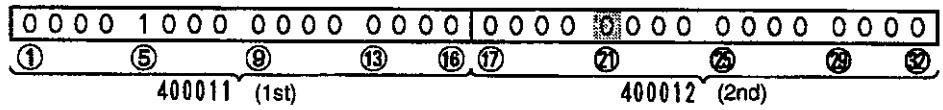
(2) The pointer value will be as follows depending on the status of input relay 100003:
100003 ON: Pointer incremented to 6.
100003 OFF: Pointer remains at 5.

(3) Coils 000101 and 000102 turn ON and coil 000103 remains OFF.

(4) The pointer value (n) can be varied between 1 and 32 to the desired bit in the destination table to 1.

c) If input relay 100002 is OFF, the following processing will take place.

If the pointer value is 21 when input 1 changes from OFF to ON, the following operation is performed. This operation will be completed in one scan.



Destination table (Size: 2)

(1) Bit #21 of the destination table is set to 0.

(2) The pointer value will be as follows depending on the status of input relay 100003:
100003 ON: Pointer incremented to 22.
100003 OFF: Pointer remains at 21.

(3) Coil 000101 turns ON and coils 000102 and 000103 remain OFF.

(4) The pointer value (n) can be varied between 1 and 32 to set the desired bit in the destination table to 0.

d) If the pointer value is for a bit number that does not exist (i.e., less than 1 or greater than 32), the following processing will take place regardless of the status of input relay 100001.

(1) The instruction is not executed.

(2) The pointer value and the data in the destination table do not change.

(3) The status of the output coils are as follows:
Coil 000101: Same status as input relay 100001
Coil 000102: OFF
Coil 000103: ON (regardless of the status of input relays 100001 and 100002)

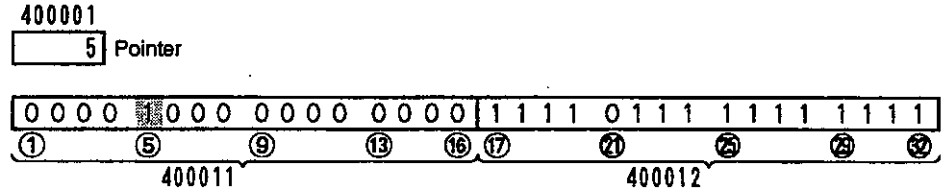
5.3.7 LOGICAL SENSE (SENS)

1. Function

A pointer (P) is used to read the status of an arbitrary bit in the destination table (DT). The operation is completed in one scan.

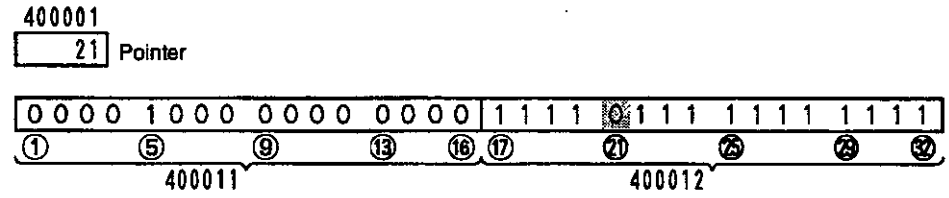
Example

1) If the pointer value is 5 and input 1 is ON when LOGICAL SENSE is executed, the status of bit #5 in the destination table will be read and output 2 will turn ON to indicate that the bit status is 1.



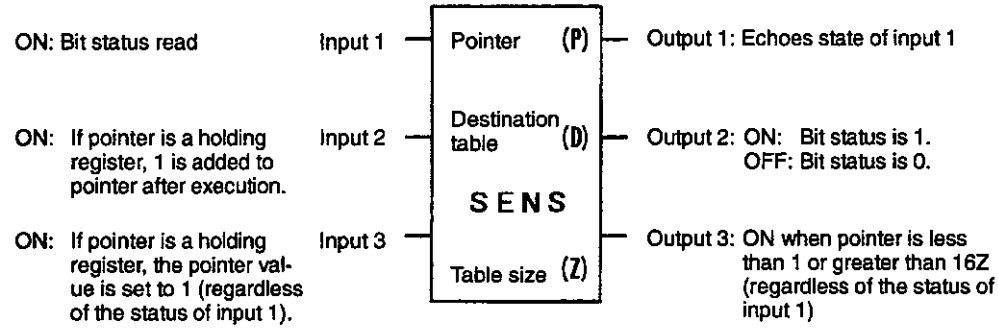
Destination table (Size: 2)

2) If the pointer value is 21 and input 1 is ON when LOGICAL SENSE is executed, the status of bit #21 in the destination table will be read and output 2 will turn OFF to indicate that the bit status is 0.



Destination table (Size: 2)

2. Structure



1) SENS is the symbol of LOGICAL SENSE.

2) SENS requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 5.11* lists the register reference numbers and constants that can be specified.

Example

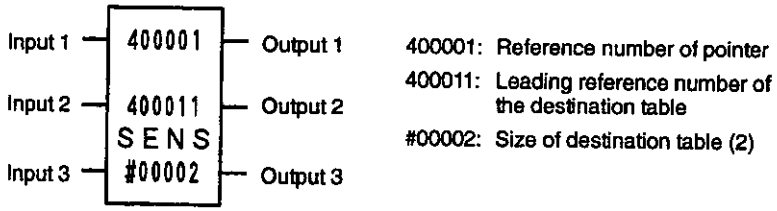


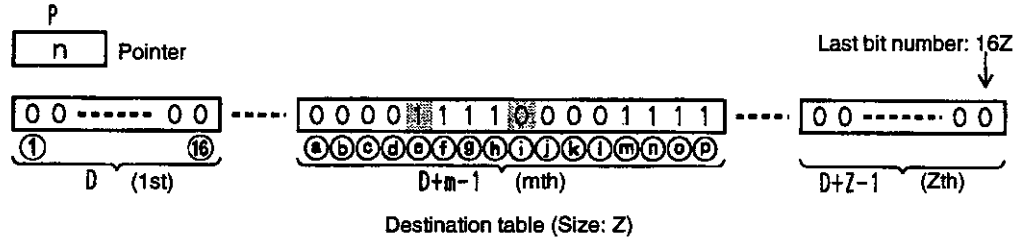
Table 5.11 Structural Elements of SENS

Element	Meaning	Possible settings	
Top (P)	Reference number of pointer	Constant:	#00001 to #09600
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 or R10001 to R11024
Middle (D)	Leading reference number of the destination table	Coil:	000001 to 008161 (O00001 to O08161)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Bottom (Z)	Size of destination table	Specify the constant. The maximum value of the constant differs with the reference type specified in middle element.	
		Coil:	#00001 to #00512
		Input relay:	#00001 to #00064
		Input register, holding register:	#00001 to #00600
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00600
		MC coil, MC relay or MC control relay:	#00001 to #00016
		MC control coil:	#00001 to #00010
		M code relay:	#00001 to #00006

Note When specifying reference numbers for coils or relays, the following equation must be satisfied ("m" represents the lower 5 digits of the reference number):
m = 16n + 1, where n = 0, 1, 2, etc.

3. Operation

1) Before Execution



2) The following processing will take place if the pointer value (n) is between 1 and 16Z.

The following operation is performed when input 1 is ON. This operation is completed in one scan.

a) If the pointer value (n) is e:

(1) The status of the bit at the position labeled e is read and the status is output to output 2. Here, the bit is 1, so output 2 turns ON.

(2) The pointer value (e) is incremented if the following two conditions are met.
The pointer is a holding or link register.
Input 2 is ON.

(3) Here, the pointer value does not change. The data in the destination table is unaltered.

(4) Output 1 turns ON and output 3 remains OFF.

b) If the pointer value (n) is i:

(1) The status of the bit at the position labeled i is read and the status is output to output 2. Here, the bit is 0, so output 2 turns OFF.

(2) The pointer value (i) is incremented if the following two conditions are met.
The pointer is a holding register.
Input 2 is ON.

(3) Here, the pointer value does not change. The data in the destination table is unaltered.

(4) Output 1 turns ON and output 3 remains OFF.

c) The pointer value (n) can be changed between 1 and 16Z (the highest bit number) to read the status of the desired bit in the destination table.

d) If the pointer is a holding register and its value is 16Z (last bit number) when inputs 1 and 2 are ON, the pointer will be set to 1 after execution of the instruction (not to 16Z+1).

e) If the pointer value is less than 1 or greater than 16Z, the following operation will be performed regardless of the status of input 1.

(1) The instruction is not executed.

(2) The pointer value and the data in the destination table do not change.

(3) The status of the outputs are as follows:
Output 1: ON
Output 2: OFF
Output 3: OFF if n = 0, ON if n < 0 or n ≥ 16Z + 1

3) If the pointer is a link register or a holding register and input 3 is ON, the pointer value will be set to 1 regardless of the status of input 1.

4) An overview of the operation of SENS is shown in the following table. n is the pointer value and Z is the size of the destination table.

Table 5.12 Operation of SENS

Inputs			Condition of n	Operation	Outputs		
1	2	3			1	2	3
ON	OFF	OFF	$1 \leq n \leq 16Z$	1) Status of bit #n in destination table read and output to output 2. 2) Value of n not changed. 3) Data in destination table not changed.	**ON**	See note.	OFF
	ON			1) Status of bit #n in destination table read and output to output 2. 2) If pointer is a holding or link register, n is incremented to n+1. Otherwise n is not changed. 3) Data in destination table not changed.			
	Any		$n \leq 0$, $n > 16Z+1$	1) Instruction is not executed. 2) Pointer value and data in destination table not changed.		OFF	**ON** if $n \neq 0$
	OFF	**ON**	None	1) Pointer set to 1 and the status of bit #1 in destination table read and output to output 2. 2) Value of n not changed. 3) Data in destination table not changed.		See note.	OFF
	ON			1) Pointer set to 1 and the status of bit #1 in destination table read and output to output 2. 2) If pointer is a holding or link register, n is incremented to 2. Otherwise n is not changed. 3) Data in destination table not changed.			

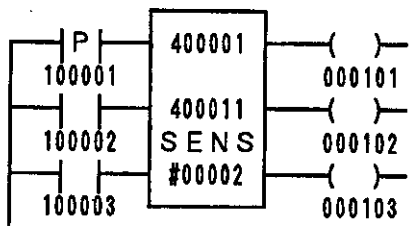
Note ON if status of the bit which has been read is 1.

Inputs			Condition of n	Operation	Outputs		
1	2	3			1	2	3
OFF	Any	OFF	$1 \le n \le 16Z$	1) Instruction is not executed.	OFF	OFF	OFF
			$n \le 0$, $n \ge 16Z+1$	2) Pointer value and data in destination table not changed.			**ON**
		ON	$1 \le n \le 16Z$	1) Instruction is not executed.			OFF
			$n \le 0$, $n \ge 16Z+1$	2) If pointer is a holding or link register, n is set to 1. Otherwise n is not changed. 3) Data in destination table not changed.			**ON** if $n \le 0$ or $n \ge 16Z+1$

◀EXAMPLE▶

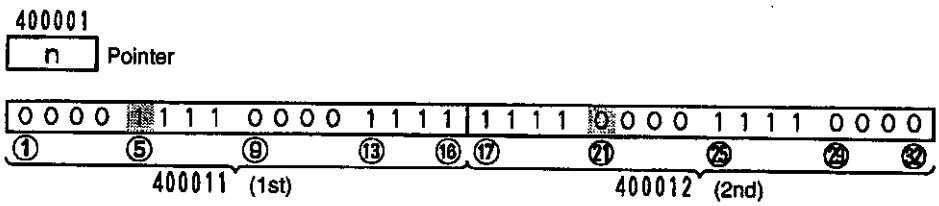
4. Application Example

1) Ladder Programming



2) Operation

a) Before Execution



Destination table (Size: 2)

b) If the pointer value (n) is 5 when input relay 100001 changes from OFF to ON, the following processing will take place. The processing will be completed in one scan.

(1) The status of bit #5 is read and the status is output to coil 000102. Here, the bit is 1, so coil 000102 turns ON.

(2) The pointer value is as follows depending on the status of input relay 100002:
100002 ON: Pointer incremented to 6.
100002 OFF: Pointer remains at 5.

(3) Coil 000101 turns ON and coil 000103 remains OFF.

c) If the pointer value (n) is 21 when input relay 100001 changes from OFF to ON, the following processing will take place. The processing will be completed in one scan.

(1) The status of bit #21 is read and the status is output to coil 000102. Here, the bit is 0, so coil 000102 turns OFF.

(2) The pointer value is as follows depending on the status of input relay 100002:
100002 ON: Pointer incremented to 22.
100002 OFF: Pointer remains at 21.

(3) Coil 000101 turns ON and coil 000103 remains OFF.

d) The pointer value (n) can be changed between 1 and 32 to read the status of any desired bit in the destination table.

e) If the pointer value is for a bit number that does not exist (i.e., less than 1 or greater than 32), the following processing will take place regardless of the status of input relay 100001.

(1) The instruction is not executed.

(2) The pointer value and the data in the destination table do not change.

(3) The status of the output coils are as follows:
Coil 000101: ON
Coil 000102: OFF
Coil 000103: ON

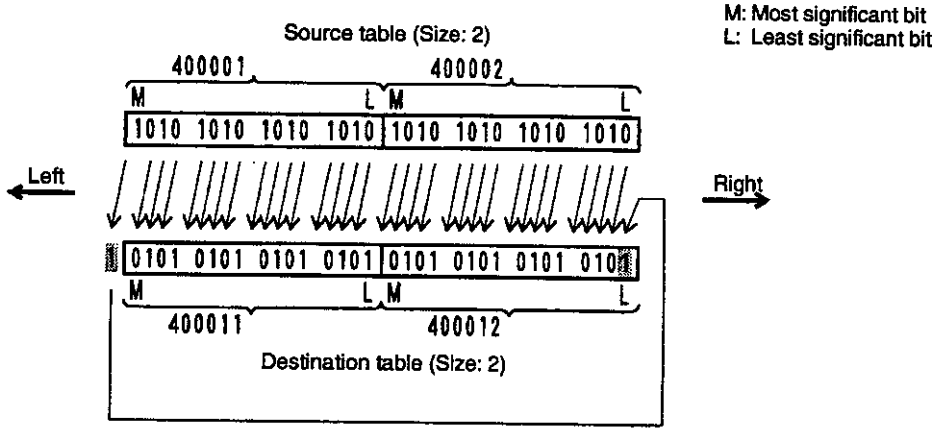
f) If input relay 100003 turns ON, the pointer value (n) will be set to 1 regardless of the status of input relay 100001.

5.3.8 LOGICAL BIT ROTATE (BROT)

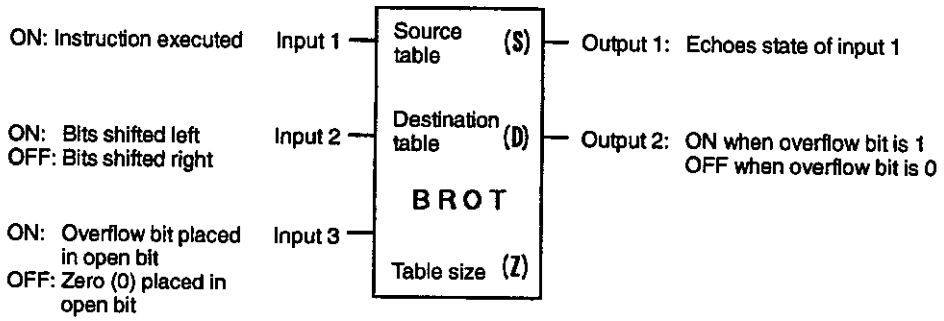
1. Function

The bit pattern in the source table is shifted one bit to the left or to the right and stored in a destination table of the same size. The operation is completed in one scan.

Example
If inputs 1, 2, and 3 are all ON, LOGICAL BIT ROTATE will output the bit pattern of the source table shifted one bit to the left and store the shifted bit pattern in the destination table. The bit that was shifted out of the left end will be input to the LSB, which became empty when the shift was executed.



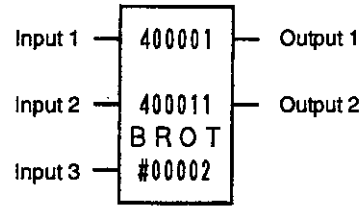
2. Structure



1) BROT is the symbol of LOGICAL BIT ROTATE.

2) BROT requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 5.13* lists the register reference numbers and constants that can be specified.

Example



400001: Leading reference number of the source table

400011: Leading reference number of the destination table

#00002: Size of source and destination tables (2)

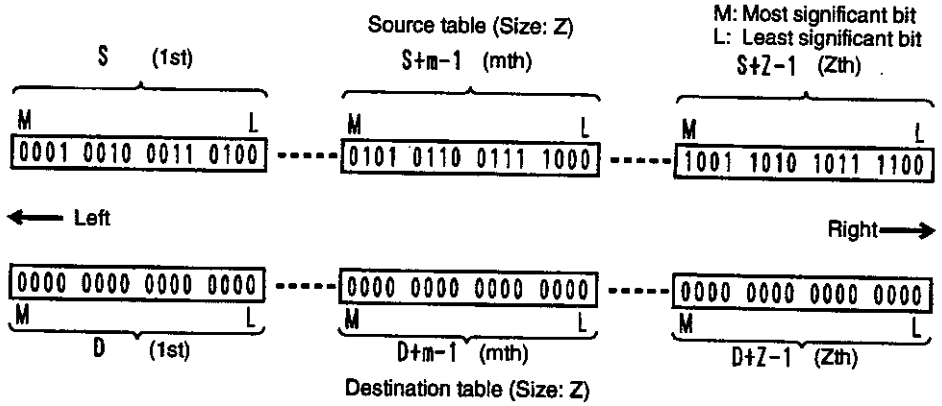
Table 5.13 Structural Elements of BROT

Element	Meaning	Possible settings	
Top (S)	Leading reference number of the source table	Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (D)	Leading reference number of the destination table	Coil:	000001 to 008161 (O00001 to O08161)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
Bottom (Z)	Size of source and destination tables	Specify the constant. The maximum value of the constant differs with the reference type.	
		Coil:	#00001 to #00100
		Input relay:	#00001 to #00064
		Input register, holding register or constant register:	#00001 to #00100
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00100
		MC coil, MC relay or MC control relay:	#00001 to #00016
		MC control coil:	#00001 to #00010
		M code relay:	#00001 to #00006

Note When specifying reference numbers for coils or relays, the following equation must be satisfied ("m" represents the lower 5 digits of the reference number):
m = 16n + 1, where n = 0, 1, 2, etc.

3. Operation

1) Before Execution



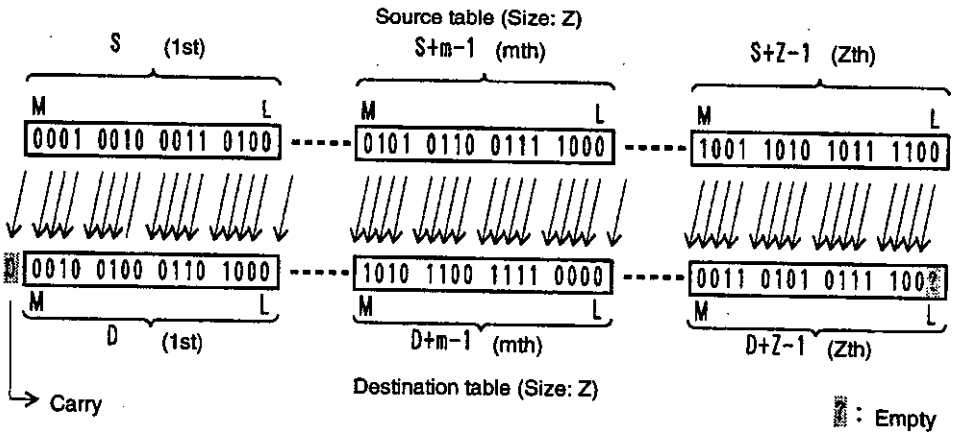
2) If input 2 is ON, the following processing will take place.

a) When input 1 turns ON, the bit pattern in the source table is shifted one bit to the left and stored in the destination table.

b) The bit shifted out of the destination table (called the carry) and the bit that became empty when data was shifted out are treated as follows:

(1) If input 3 is ON, the carry is placed in the empty bit.

(2) If input 3 is OFF, the carry is thrown away and 0 is placed in the empty bit.



c) The source data does not change.

d) The status of the outputs is treated as follows:

(1) The status of output 1 is the same as that of input 1.

(2) If the carry is 1, output 2 turns ON; if the carry is 0, output 2 turns OFF.

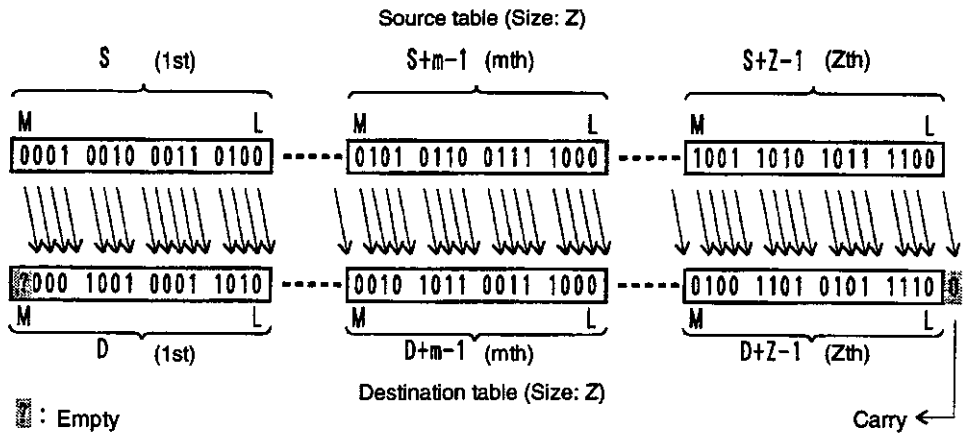
3) If input 2 is OFF, the following processing will take place.

a) When input 1 turns ON, the bit pattern in the source table is shifted one bit to the right and stored in the destination table.

b) The bit shifted out of the destination table (called the carry) and the bit that became empty when data was shifted out are treated as follows:

(1) If input 3 is ON, the carry is placed in the empty bit.

(2) If input 3 is OFF, the carry is thrown away and 0 is placed in the empty bit.



c) The source data does not change.

d) The status of the outputs are as follows:

(1) The status of output 1 is the same as that of input 1.

(2) If the carry is 1, output 2 turns ON; if the carry is 0, output 2 turns OFF.

4) An overview of the operation of BROT is shown in the following table. "Z" is the size of the source and destination tables.

Table 5.14 Operation of BROT

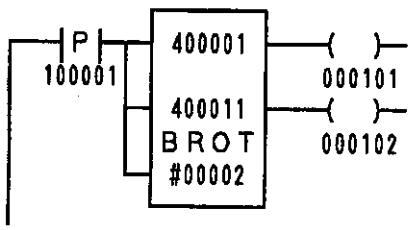
Inputs			Operation	Outputs	
1	2	3		1	2
ON	ON	OFF	1) Bit pattern in source table is shifted one bit to left and stored in destination table. 2) Carry is discarded and 0 placed in empty bit. 3) Data in source table does not change.	ON	ON when carry=1 OFF when carry=0
		ON	1) Bit pattern in source table is shifted one bit to left and stored in destination table. 2) Carry is placed in empty bit. 3) Data in source table does not change.		
	OFF	OFF	1) Bit pattern in source table is shifted one bit to right and stored in destination table. 2) Carry is discarded and 0 placed in empty bit. 3) Data in source table does not change.		
		ON	1) Bit pattern in source table is shifted one bit to right and stored in destination table. 2) Carry is placed in empty bit. 3) Data in source table does not change.		
OFF	Any	Any	1) Nothing is done. 2) Data in source and destination tables does not change.	OFF	OFF

5

4. Application Examples

◀EXAMPLE▶ **Example 1**

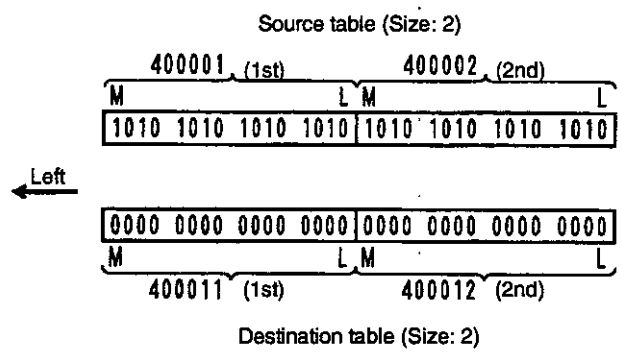
1) Ladder Programming



2) Operation

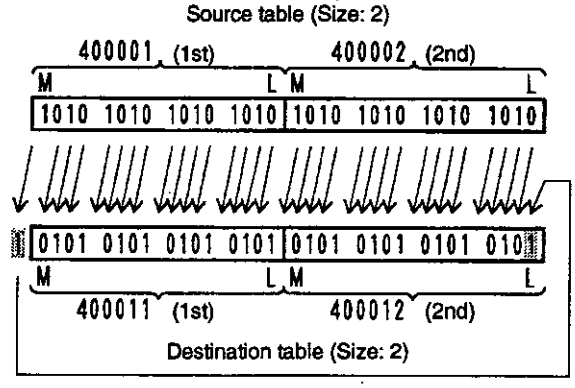
a) Before Execution

M: Most significant bit
L: Least significant bit



b) When input relay 100001 changes from OFF to ON, the bit pattern in the source table is shifted one bit to the left and stored in the destination table.

c) The bit shifted out of the destination table (called the carry) is placed in the bit that became empty when data was shifted out.



d) The data in the source table does not change.

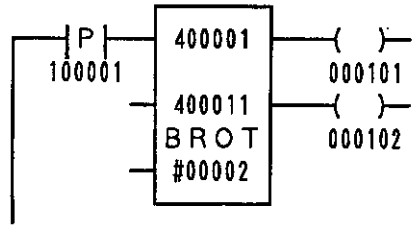
e) The status of the output coils are as follows:

(1) 000101: ON

(2) 000102: ON (because carry is 1)

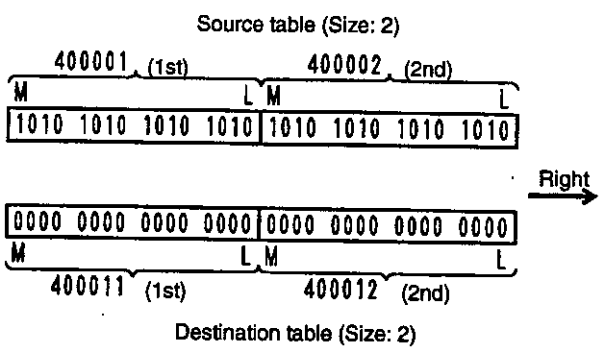
◀EXAMPLE▶ **Example 2**

1) Ladder Programming



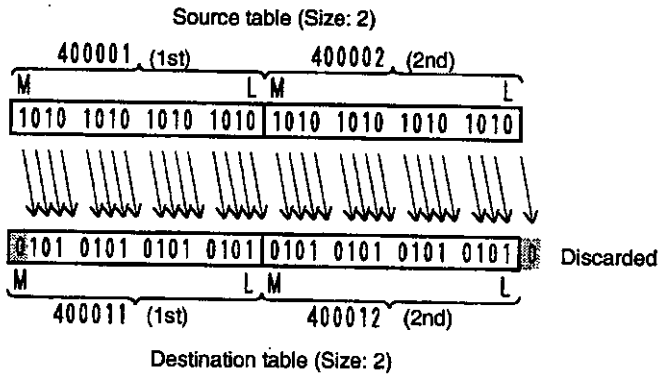
2) Operation

a) Before Execution



b) When input relay 100001 changes from OFF to ON, the bit pattern in the source table is shifted one bit to the right and stored in the destination table.

c) The bit shifted out of the destination table (called the carry) is discarded and 0 is placed in the bit that became empty when data was shifted out.



d) The data in the source table does not change.

e) The status of the output coils are as follows:

(1) 000101: ON

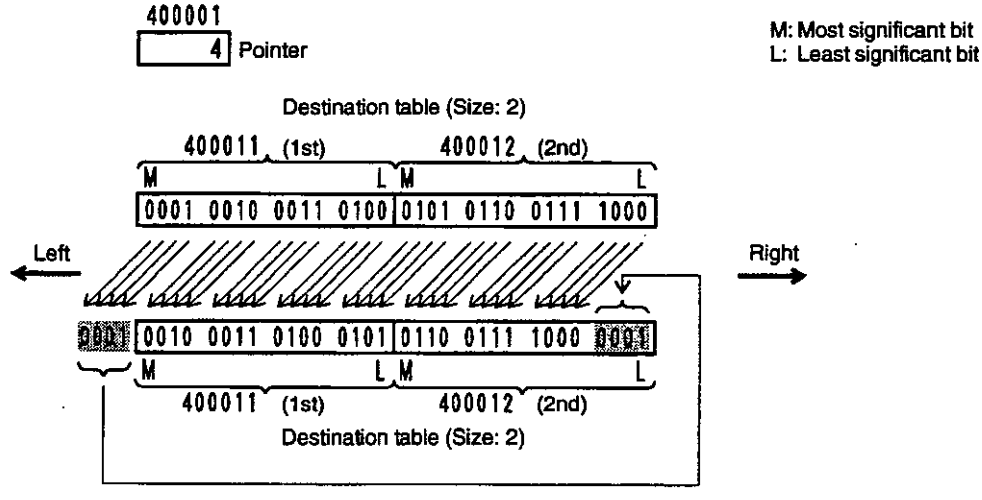
(2) 000102: OFF (because carry is 0)

5.3.9 LOGICAL MULTI-BIT ROTATE (MROT)

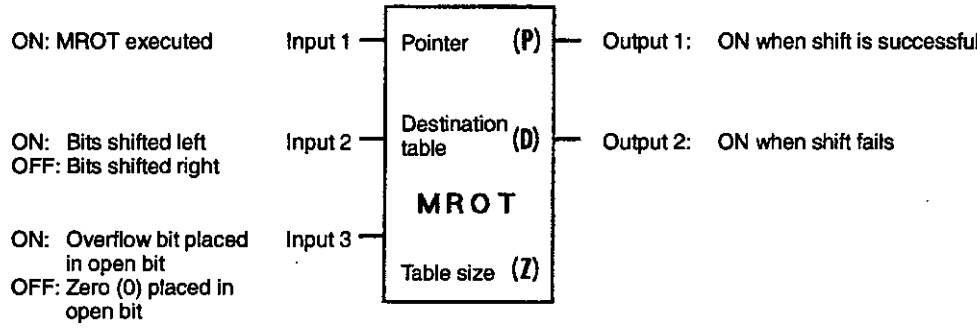
1. Function

The bits in the destination table are shifted to the left or to the right by the number of bits specified in the pointer (1 to 15). The operation is completed in one scan.

Example
If the pointer value is 4 and inputs 1, 2, and 3 are all ON, LOGICAL MULTI-BIT ROTATE will shift the bit pattern in the destination table four bits to the left, as shown in the following diagram. The bits that were shifted out of the left end will be input to the bits which became empty when the shift was executed.



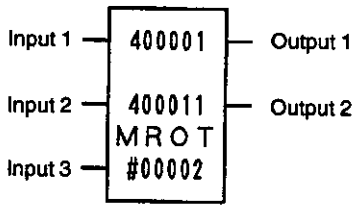
2. Structure



1) MROT is the symbol of LOGICAL MULTI-BIT ROTATE.

2) MROT requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 5.15* lists the register reference numbers and constants that can be specified.

Example



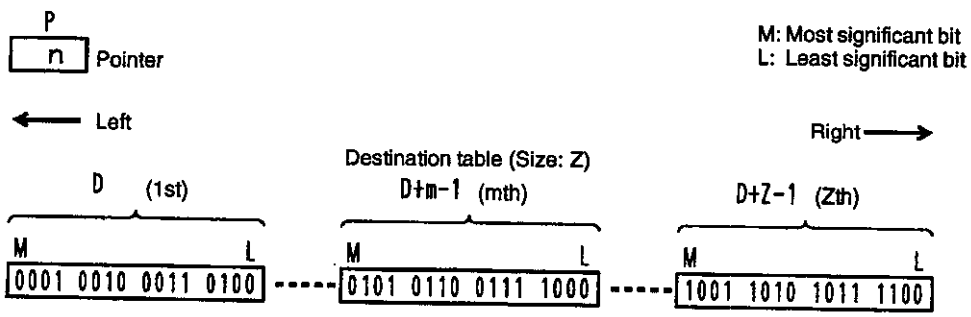
400001: Reference number of the pointer
400011: Leading reference number of the destination table
#00002: Size of the destination table (2)

Table 5.15 Structural Elements of MROT

Element	Meaning	Possible settings	
Top (P)	Reference number of the pointer	Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 or R20001 to R21024
Middle (D)	Leading reference number of the destination table	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 or R20001 to R21024
Bottom (Z)	Size of destination table	Constant:	#00001 to #00100

3. Operation

1) Before Execution



2) If input 2 is ON, the following processing will take place.

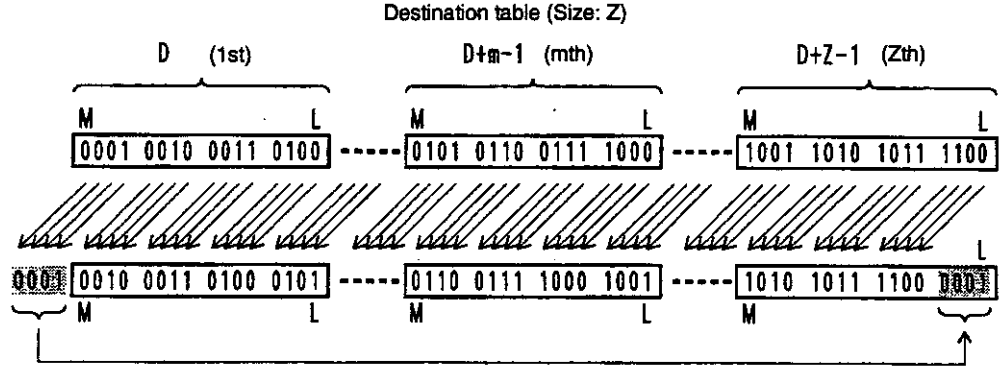
a) When input 1 turns ON, the bit pattern in the destination table is shifted n bits to the left. "n" is the pointer value and must be between 1 and 15.

b) The bits shifted out of the destination table (called the carries) and the bits which became empty when data was shifted out are treated as follows:

(1) If input 3 is ON, the carries are placed in the empty bits.

(2) If input 3 is OFF, the carries are thrown away and 0s are placed in the empty bits.

c) If n = 4 and inputs 1, 2, and 3 are ON, the operation shown below is performed, with the carries being placed in the empty bits.



d) The pointer value does not change.

e) The status of the outputs is treated as follows:

(1) Output 1: ON if shift is successful.

(2) Output 2: ON if the shift fails for any of the following reasons.
The pointer is part of the destination table.
The pointer value is not between 1 and 15, inclusive.

3) If input 2 is OFF, the following processing will take place.

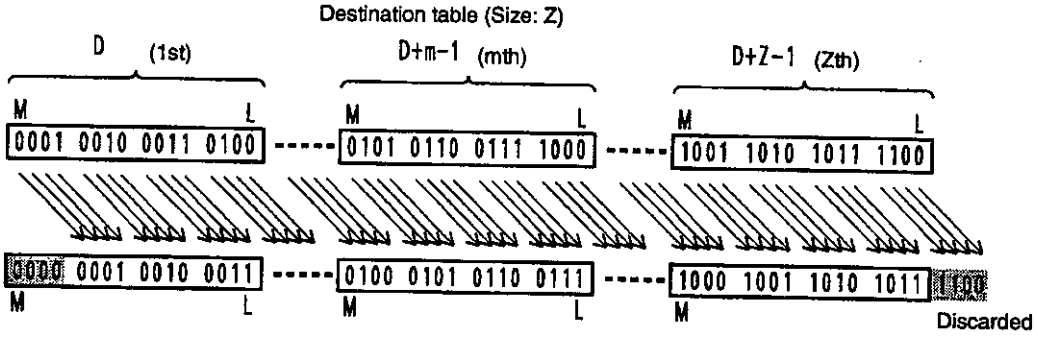
a) When input 1 turns ON, the bit pattern in the destination table is shifted n bits to the right. "n" is the pointer value and must be between 1 and 15.

b) The bits shifted out of the destination table (called the carries) and the bits which became empty when data was shifted out are treated as follows:

(1) If input 3 is ON, the carries are placed in the empty bits.

(2) If input 3 is OFF, the carries are thrown away and 0s are placed in the empty bits.

c) If n = 4 and only input 1 is ON, the operation shown below is performed. The carries are discarded and 0s are placed in the empty bits.



d) The pointer value does not change.

e) The status of the outputs are as follows:

(1) Output 1: ON if shift is successful.

(2) Output 2: ON if the shift fails for any of the following reasons.
The pointer is part of the destination table.
The pointer value is not between 1 and 15, inclusive.

4) If the pointer value (n) is 0, no bits are shifted, output 1 will turn ON and output 2 will turn OFF.

5) An overview of the operation of MROT is shown in the following table where "n" is the pointer value.

Table 5.16 Operation of MROT

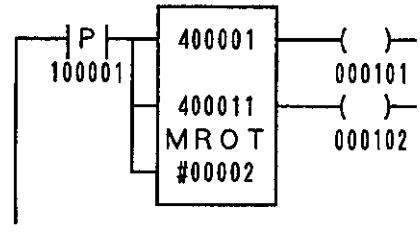
Inputs			Condition	Operation	Outputs	
1	2	3			1	2
ON	**ON**	OFF	$1 \le n \le 15$	1) Bits in destination table are shifted n bits to left. 2) Carries are discarded and 0s placed in empty bits. 3) Value of n does not change.	**ON**	OFF
		ON		1) Bits in destination table are shifted n bits to left. 2) Carries are placed in empty bits. 3) Value of n does not change.		
	OFF	OFF		1) Bits in destination table are shifted n bits to right. 2) Carries are discarded and 0s placed in empty bits. 3) Value of n does not change.		
		ON		1) Bits in destination table are shifted n bits to right. 2) Carries are placed in empty bits. 3) Value of n does not change.		
	Any	Any	n = 0	1) Nothing is done. 2) Pointer value and data in destination table do not change.	**ON**	OFF
			n < 0, n ≥ 16		OFF	**ON**
OFF	Any	Any	None			OFF

Note If the pointer is part of the destination table, nothing will be shifted even if input 1 is ON. Output 1 will turn OFF and output 2 will turn ON.

4. Application Examples

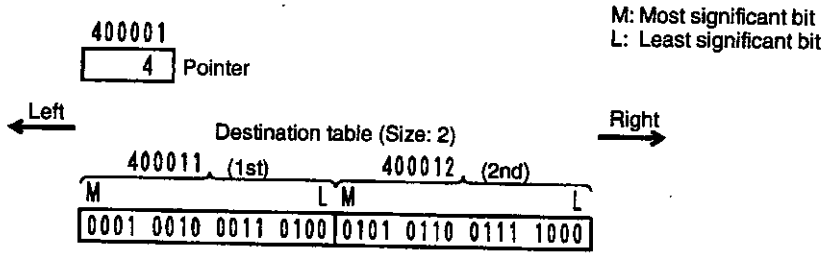
◀EXAMPLE▶ **Example 1**

1) Ladder Programming



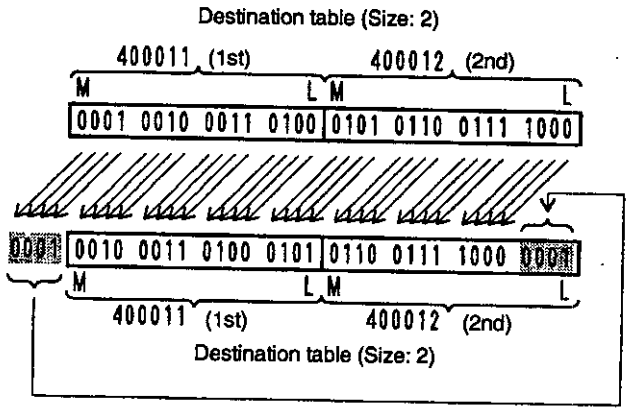
2) Operation

a) Before Execution



b) If the pointer value (n) is 4 when input relay 100001 changes from OFF to ON, the bits in the destination table are shifted 4 bits to the left.

c) The bits shifted out of the destination table (called the carries) are placed in the bits that became empty when data was shifted out.



d) The pointer value does not change.

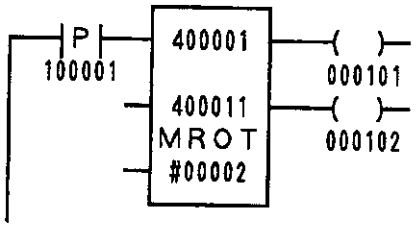
e) The status of the output coils are as follows:

(1) 000101: ON

(2) 000102: OFF

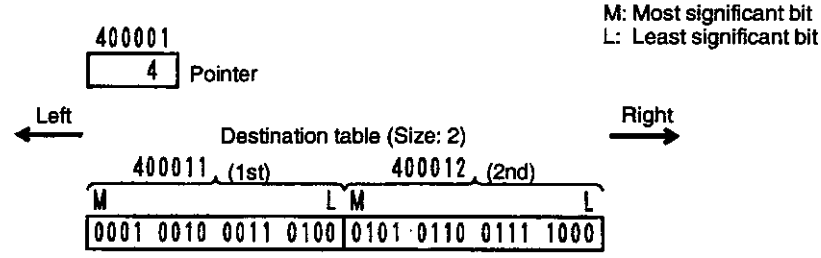
◀EXAMPLE▶ **Example 2**

1) Ladder Programming



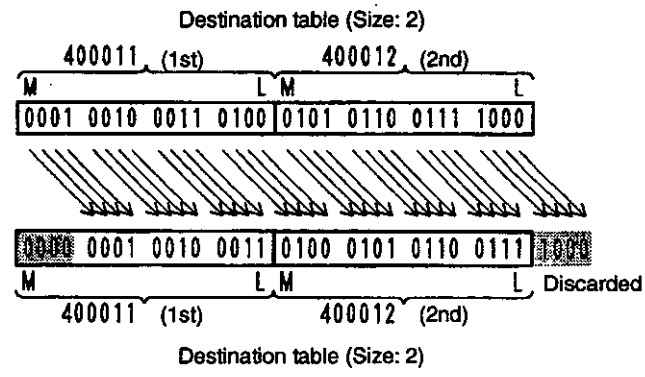
2) Operation

a) Before Execution



b) If the pointer value (n) is 4 when input relay 100001 changes from OFF to ON, the bits in the destination table are shifted 4 bits to the right.

c) The bits shifted out of the destination table (called the carries) are discarded and 0s are placed in the bits that became empty when data was shifted out.



d) The pointer value does not change.

e) The status of output coils are as follows:

(1) 000101: ON

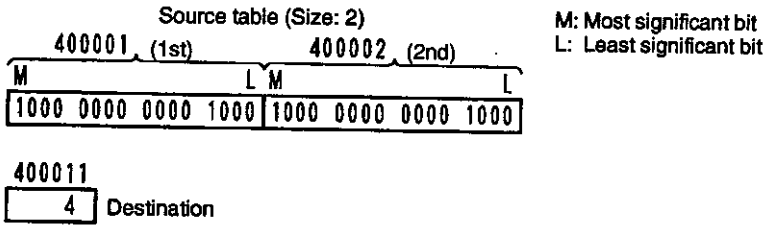
(2) 000102: OFF

5.3.10 LOGICAL BIT COUNT (BCNT)

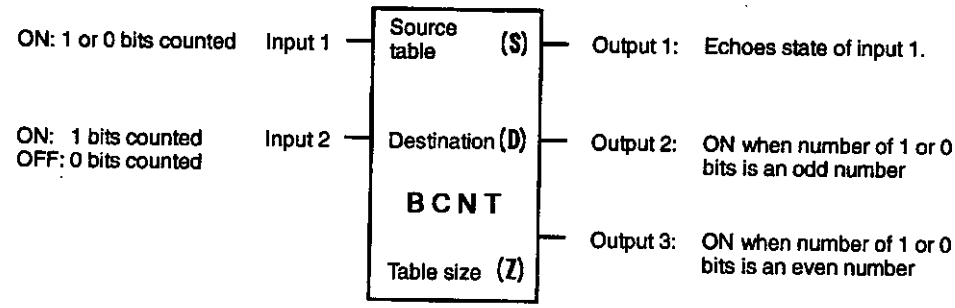
1. Function

The number of 1 or 0 bits in the source table is counted. The operation is completed in one scan.

Example
If inputs 1 and 2 are all ON, LOGICAL BIT COUNT will count the number of 1 bits in the source table, as shown in the following diagram. The number (4) will be stored in the destination.



2. Structure



1) BCNT is the symbol of LOGICAL BIT COUNT.

2) BCNT requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. *Table 5.17* lists the register reference numbers and constants that can be specified.

Example

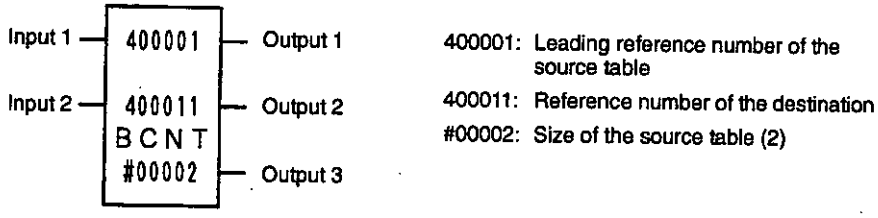
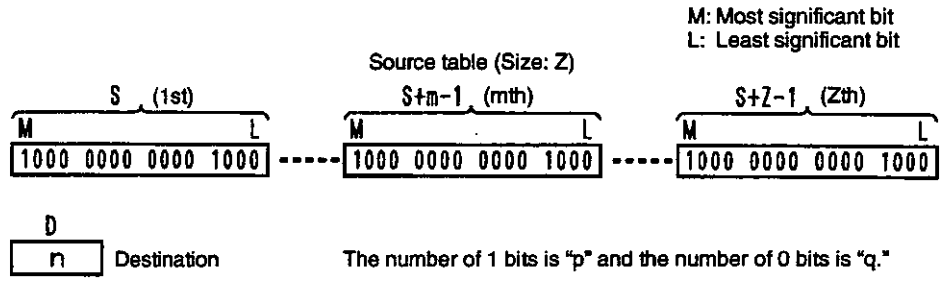


Table 5.17 Structural Elements of BCNT

Element	Meaning	Possible settings	
Top (S)	Leading reference number of the source table	Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 or R20001 to R21024
Middle (D)	Reference number of the destination	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 or R20001 to R21024
Bottom (Z)	Size of the source table	Constant:	#00001 to #00100

3. Operation

1) Before Execution



2) If inputs 1 and 2 are ON, the following processing will take place. This processing will be completed in one scan.

   a) The number of 1 bits in the source table is counted and the number (p) is stored in the destination:

   b) The data in the source table does not change.

   c) The status of the outputs are as follows:

      (1) Output 1: ON

      (2) Output 2: ON if number of 1 bits (p) is an odd number

      (3) Output 3: ON if number of 1 bits (p) is an even number

3) If input 1 is ON and input 2 is OFF, the following processing will take place. This processing will be completed in one scan.

a) The number of 0 bits in the source table is counted and the number (q) is stored in the destination:

b) The data in the source table does not change.

c) The status of the outputs are as follows:

(1) Output 1: ON

(2) Output 2: ON if number of 0 bits (q) is an odd number

(3) Output 3: ON if number of 0 bits (q) is an even number

4) An overview of the operation of BCNT is shown in the following table.

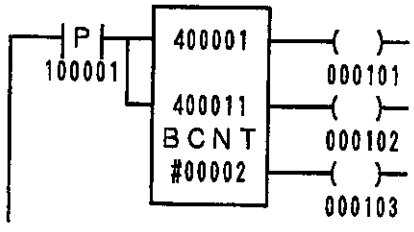
Table 5.18 Operation of BCNT

Inputs		Operation	Outputs		
1	2		1	2	3
ON	**ON**	1) Number of 1s in source table is counted and number is stored in destination. 2) Data in source table does not change.	**ON**	**ON** if number of 1s is odd	**ON** if number of 1s is even
	OFF	1) Number of 0s in source table is counted and number is stored in destination. 2) Data in source table does not change.		**ON** if number of 0s is odd	**ON** if number of 0s is even
OFF	Any	1) Nothing is done. 2) Data in source table does not change.	OFF	OFF	OFF

◀EXAMPLE▶

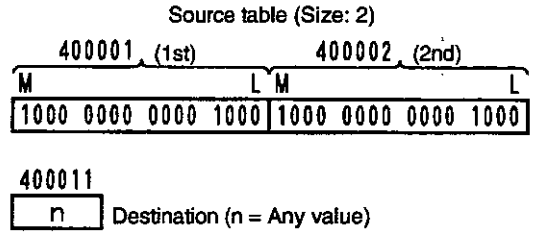
4. Application Example

1) Ladder Programming



2) Operation

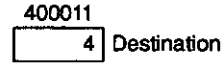
a) Before Execution



M: Most significant bit
L: Least significant bit

3) When input 1 changes from OFF to ON, the following processing will take place. This processing will be completed in one scan.

a) The number of 1 bits in the source table is counted and the number (4) is stored in the destination:



b) The data in the source table does not change.

c) The status of the output coils are as follows:

(1) 000101: ON

(2) 000102: OFF (because the number of 1 bits (4) is not odd)

(3) 000103: ON (because the number of 1 bits (4) is even)

5.4 Building Programs

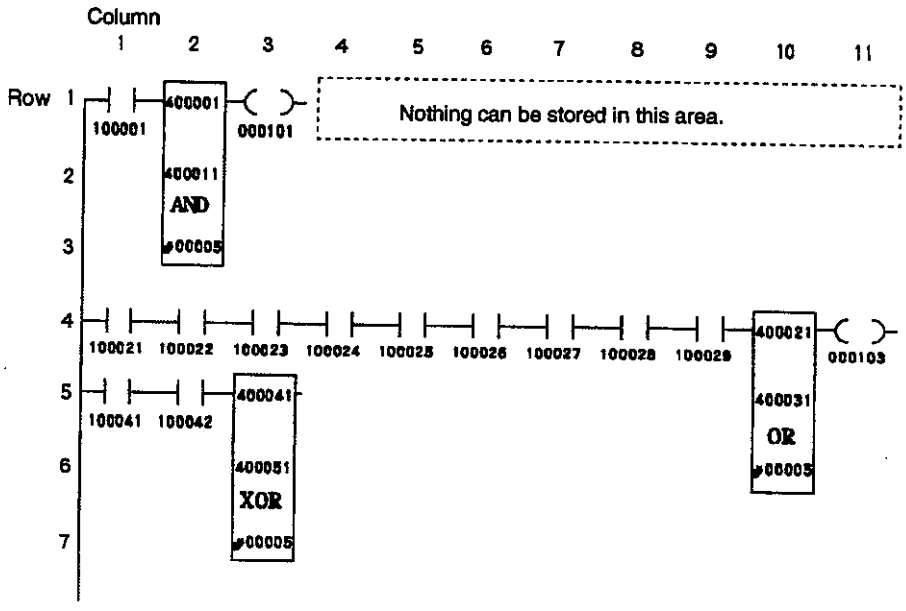
This section describes how to build a program using matrix instructions.

5.4.1 Storage Locations on Networks

All matrix instructions require three vertical elements on a network, one top element, one middle element, and one bottom element. They can thus be stored anywhere on a 5-row by 10-column matrix (rows 1 through 5 and columns 1 through 10) on the network.

Note Matrix instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

Example

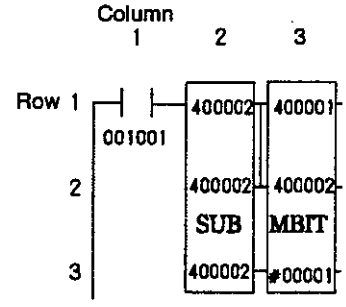


5

5.4.2 Inputs

Inputs to matrix instruction can be connected to relay elements (except coils) and outputs from timers, counters, math instructions, data transfer instructions, other instructions, etc.

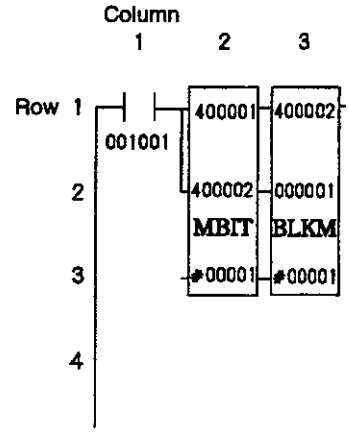
Example



5.4.3 Outputs

Outputs to matrix instruction can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data transfer instructions, etc.

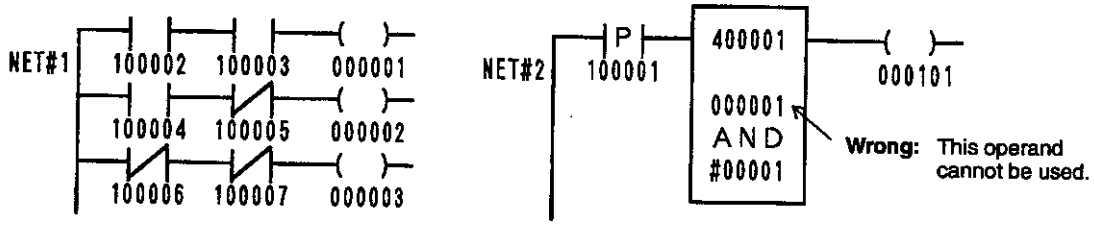
Example



5.4.4 Duplicate Coil Usage

1) A coil table that includes coils that have already been used cannot be used as a destination.

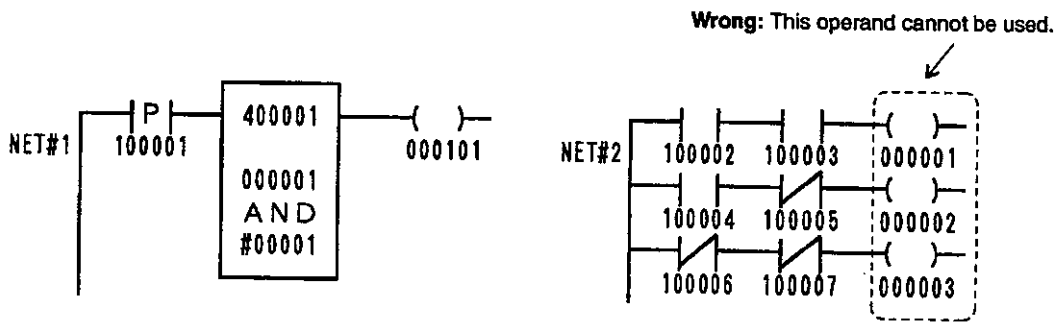
Example: Incorrect Application



Coils 000001 to 000003 are used in network #1, so coil 000001 cannot be used as the reference for a destination, such as the one shown above in network #2.

2) The coils in coil tables used as destinations cannot be used again as coils. The coils on the right in the following diagram cannot be used because they have already been used as a destination.

Example: Incorrect Application

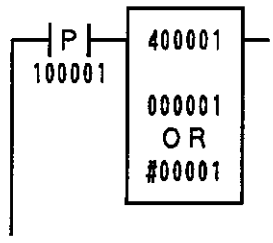


Coils 000001 to 000016 are used in network #1, so coils 000001 to 000003 cannot be used as the references in network #2.

5.4.5 Operation of Disabled Coils

Do not execute matrix instructions containing disabled coils (including coils, link coils, MC coils, and MC control coils) as destinations. If disabled coils are used as destinations, their status will be overwritten by the matrix instruction, as shown in the following example. The disabled coils, however, will not be enabled.

Example



1) Assume the following status for holding register 400001 and coils 000001 to 000016:

400001 1111 1111 1111 1111

Coils 000001 to 000016: All disabled OFF

2) When input relay 100001 changes from OFF to ON, all coils from 000001 to 000016 will be disabled ON.

Bit Manipulation Instructions

6

This chapter describes the instructions used to manipulate register bits like contacts and coils.

6.1 Bit Manipulation Instructions

This section introduces the bit manipulation instructions, which are used to apply contact and coil functions for relay elements to register bits.

- The five bit manipulation instructions are described in the following table.

Table 6.1 Bit Manipulation Instructions

Name	Symbol	Function	Page
NORMALLY OPEN BIT	NOBT	1) The logic state of a specified bit in a specified register is tested and output 1 is turned ON or OFF accordingly. 2) The same operation is performed as that for a normally open contact relay element. For a normally open contact, power flow is controlled based on the ON/OFF status of a coil. With NORMALLY OPEN BIT, power flow is controlled based on the logic state (1/0) of a specified bit.	6-3
NORMALLY CLOSED BIT	NCBT	1) The logic state of a specified bit in a specified register is tested and output 1 is turned ON or OFF accordingly. 2) The same operation is performed as that for a normally closed contact of a relay element. For a normally closed contact, power flow is controlled based on the ON/OFF status of a coil. With NORMALLY CLOSED BIT, power flow is controlled based on the logic state (1/0) of a specified bit.	6-5
NORMAL BIT	NBIT	1) The logic state of a specified bit in a specified register is set to 1 or reset to 0. 2) The same operation is performed as that for a coil of a relay element. For a coil, power flow controls the ON/OFF status of a coil. With NORMAL BIT, power flow controls the logic state (1/0) of a specified bit.	6-7
SET BIT	SBIT	1) The logic state of a specified bit in a specified register is set to 1. SET BIT can be combined with RESET BIT to perform the same operation as that for a latched coil of a relay element. 2) The logic state of a specified coil in 16 consecutive coils is set to ON. The function allows duplicate coil usage, i.e., a coil can be controlled from more than one place in the program. This function can also be used for link coils, MC coils and MC control coils.	6-9
RESET BIT	RBIT	1) The logic state of a specified bit in a specified register is reset to 0. RESET BIT can be combined with SET BIT to perform the same operation as that for a latched coil of a relay element. 2) The logic state of a specified coil in 16 consecutive coils is set to OFF. The function allows duplicate coil usage, i.e., a coil can be controlled from more than one place in the program. This function can also be used for link coils, MC coils and MC control coils.	6-11

6.2 Details of Bit Manipulation Instructions

This section describes the functions, structures, and operation of each bit manipulation instruction, and provides simple examples of their application.

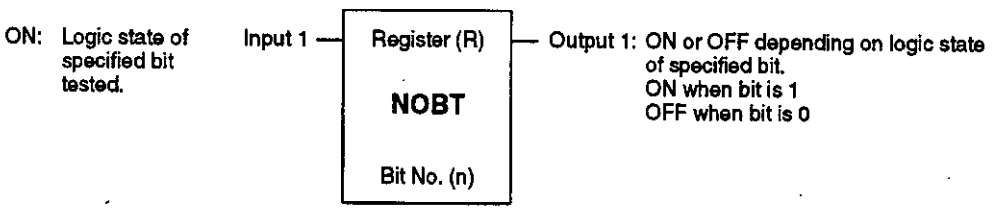
6.2.1 NORMALLY OPEN BIT (NOBT)

1. Function

1) The logic state of a specified bit in a specified register is tested and output 1 is turned ON or OFF accordingly.

2) The same operation is performed as that for a normally open contact relay element. For a normally open contact, power flow is controlled based on the ON/OFF status of a coil. With NORMALLY OPEN BIT, power flow is controlled based on the logic state (1/0) of a specified bit.

2. Structure



1) NOBT is the symbol for NORMALLY OPEN BIT.

2) NOBT requires two elements, one top element and one bottom element, located vertically on the network. Refer to *Table 6.2* for details on specifying constants or registers for these elements.

Example

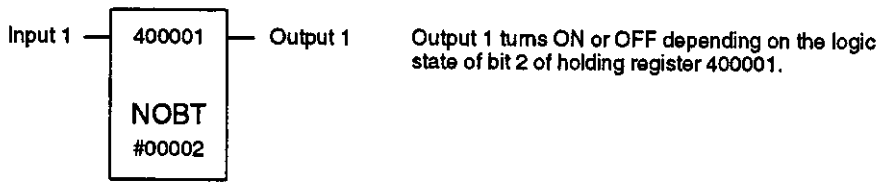
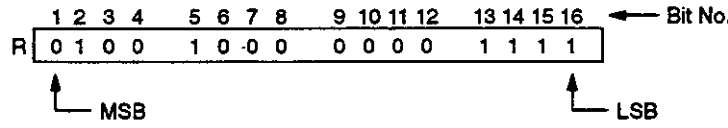


Table 6.2 Structural Elements of NOBT

Element	Meaning	Possible Settings	
Top (R)	The logic state of bit n in R, the register specified for the top element, is tested and output 1 is turned ON or OFF accordingly.	Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 R20001 to R21024
Bottom (n)		#00001 to #00016	

3. Operation

1) Status Before Execution



2) The following processing will be performed when input 1 turns ON.

a) The value of n specified for the bottom element ($1 \leq n \leq 16$) is taken as a bit number, the logic state of bit n in the register specified for the top element is tested, and output 1 is turned ON or OFF accordingly.

(1) If the bit is 1, output 1 will be turned ON.

(2) If the bit is 0, output 1 will be turned OFF.

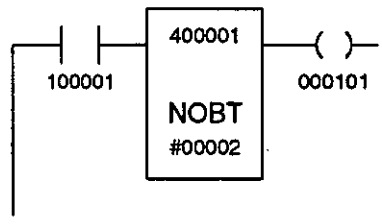
b) The data in the register does not change.

3) Output 1 will be turned OFF when input 1 turns OFF.

◀EXAMPLE▶

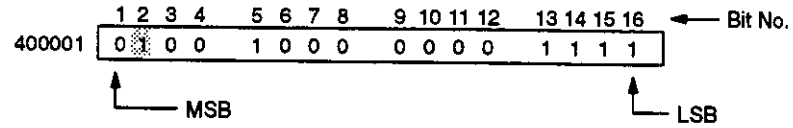
4. Application Example

1) Ladder Programming



2) Processing

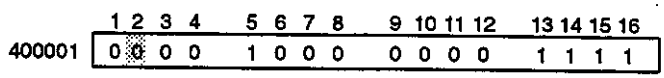
a) Before Execution



b) If input relay 100001 turns ON with the status shown above, coil 000101 will be turned ON because bit 2 of holding register 400001 is 1. The contents of holding register 400001 does not change.

c) Coil 000101 will turn OFF when input relay 100001 turns OFF.

d) If input relay 100001 turns ON with the status shown below, coil 000101 will be turned OFF because bit 2 of holding register 400001 is 0.



e) When input 1 turns ON, NOBT will thus turn output 1 ON or OFF according to the logic state of the specified bit, as shown above.

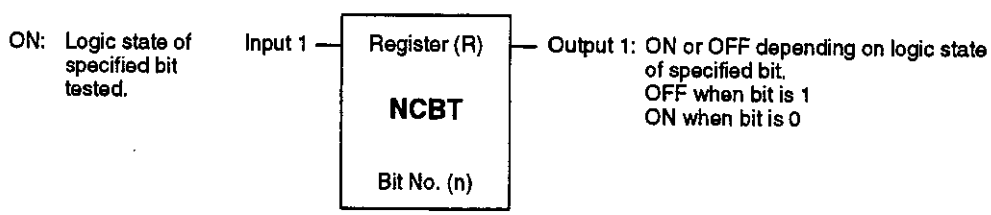
6.2.2 NORMALLY CLOSED BIT (NCBT)

1. Function

1) The logic state of a specified bit in a specified register is tested and output 1 is turned ON or OFF accordingly.

2) The same operation is performed as that for a normally closed contact relay element. For a normally closed contact, power flow is controlled based on the ON/OFF status of a coil. With NORMALLY CLOSED BIT, power flow is controlled based on the logic state (1/0) of a specified bit.

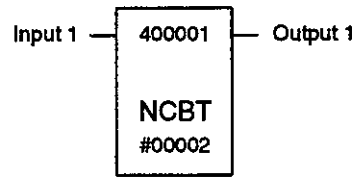
2. Structure



1) NCBT is the symbol for NORMALLY CLOSED BIT.

2) NCBT requires two elements, one top element and one bottom element, located vertically on the network. Refer to *Table 6.3* for details on specifying constants or registers for these elements.

Example



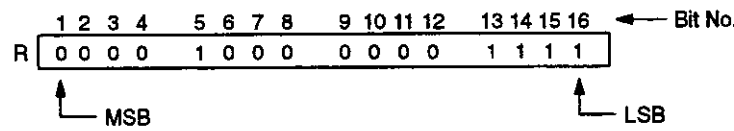
Output 1 turns ON or OFF depending on the logic state of bit 2 of holding register 400001.

Table 6.3 Structural Elements of NCBT

Element	Meaning	Possible Settings	
Top (R)	The logic state of bit n in R, the register specified for the top element, is tested and output 1 is turned ON or OFF accordingly.	Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 R20001 to R21024
Bottom (n)		#00001 to #00016	

3. Operation

1) Status Before Execution



2) The following processing will be performed when input 1 turns ON.

a) The value of n specified for the bottom element ($1 \leq n \leq 16$) is taken as a bit number, the logic state of bit n in the register specified for the top element is tested, and output 1 is turned ON or OFF accordingly.

(1) If the bit is 1, output 1 will be turned OFF.

(2) If the bit is 0, output 1 will be turned ON.

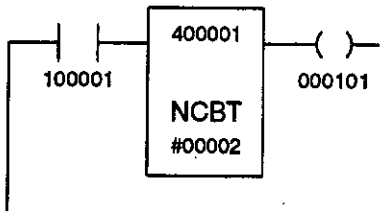
b) The data in the register does not change.

3) Output 1 will be turned OFF when input 1 turns OFF.

◀EXAMPLE▶

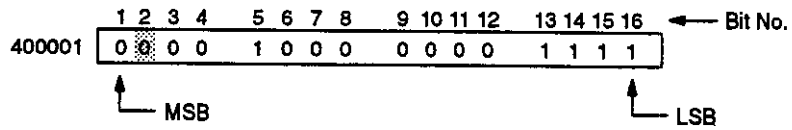
4. Application Example

1) Ladder Programming



2) Processing

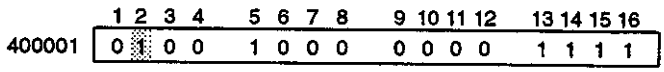
a) Before Execution



b) If input relay 100001 turns ON with the status shown above, coil 000101 will be turned ON because bit 2 of holding register 400001 is 0. The contents of holding register 400001 will not change.

c) Coil 000101 will be turned OFF when input relay 100001 turns OFF.

d) If input relay 100001 turns ON with the status shown below, coil 000101 will remain OFF because bit 2 of holding register 400001 is 1.



e) When input 1 turns ON, NCBT will thus turn output 1 ON or OFF according to the logic state of the specified bit, as shown above.

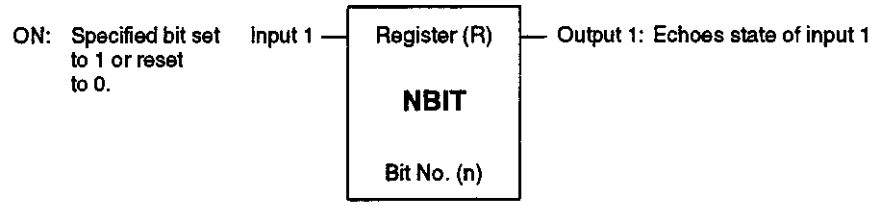
6.2.3 NORMAL BIT (NBIT)

1. Function

1) The logic state of a specified bit in a specified register is set to 1 or reset to 0.

2) The same operation is performed as that for a coil relay element. For a coil, power flow controls the ON/OFF status of a coil. With NORMAL BIT, power flow controls the logic state (1/0) of a specified bit.

2. Structure



1) NBIT is the symbol for NORMAL BIT.

2) NBIT requires two elements, one top element and one bottom element, located vertically on the network. Refer to *Table 6.4* for details on specifying constants or registers for these elements.

Example

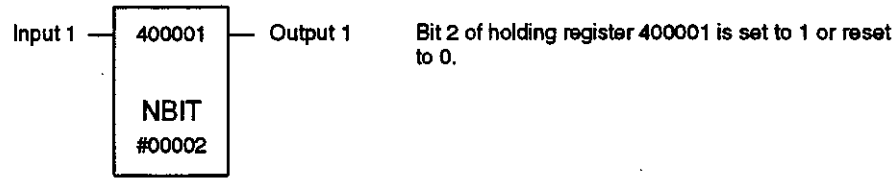
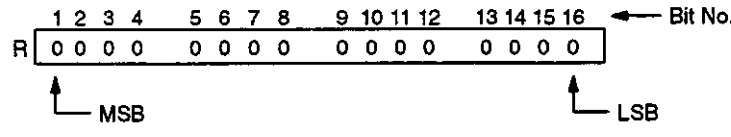


Table 6.4 Structural Elements of NBIT

Element	Meaning	Possible Settings
Top (R)	The logic state of bit n in R, the register specified for the top element, is set to 1 or reset to 0.	Holding register: 400001 to 409999 (W00001 to W09999) Link register: R10001 to R11024 R20001 to R21024
Bottom (n)		#00001 to #00016

3. Operation

1) Status Before Execution



2) The following processing will be performed when input 1 turns ON.

a) The value of n specified for the bottom element ($1 \leq n \leq 16$) is taken as a bit number and the logic state of bit n in the register specified for the top element is set to 1.

b) Output 1 is turned ON.

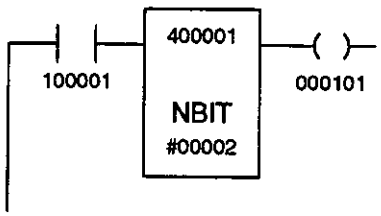
3) The following processing will be performed when input 1 turns OFF.

a) The logic state of bit n in the register specified for the top element is reset to 0.

b) Output 1 is turned OFF.

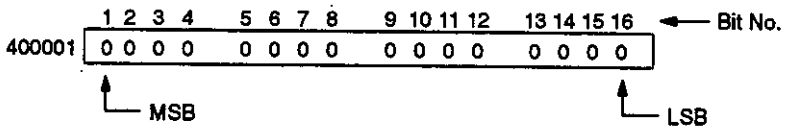
◀EXAMPLE▶ **4. Application Example**

1) Ladder Programming



2) Processing

a) Before Execution



b) When input relay 100001 turns ON, bit 2 of holding register 400001 will be set to 1 and coil 000101 will be turned ON.

c) When input relay 100001 turns OFF, bit 2 of holding register 400001 will be reset to 0 and coil 000101 will be turned OFF.

d) NBIT thus sets or resets the specified bit according to the ON/OFF status of input 1, as shown above.

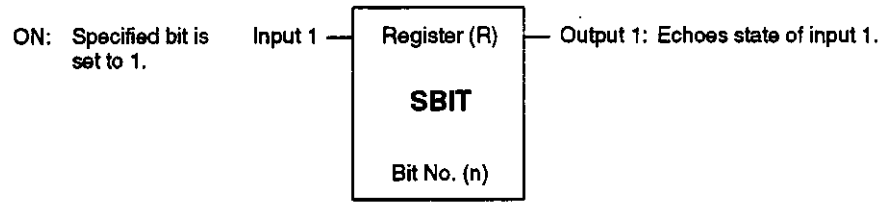
6.2.4 SET BIT (SBIT)

1. Function

1) The logic state of a specified bit in a specified register is set to 1.

2) SET BIT can be combined with RESET BIT (see *6.2.5 RESET BIT (RBIT)*) to perform the same operation as that for a latched coil relay element.

2. Structure



1) SBIT is the symbol for SET BIT.

2) SBIT requires two elements, one top element and one bottom element, located vertically on the network. Refer to *Table 6.5* for details on specifying constants or registers for these elements.

Example

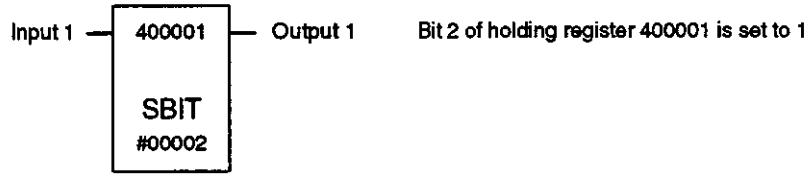
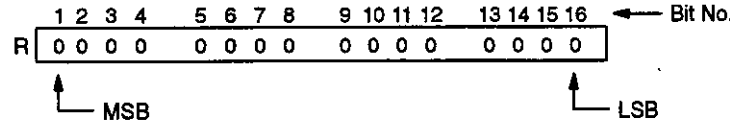


Table 6.5 Structural Elements of SBIT

Element	Meaning	Possible Settings	
Top (R)	The logic state of bit n in R, the register specified for the top element, is set to 1.	Holding register:	400001 to 409999 (W00001 to W09999)
		Coil:	000001 to 008177
		Link coil:	D10001 to D11009
		MC coil:	Y10001 to Y10241
		MC control coil:	Q10001 to Q10145
		Link register:	R10001 to R11024 R20001 to R21024
Bottom (n)		#00001 to #00016	

3. Operation

1) Status Before Execution



2) The following processing will be performed when input 1 turns ON.

a) The value of n specified for the bottom element ($1 \leq n \leq 16$) is taken as a bit number and the logic state of bit n in the register specified for the top element is set to 1.

b) Output 1 is turned ON.

3) The following processing will be performed when input 1 turns OFF.

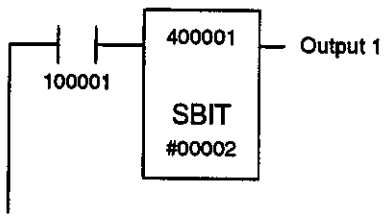
a) The logic state of bit n in the register specified for the top element remains set to 1.

b) Output 1 is turned OFF.

◀EXAMPLE▶

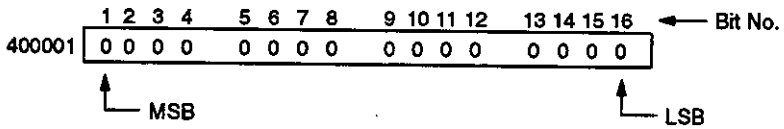
4. Application Example

1) Ladder Programming



2) Processing

a) Before Execution



b) When input relay 100001 turns ON, bit 2 of holding register 400001 will be set to 1 and output 1 will be turned ON.

c) When input relay 100001 turns OFF, bit 2 of holding register 400001 will remain at 1 and output 1 will be turned OFF.

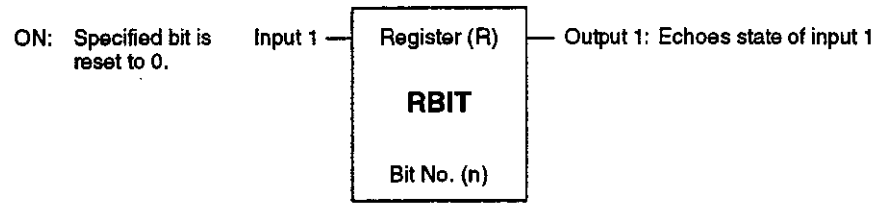
6.2.5 RESET BIT (RBIT)

1. Function

1) The logic state of a specified bit in a specified register is reset to 0.

2) RESET BIT can be combined with SET BIT (see *6.2.4 SET BIT (SBIT)*) to perform the same operation as that for a latched coil relay element.

2. Structure



1) RBIT is the symbol for RESET BIT.

2) RBIT requires two elements, one top element and one bottom element, located vertically on the network. Refer to *Table 6.6* for details on specifying constants or registers for these elements.

Example

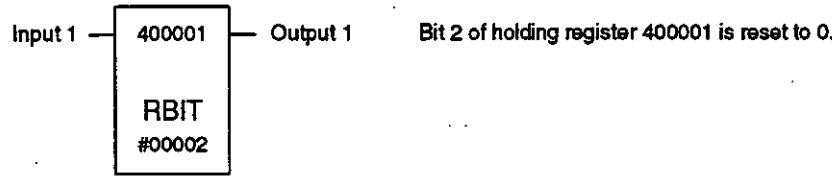
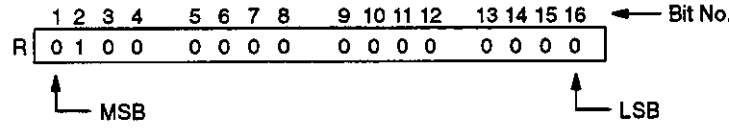


Table 6.6 Structural Elements of RBIT

Element	Meaning	Settings	
Top (R)	The logic state of bit n in R, the register specified for the top element, is reset to 0.	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 R20001 to R21024
Bottom (n)		#00001 to #00016	

3. Operation

1) Status Before Execution



2) The following processing will be performed when input 1 turns ON.

a) The value of n specified for the bottom element ($1 \leq n \leq 16$) is taken as a bit number and the logic state of bit n in the register specified for the top element is reset to 0.

b) Output 1 is turned ON.

3) The following processing will be performed when input 1 turns OFF.

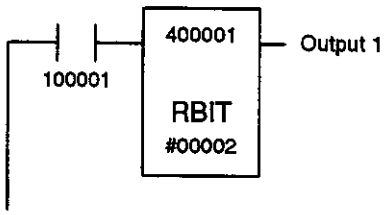
a) The logic state of bit n in the register specified for the top element remains reset to 0.

b) Output 1 is turned OFF.

◀EXAMPLE▶

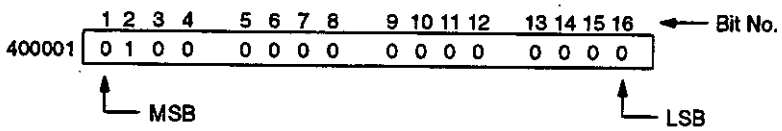
4. Application Example

1) Ladder Programming



2) Processing

a) Before Execution



b) When input relay 100001 turns ON, bit 2 of holding register 400001 will be reset to 0 and output 1 will be turned ON.

c) When input relay 100001 turns OFF, bit 2 of holding register 400001 will remain at 0 and output 1 will be turned OFF.

6

6.3 Building Programs

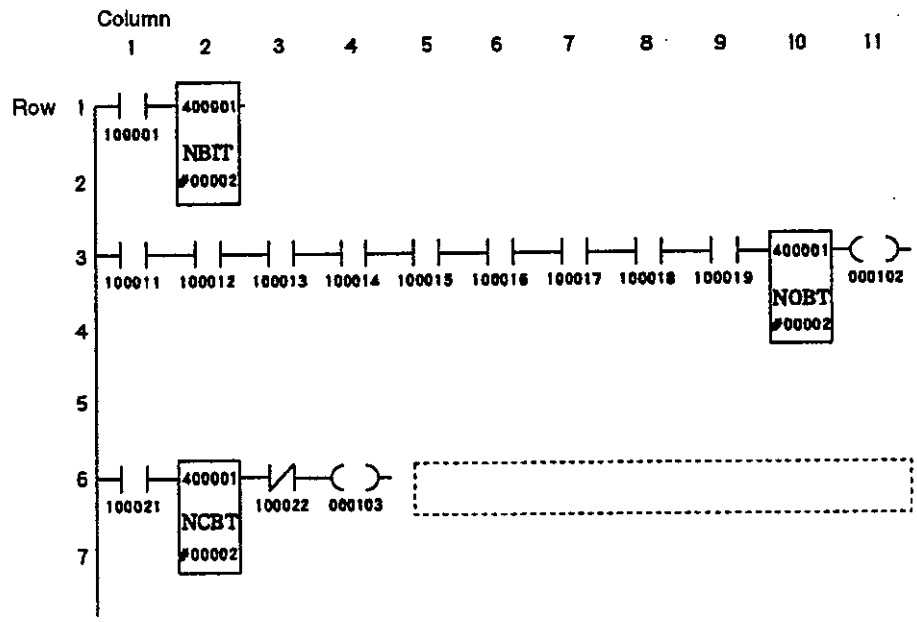
This section describes precautions that should be taken when designing programs that contain bit manipulation instructions.

6.3.1 Storage Locations on Networks

All bit manipulation instructions require two elements (top and bottom) located vertically on the network, so they can be stored anywhere on a 6-row by 10-column matrix (rows 1 through 6 and columns 1 through 10).

Note Bit manipulation instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

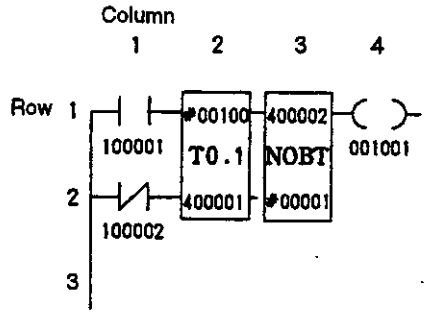
Example



6.3.2 Inputs

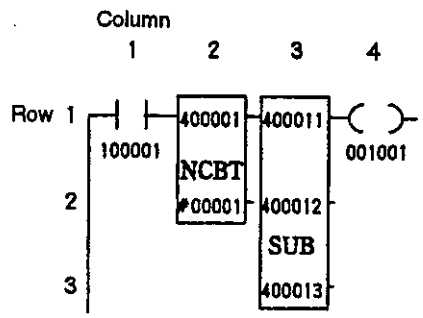
Inputs to bit manipulation instructions can be connected to relay elements (except coils) and/or outputs from timers, counters, math instructions, data transfer instructions, other instructions, etc.

Example



6.3.3 Outputs

Outputs from bit manipulation instructions can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data transfer instructions, etc.



Data Conversion Instructions

7

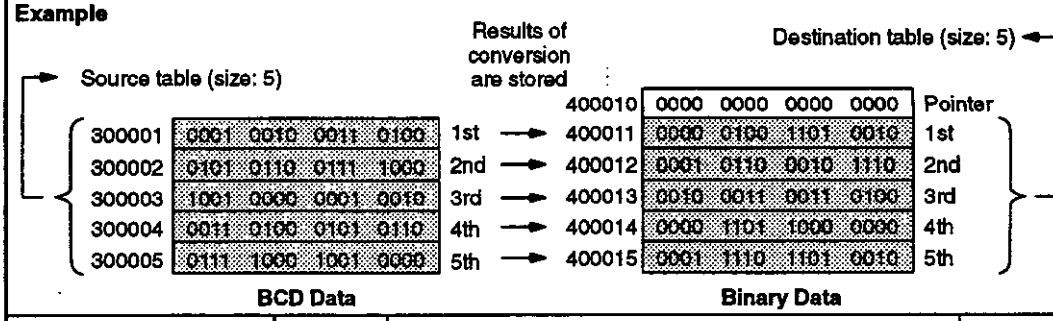
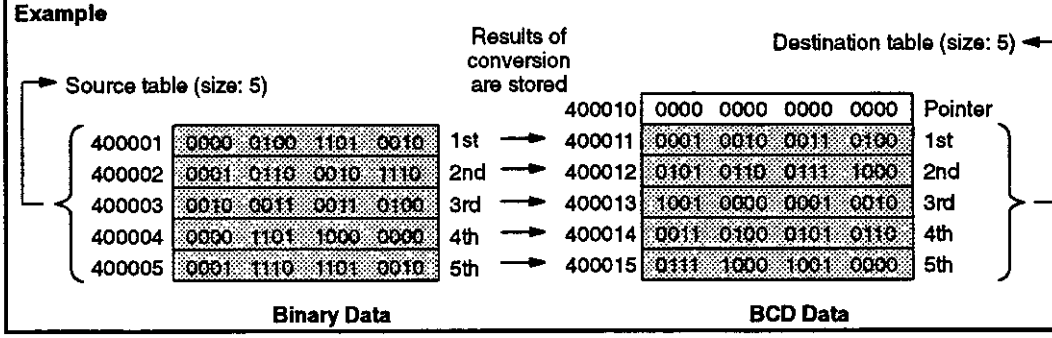
This chapter describes the instructions used to convert data between different types of expression.

7.1 Data Conversion Instructions

This section describes how to convert the expressions used to store data in holding registers and other registers.

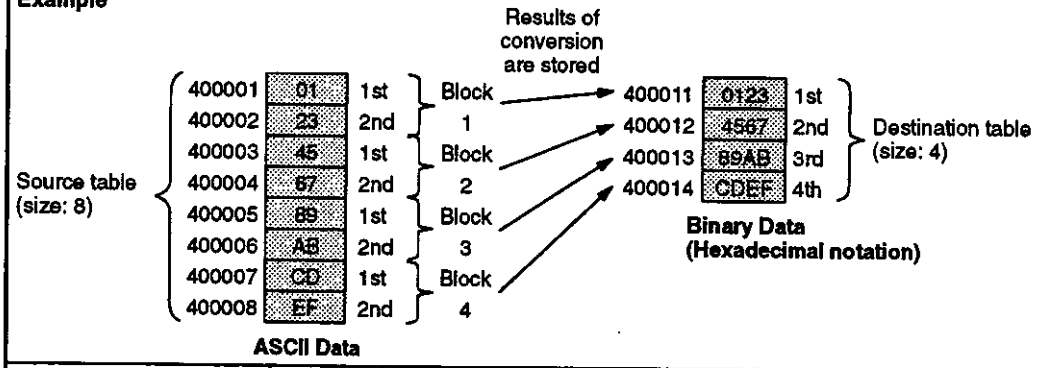
- The six data conversion instructions are described in the following table.

Table 7.1 Data Conversion Instructions

Name	Symbol	Function	Page
BCD-TO-BINARY CONVERSION	BIN	The data in the registers of the source table is treated as 4-digit BCD data, converted to binary data, and stored in corresponding registers of the destination table. The conversion is completed in one scan.	7-5
Example  			
BINARY-TO-BCD CONVERSION	BCD	The data in the registers of the source table is converted from binary data to 4-digit BCD data and stored in corresponding registers of the destination table. The conversion is completed in one scan.	7-11
Example  			

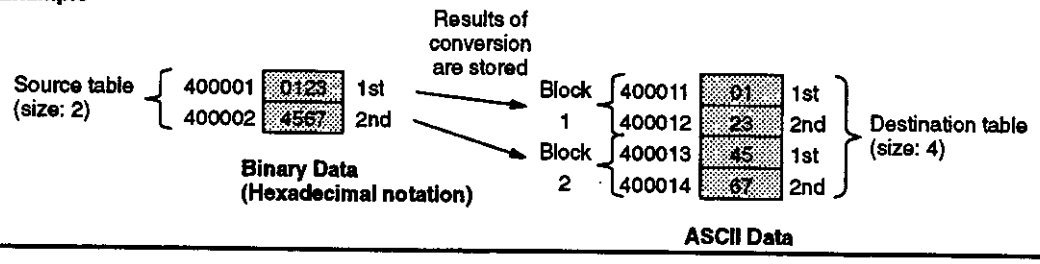
Name	Symbol	Function	Page
ASCII-TO-BINARY CONVERSION	ATOB	The data in each pair of registers forming a block in the source table is treated as four ASCII characters and converted to 4-bit binary data and stored in corresponding registers of the destination table. The conversion is completed in one scan. The only ASCII characters that can be converted are 0 to 9 and A to F.	7-17

Example



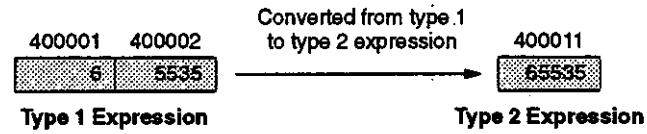
BINARY-TO-ASCII CONVERSION	BTOA	The data in each register of the source table is separated into 4 sets of 4-bit binary data, converted into four ASCII characters, and stored in corresponding blocks of the destination table. The conversion is completed in one scan.	7-23

Example



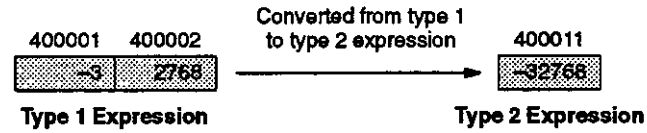
Name	Symbol	Function	Page
16-BIT DATA CONVERSION	CAST	The numeric expression of a 16-bit binary integer is changed from type 1 to type 2 or from type 2 to type 1. The conversion is completed in one scan.	7-27

Example 1



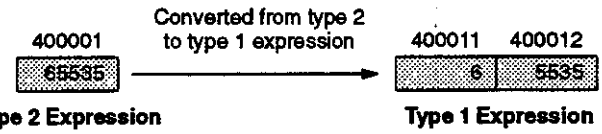
Type 1 Expression — Type 2 Expression

Example 2



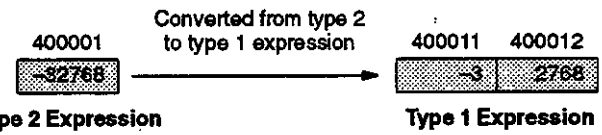
Type 1 Expression — Type 2 Expression

Example 3



Type 2 Expression — Type 1 Expression

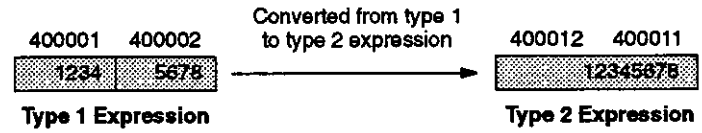
Example 4



Type 2 Expression — Type 1 Expression

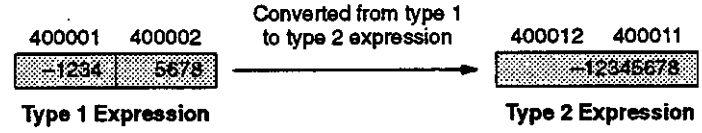
32-BIT DATA CONVERSION	DCST	The numeric expression of a signed or unsigned 8-digit decimal integer is changed from type 1 to type 2 or from type 2 to type 1. The conversion is completed in one scan.	7-36

Example 1



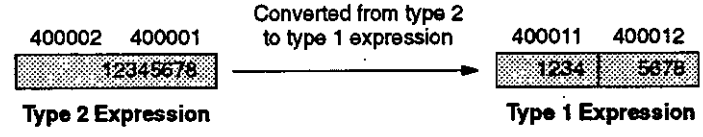
Type 1 Expression — Type 2 Expression

Example 2



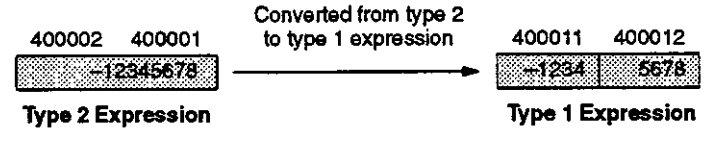
Type 1 Expression — Type 2 Expression

Example 3



Type 2 Expression — Type 1 Expression

Example 4



Type 2 Expression — Type 1 Expression

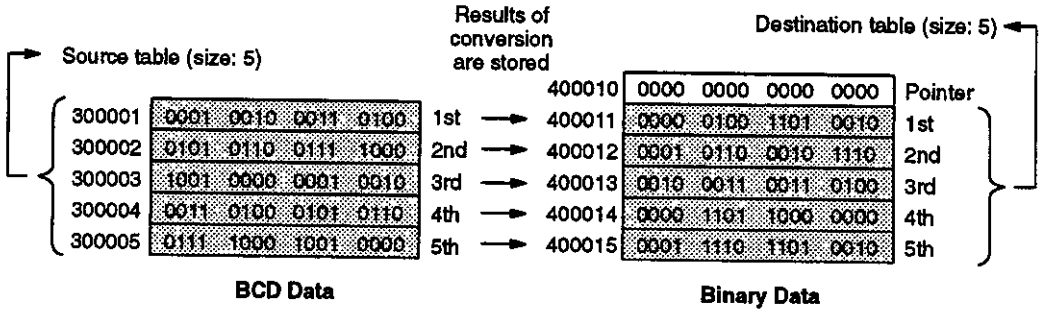
7.2 Details of Data Conversion Instructions

This section describes the function, structures, and operation of each data conversion instruction and provides simple examples of their application.

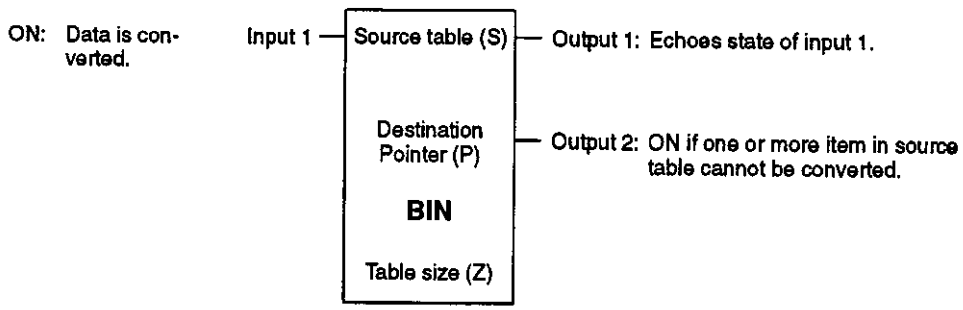
7.2.1 BCD-TO-BINARY CONVERSION (BIN)

1. Function

The data in the registers of the source table is treated as 4-digit BCD data, converted to binary data, and stored in corresponding registers of the destination table. The conversion is completed in one scan.



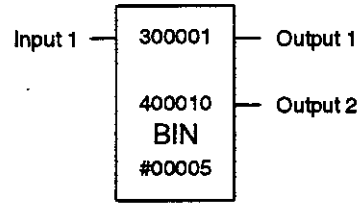
2. Structure



1) BIN is the symbol for BCD-TO-BINARY CONVERSION.

2) BIN requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. Refer to *Table 7.2* for details on specifying constants or registers for these elements.

Example



300001: Leading reference number of source table

400010: Reference number of pointer (Next register after pointer is first register in destination table.)

#00005: Size of source and destination tables (5)

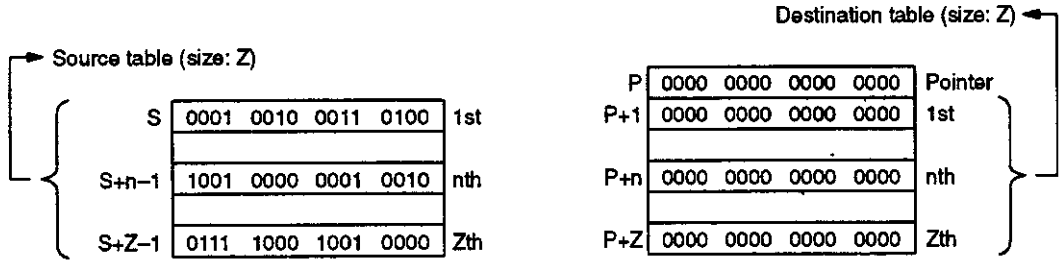
Table 7.2 Structural Elements of BIN

Element	Meaning	Possible Settings	
Top (S)	Leading reference number of source table	Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 or R20001 to R21024
Middle (P)	Reference number of pointer	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 or R20001 to R21023
Bottom (Z)	Size of source and destination tables	Constant:	#00001 to #00016

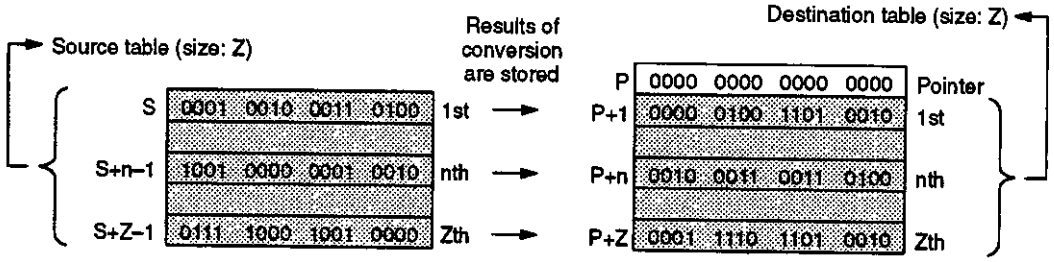
Note The next register after the pointer is the first register in the destination table.

3. Operation

1) Status Before Execution



2) If the data in all of the registers in the source table is 4-digit BCD, the following data conversion will be executed when input 1 turns ON. The conversion is completed in one scan.



a) The data in each register of the source table is converted to binary data and stored in the corresponding register in the destination table.

b) The data in the registers of the source table does not change.

c) The contents of the pointer is set to all zeros.

d) The outputs are set as follows:

(1) Output 1: ON

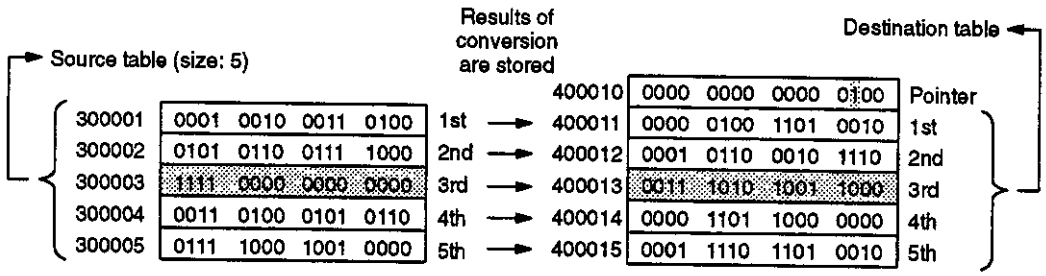
(2) Output 2: OFF

3) If the data in any of the registers in the source table is not 4-digit BCD, the following data conversion will be executed when input 1 turns ON. The conversion is completed in one scan.

a) The data in each register of the source table containing 4-digit BCD is converted to binary data and stored in the corresponding register in the destination table.

b) The data in each register of the source table not containing 4-digit BCD is not converted to binary data, and invalid data (not necessarily all zeros) is stored in the corresponding register in the destination table. The register in the destination table containing invalid data, counting from the first register in the table, is indicated by the value placed in the pointer as shown in the following example.

Example: If the 3rd register in the source table does not contain 4-digit BCD when the conversion instruction is executed, invalid data will be stored in the 3rd register in the destination table, and the 3rd bit in the pointer counting from the LSB will be set to 1.



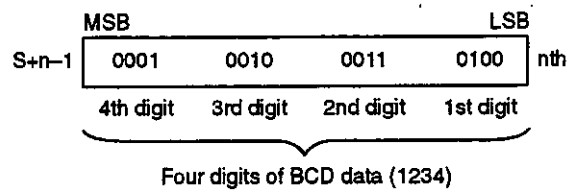
c) The outputs are set as follows:

(1) Output 1: ON

(2) Output 2: ON

4) Data is converted as described next.

a) The data in each register of the source table is handled as 4-digit BCD as shown in the following illustration.



b) Each digit of the BCD data is converted to binary data as shown in the following table.

Table 7.3 BCD-TO-BINARY CONVERSION

Data Before Conversion		Converted Data (Decimal Notation)	Data Before Conversion		Converted Data (Decimal Notation)
Digit	Bit Pattern		Digit	Bit Pattern	
4th	0000	0	2nd	0000	0
	0001	1000		0001	10
	0010	2000		0010	20
	0011	3000		0011	30
	0100	4000		0100	40
	0101	5000		0101	50
	0110	6000		0110	60
	0111	7000		0111	70
	1000	8000		1000	80
	1001	9000		1001	90
	Other	Invalid data (unpredictable)		Other	Invalid data (unpredictable)
3rd	0000	0	1st	0000	0
	0001	100		0001	1
	0010	200		0010	2
	0011	300		0011	3
	0100	400		0100	4
	0101	500		0101	5
	0110	600		0110	6
	0111	700		0111	7
	1000	800		1000	8
	1001	900		1001	9
	Other	Invalid data (unpredictable)		Other	Invalid data (unpredictable)

c) The binary values for the converted digits are added and the result is stored in the corresponding register of the destination table.

d) An example of the conversion process is provided next for the nth register of the source table.

(1) The BCD contents of the nth register of the source table is as follows:

S+n–1	0001	0010	0011	0100	nth	**BCD Data**
	4th digit	3rd digit	2nd digit	1st digit		

(2) Each digit of BCD data is converted to binary and then added.
Sum = 1000 + 200 + 30 + 4 (decimal)
= 1234 (decimal)
= 0000 0100 1101 0010 (binary)

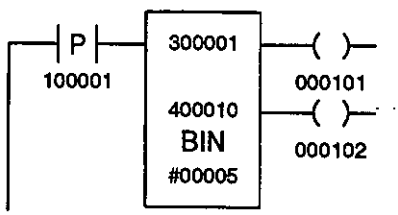
(3) The sum is stored in the nth register of the destination table.

	MSB			LSB		
P+n	0000	0100	1101	0010	nth	**Binary Data**

4. Application Examples

◀EXAMPLE▶ **Example 1**

1) Ladder Programming



2) Conversion Process

a) Before Conversion

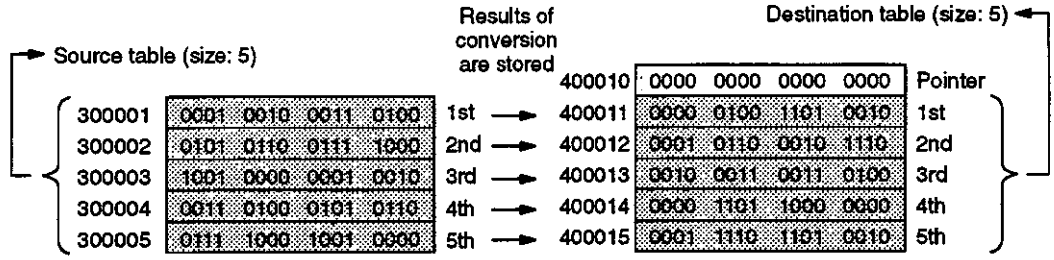
Source table (size: 5)

300001	0001	0010	0011	0100	1st
300002	0101	0110	0111	1000	2nd
300003	1001	0000	0001	0010	3rd
300004	0011	0100	0101	0110	4th
300005	0111	1000	1001	0000	5th

Destination table (size: 5)

400010	0000	0000	0000	0000	Pointer
400011	0000	0000	0000	0000	1st
400012	0000	0000	0000	0000	2nd
400013	0000	0000	0000	0000	3rd
400014	0000	0000	0000	0000	4th
400015	0000	0000	0000	0000	5th

b) The following data conversion will be performed when input relay 100001 changes from OFF to ON. The conversion will be completed in one scan.



(1) The binary conversion of the data in each register in the source table will be stored in the corresponding register in the destination table.

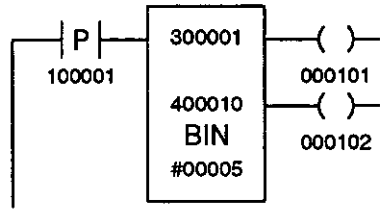
(2) The data in the registers of the source table does not change.

(3) The value of the pointer remains as all zeros.

(4) The status of the coils is as follows:
Coil 000101: ON only for the scan in which input relay 100001 changed from OFF to ON.
Coil 000102: Remains OFF.

◀EXAMPLE▶ **Example 2**

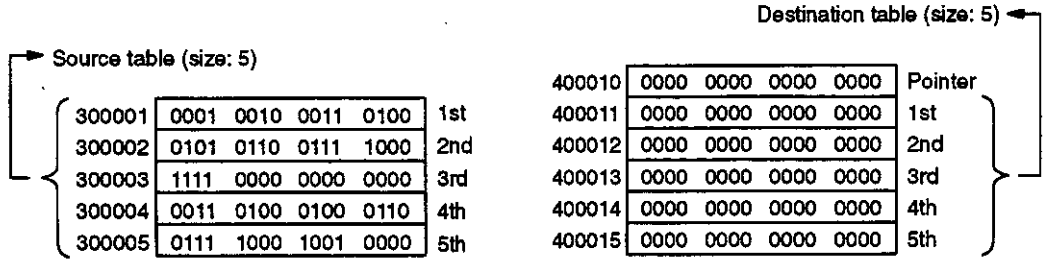
1) Ladder Programming



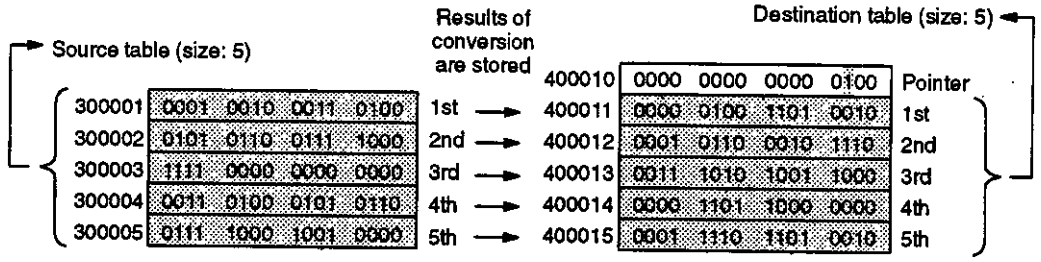
2) Conversion Process

a) Before Conversion

The data in the 3rd register of the source table is not 4-digit BCD.



b) The following data conversion will be performed when input relay 100001 changes from OFF to ON. The conversion will be completed in one scan.



(1) The binary conversion of the data in each register in the source table, except for the 3rd register, is stored in the corresponding register in the destination table.

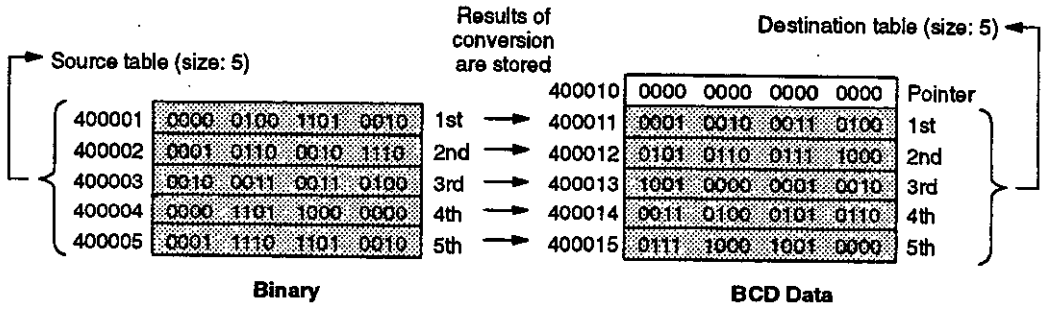
(2) The data in the 3rd register of the source table will be not 4-digit BCD so it will be not converted to binary. Invalid data will be stored in the 3rd register of the destination table and the 3rd bit from the LSB of the pointer will be set to 1 to indicate that the 3rd register in the destination table contains invalid data.

(3) Coil 000101 and coil 000102 turn ON only for the scan in which input relay 100001 changed from OFF to ON.

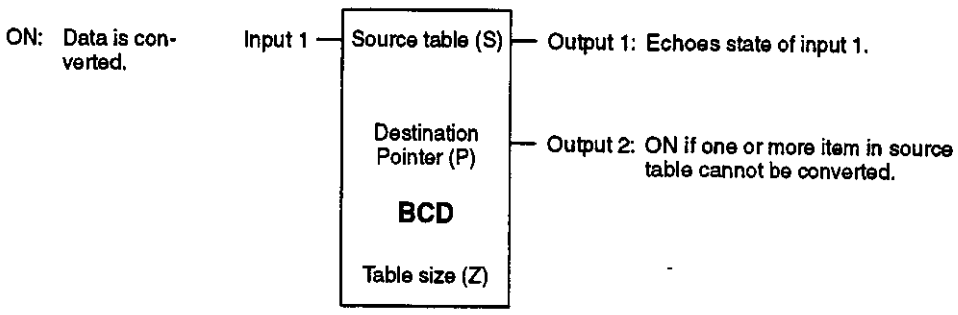
7.2.2 BINARY-TO-BCD CONVERSION (BCD)

1. Function

The data in the registers of the source table is converted from binary data to 4-digit BCD data and stored in corresponding registers of the destination table. The conversion is completed in one scan.



2. Structure



1) BCD is the symbol for BINARY-TO-BCD CONVERSION.

2) BCD requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. Refer to *Table 7.4* for details on specifying constants or registers for these elements.

Example

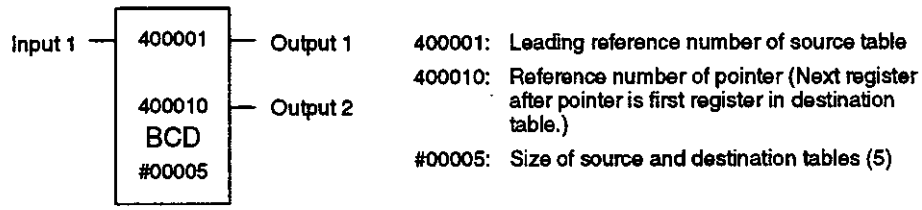


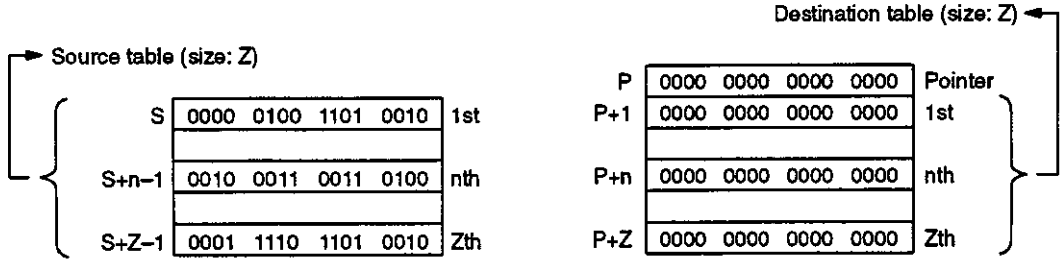
Table 7.4 Structural Elements of BCD

Element	Meaning	Possible Settings	
Top (S)	Leading reference number of source table	Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 or R20001 to R21024
Middle (P)	Reference number of pointer	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 or R20001 to R21023
Bottom (Z)	Size of source and destination tables	Constant:	#00001 to #00016

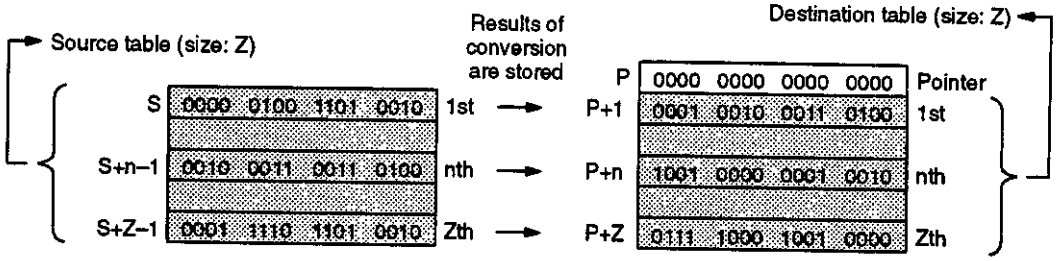
Note The next register after the pointer is the first register in the destination table.

3. Operation

1) Status Before Execution



2) If the data in all of the registers in the source table can be converted to 4-digit BCD, the following data conversion will be executed when input 1 turns ON. The conversion is completed in one scan.



a) The data in each register of the source table is converted to 4-digit BCD data and stored in the corresponding register in the destination table.

b) The data in the registers of the source table does not change.

c) The contents of the pointer is set to all zeros.

d) The outputs are set as follows:

(1) Output 1: ON

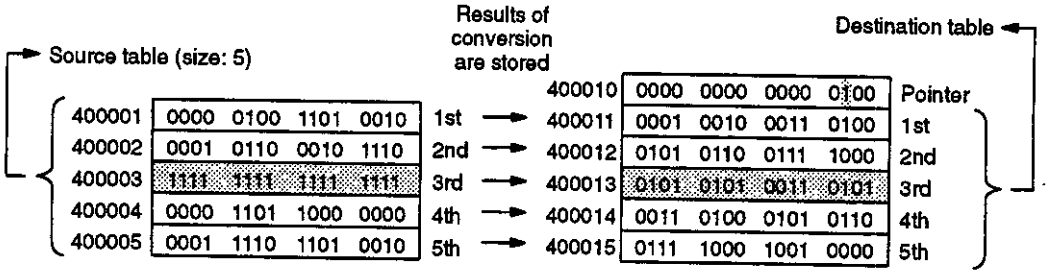
(2) Output 2: OFF

3) If the data in any of the registers in the source table cannot be converted to 4-digit BCD, the following data conversion will be executed when input 1 turns ON. The conversion is completed in one scan.

a) The data in each register of the source table that can be converted to 4-digit BCD is converted to 4-digit BCD and stored in the corresponding register in the destination table.

b) The data in each register of the source table that cannot be converted to 4-digit BCD is not converted, and invalid data (not necessarily all zeros) is stored in the corresponding register in the destination table. The register in the destination table containing invalid data, counting from the first register in the table, is indicated by the value placed in the pointer as shown in the following example.

Example: If the data in the 3rd register in the source table cannot be converted to 4-digit BCD when the conversion instruction is executed, invalid data will be stored in the 3rd register in the destination table, and the third bit in the pointer counting from the LSB will be set to 1.



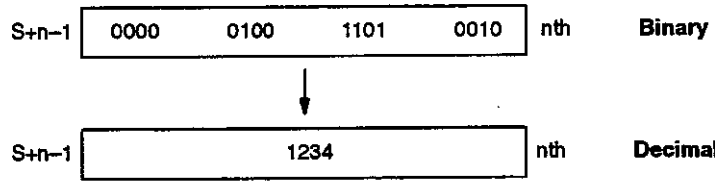
c) The outputs are set as follows:

(1) Output 1: ON

(2) Output 2: ON

4) Data is converted as described next.

a) The data in each register of the source table is handled as 4 digits as shown in the following illustration.

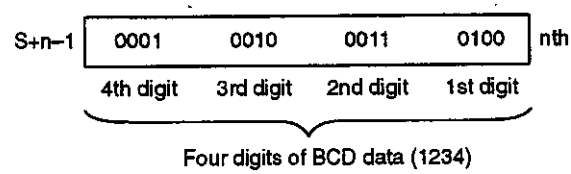


b) Each digit is converted to BCD data as shown in the following table.

Table 7.5 BINARY-TO-BCD CONVERSION

Data Before Conversion (as Decimal)	Converted Data (Bit Pattern)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

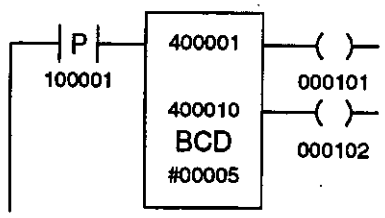
c) The bit patterns resulting from conversion are stored in the corresponding registers of the destination table as shown below.



4. Application Examples

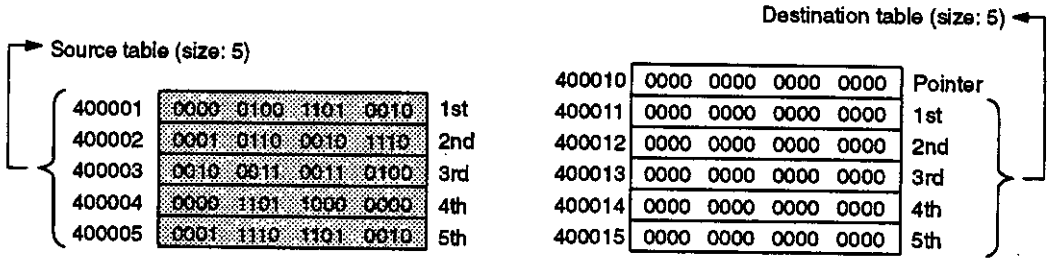
◀EXAMPLE▶ **Example 1**

1) Ladder Programming

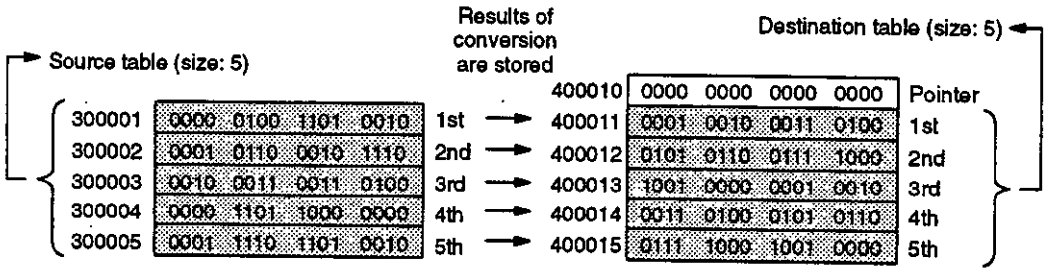


2) Conversion Process

a) Before Conversion



b) The following data conversion will be performed when input relay 100001 changes from OFF to ON. The conversion will be completed in one scan.



(1) The data in each register in the source table will be converted to 4-digit BCD and stored in the corresponding register in the destination table.

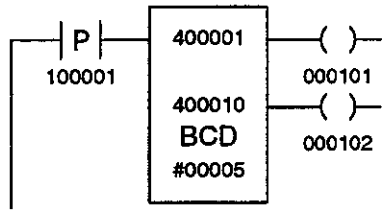
(2) The data in the registers of the source table does not change.

(3) The value of the pointer remains as all zeros.

(4) The status of the coils is as follows:
Coil 000101: ON only for the scan in which input relay 100001 changed from OFF to ON.
Coil 000102: Remains OFF.

◀EXAMPLE▶ **Example 2**

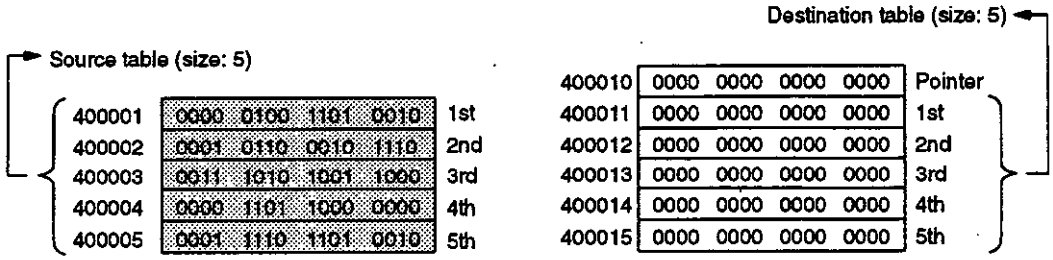
1) Ladder Programming



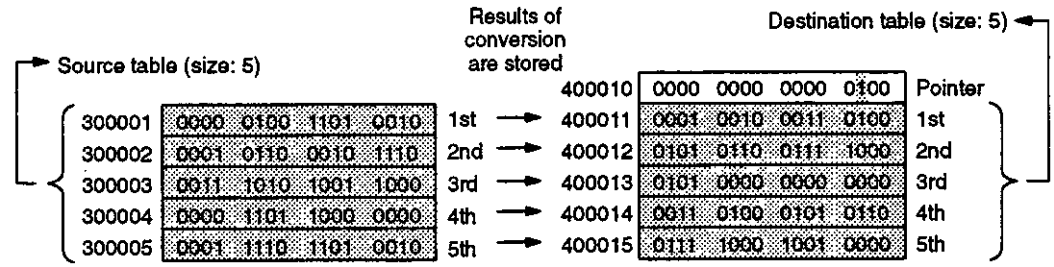
2) Conversion Process

a) Before Conversion

The data in the 3rd register of the source table cannot be converted to 4-digit BCD.



b) The following data conversion will be performed when input relay 100001 changes from OFF to ON. The conversion will be completed in one scan.



(1) The data in any register other then the 3rd one in the source table is converted to 4-digit BCD and stored in the corresponding register in the destination table.

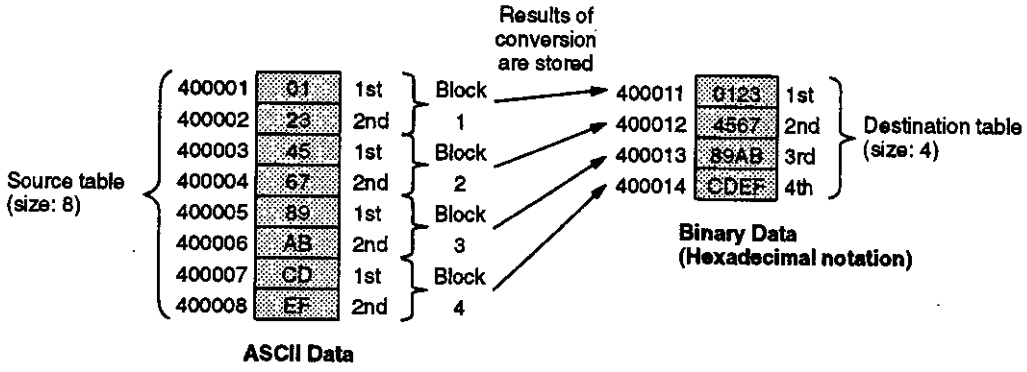
(2) The data in the 3rd register of the source cannot be converted to 4-digit BCD so it will be not converted. Invalid data will be stored in the 3rd register of the destination table and the 3rd bit from the LSB of the pointer will be set to 1 to indicate that the 3rd register in the destination table contains invalid data.

(3) Coil 000101 and coil 000102 will turn ON only for the a scan in which input relay 100001 went from OFF to ON.

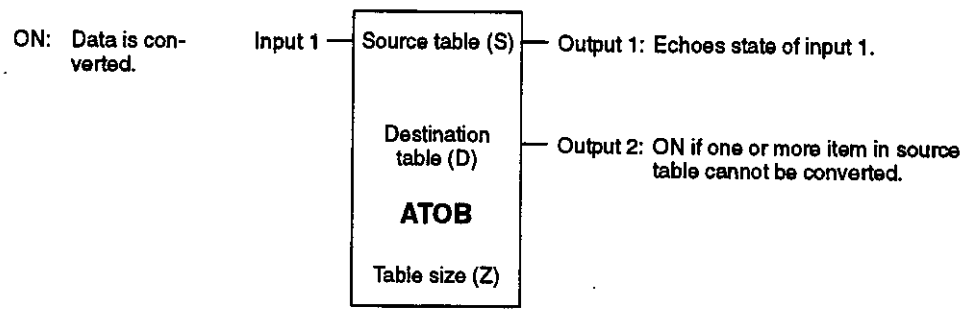
7.2.3 ASCII-TO-BINARY CONVERSION (ATOB)

1. Function

The data in each pair of registers forming a block in the source table is treated as four ASCII characters and converted to 4-bit binary data and stored in corresponding registers of the destination table. The conversion is completed in one scan. The only ASCII characters that can be converted are 0 to 9 and A to F.



2. Structure



1) ATOB is the symbol for ASCII-TO-BINARY CONVERSION.

2) ATOB requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. Refer to *Table 7.6* for details on specifying constants or registers for these elements.

Example

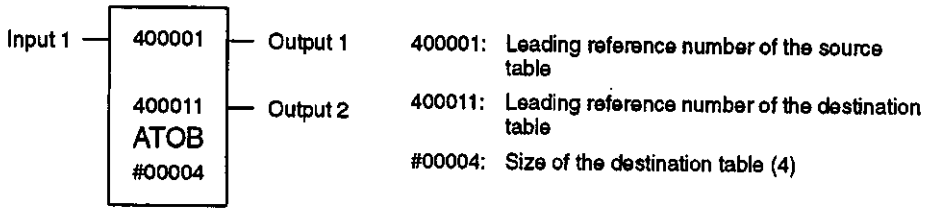


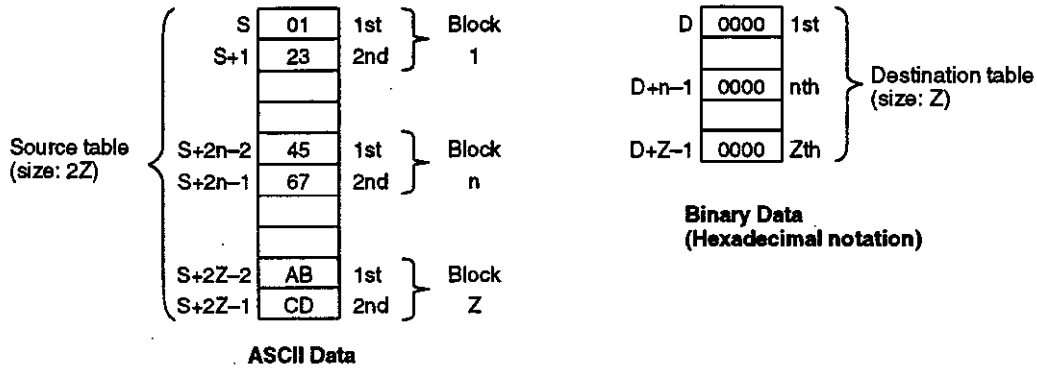
Table 7.6 Structural Elements of ATOB

Element	Meaning	Possible Settings	
Top (S)	Leading reference number of source table	Input register:	300001 to 300511 (Z00001 to Z00511)
		Holding register:	400001 to 409998 (W00001 to W09998)
		Constant register:	700001 to 704095 (K00001 to K04095)
		Link register:	R10001 to R11023 R20001 to R21023
Middle (D)	Leading reference number of destination table	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 R20001 to R21024
Bottom (Z)	Size of destination table	Constant:	#00001 to #00100

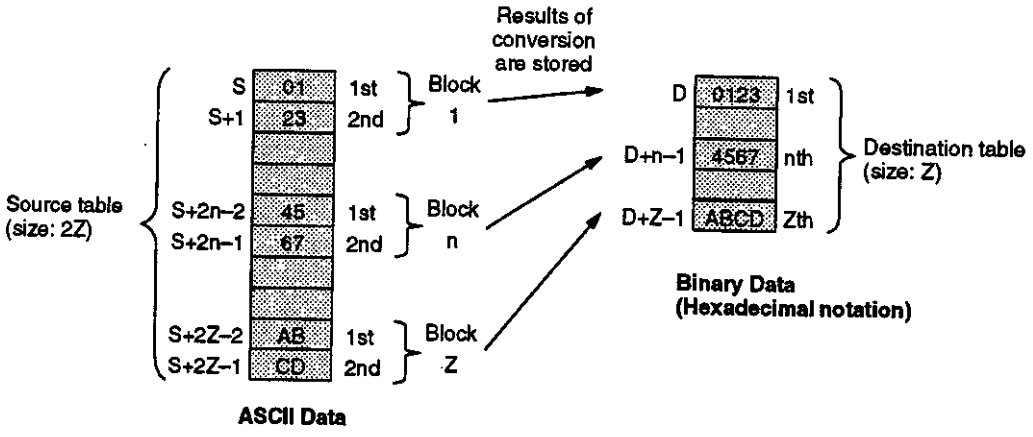
Note The source table is twice the size of the destination table.

3. Operation

1) Status Before Execution



2) If the ASCII data (0 to 9, A to F) in all of the blocks in the source table can be converted, the following data conversion will be executed when input 1 turns ON. The conversion will be completed in one scan.



a) The data in each block of the source table (4 ASCII characters) is converted to binary data and stored in the corresponding register in the destination table (4 sets of 4-bit binary data). The data converted for block n of the source table is stored in register n of the destination table.

b) The data in the blocks of the source table does not change.

c) The outputs are set as follows:

(1) Output 1: ON

(2) Output 2: OFF

3) If the ASCII data in any of the blocks in the source table cannot be converted (i.e., contains any characters except 0 to 9 and A to F), the following (partial) data conversion will be executed when input 1 turns ON. The conversion will be completed in one scan.

a) The ASCII data in each block is converted from the beginning block and stored in corresponding registers in the destination tables until the first block that cannot be converted is discovered.

b) No other blocks are converted after a block that cannot be converted is discovered.

c) The data in the blocks of the source table does not change.

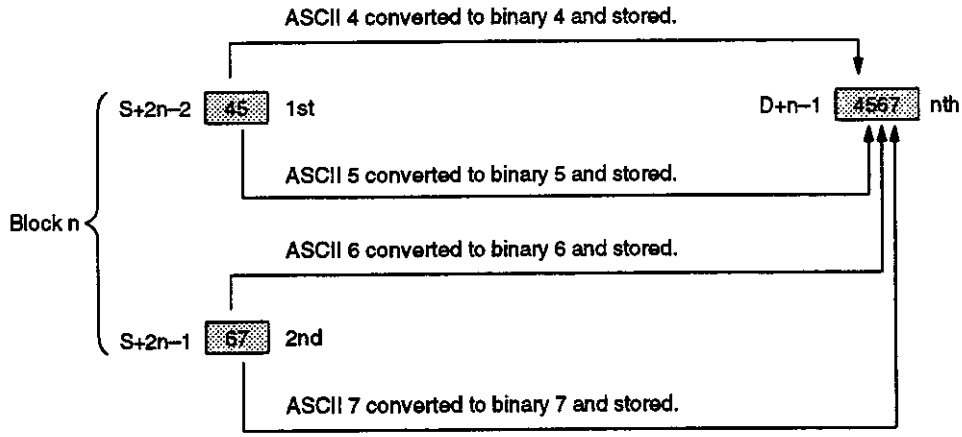
d) The outputs are set as follows:

(1) Output 1: ON

(2) Output 2: ON

4) Data is converted as described next.

a) As shown below, each 4 ASCII characters are converted to 4 sets of 4-bit binary data (i.e., 4 hexadecimal characters) and stored in the corresponding register in the destination table. The data converted for block n of the source table is stored in register n of the destination table.



b) Each ASCII character is converted to 4-bit binary data as shown in the following table.

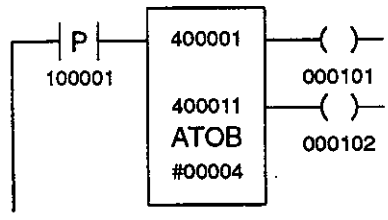
Table 7.7 ASCII-TO-BINARY CONVERSION

Data Before Conversion		Converted Data	Data Before Conversion		Converted Data
ASCII Notation	Hexadecimal Notation	Hexadecimal Notation	ASCII Notation	Hexadecimal Notation	Hexadecimal Notation
0	30	0	8	38	8
1	31	1	9	39	9
2	32	2	A	41	A
3	33	3	B	42	B
4	34	4	C	43	C
5	35	5	D	44	D
6	36	6	E	45	E
7	37	7	F	46	F

4. Application Examples

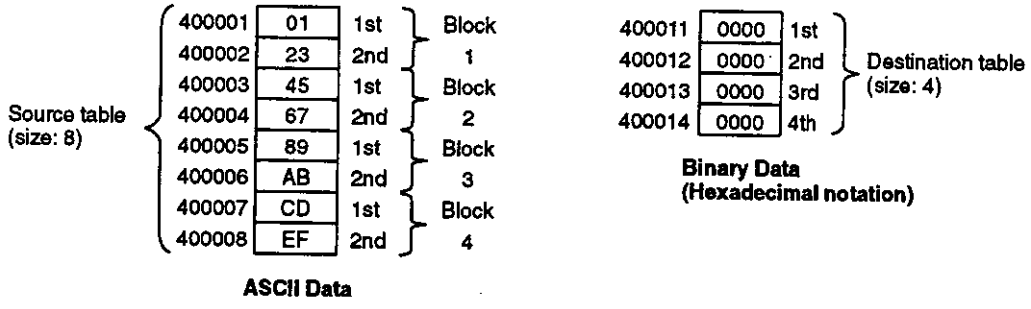
◀EXAMPLE▶ **Example 1**

1) Ladder Programming

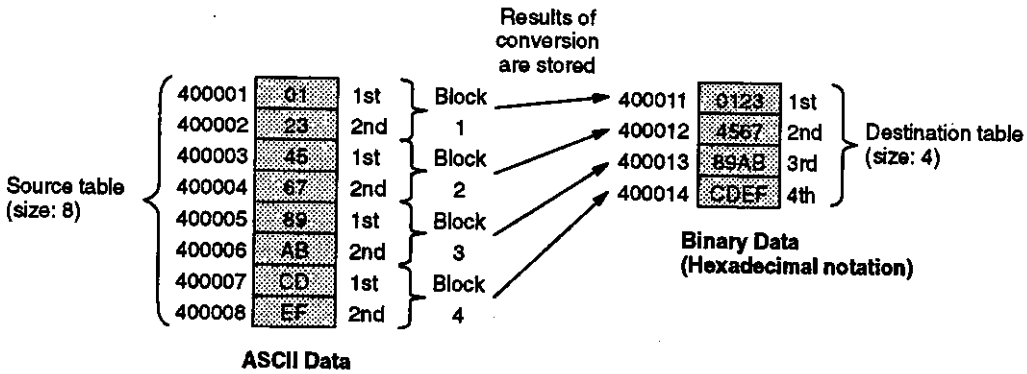


2) Conversion Process

a) Before Conversion



b) The following data conversion will be performed when input relay 100001 changes from OFF to ON. The conversion will be completed in one scan.



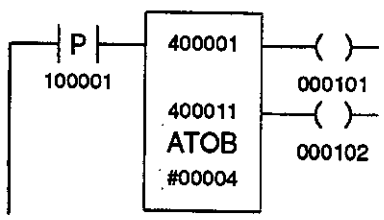
(1) The data in each block of the source table (4 ASCII characters) is converted to 4-bit binary data and stored in the corresponding register in the destination table (4 sets of 4-bit binary data). The data converted for block n of the source table is stored in register n of the destination table (n = 1 to 4).

(2) The data in the blocks of the source table does not change.

(3) The status of the coils is as follows:
Coil 000101: ON only for the scan in which input relay 100001 changed from OFF to ON.
Coil 000102: Remains OFF.

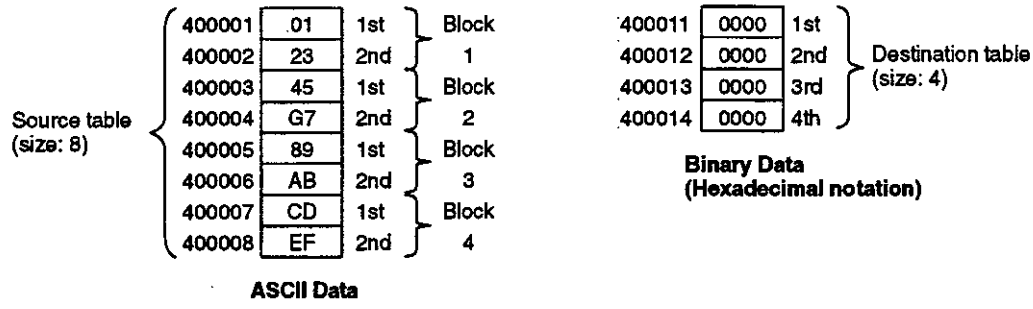
◀EXAMPLE▶ **Example 2**

1) Ladder Programming

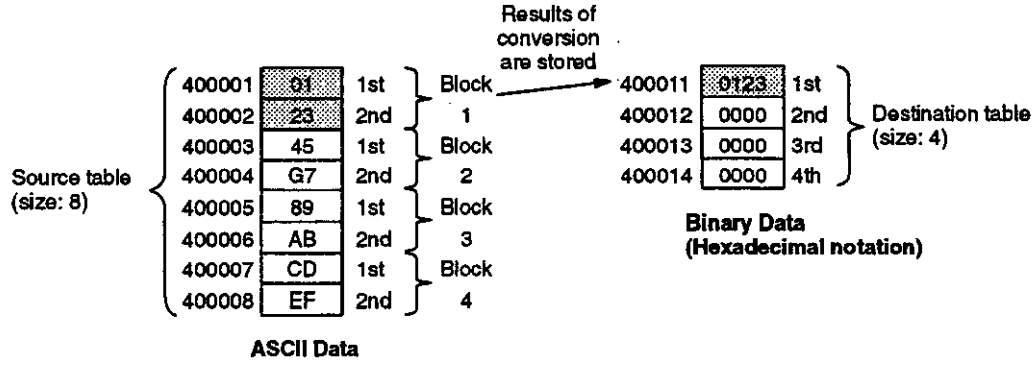


2) Conversion Process

a) Before Conversion



b) The following data conversion will be performed when input relay 100001 changes from OFF to ON. The conversion will be completed in one scan.



(1) The data in block 1 of the source table (4 ASCII characters) is converted to binary data and stored in register 1 in the destination table (4-bit binary data).

(2) The ASCII data (G) in the 2nd register in block 2 of the source table cannot be converted; therefore conversion does not carried out for the rest of the blocks in the source table.

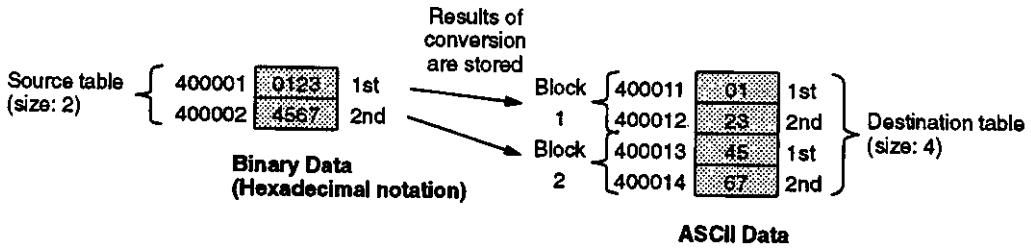
(3) The data in the blocks of the source table does not change.

(4) Coil 000101 and coil 000102 will turn ON only for the scan in which input relay 100001 changed from OFF to ON.

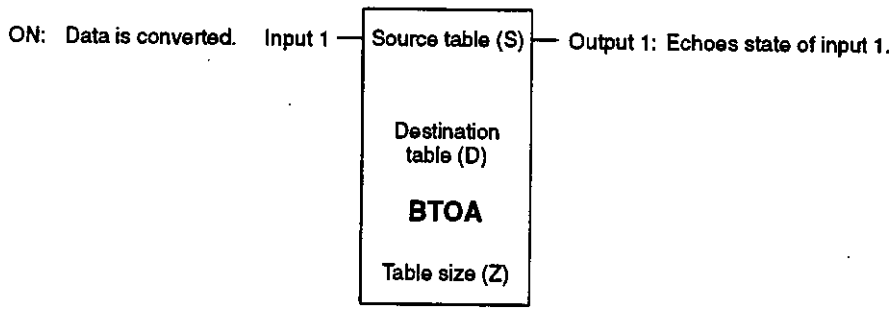
7.2.4 BINARY-TO-ASCII CONVERSION (BTOA)

1. Function

The data in each register of the source table is separated into 4 sets of 4-bit binary data, converted into four ASCII characters, and stored in corresponding blocks of the destination table. The conversion is completed in one scan.



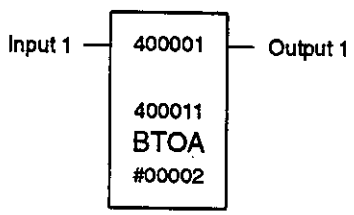
2. Structure



1) BTOA is the symbol for BINARY-TO-ASCII CONVERSION.

2) BTOA requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. Refer to *Table 7.8* for details on specifying constants or registers for these elements.

Example



400001: Leading reference number of the source table

400011: Leading reference number of the destination table

#00002: Size of the source table (2)

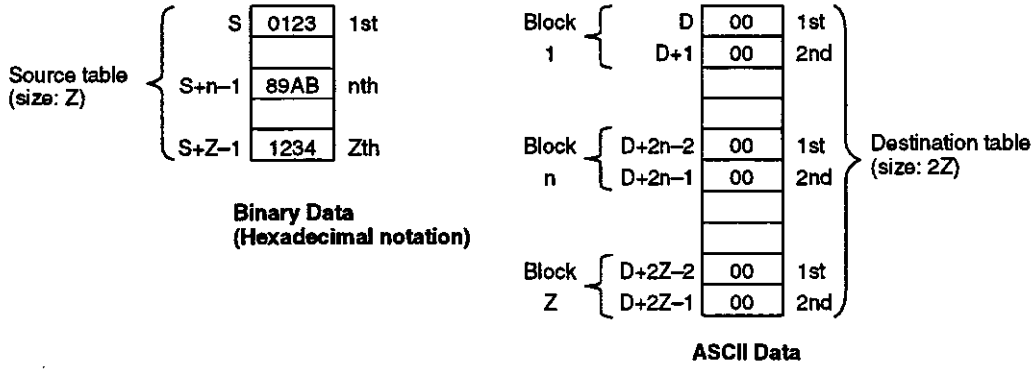
Table 7.8 Structural Elements of BTOA

Element	Meaning	Possible Settings	
Top (S)	Leading reference number of source table	Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 R20001 to R21024
Middle (D)	Leading reference number of destination table	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 R20001 to R21023
Bottom (Z)	Size of source table	Constant:	#00001 to #00100

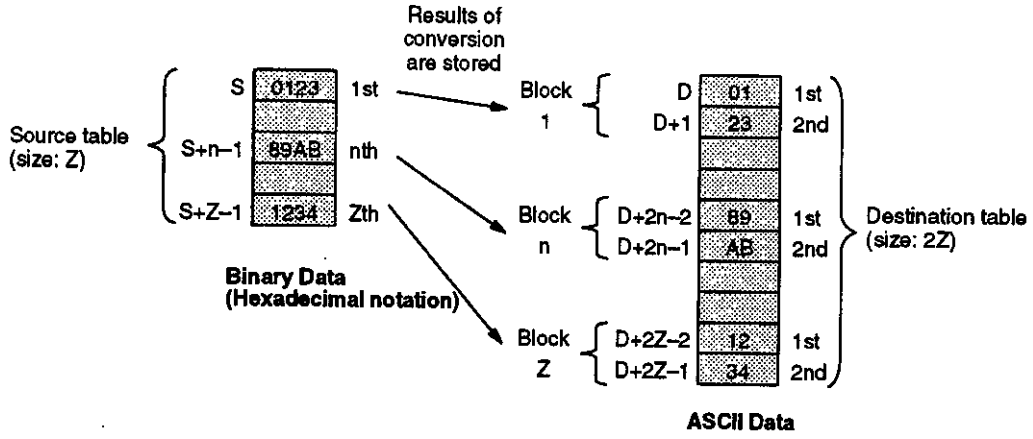
Note The destination table is twice the size of the source table.

3. Operation

1) Status Before Execution



2) The following data conversion will be executed when input 1 turns ON. The conversion will be completed in one scan.

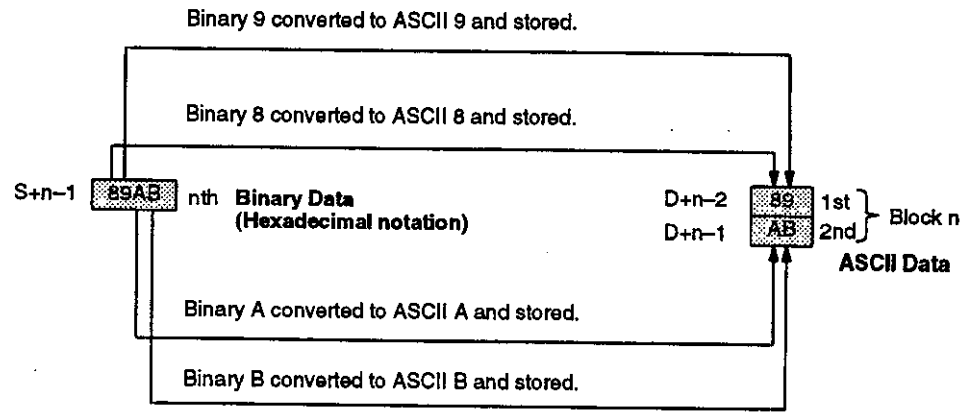


a) The data in each register of the source table is separated into 4 sets of 4-bit binary data, converted into four ASCII characters, and stored in corresponding blocks of the destination table. The data converted for register n of the source table is stored in block n of the destination table.

b) The data in the registers of the source table does not change.

c) The output 1 turns ON.

3) Data is converted as shown below.



4) Each 4 bits of binary data are converted to ASCII as shown in the following table.

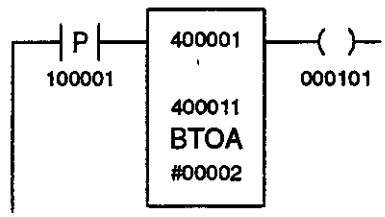
Table 7.9 BINARY-TO-ASCII CONVERSION

Data Before Conversion (HEX Notation)	Converted Data (HEX Notation)	Data Before Conversion (HEX Notation)	Converted Data (HEX Notation)
0	30	8	38
1	31	9	39
2	32	A	41
3	33	B	42
4	34	C	43
5	35	D	44
6	36	E	45
7	37	F	46

◀EXAMPLE▶

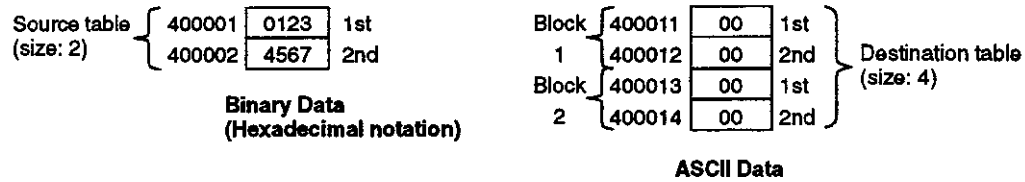
4. Application Example

1) Ladder Programming

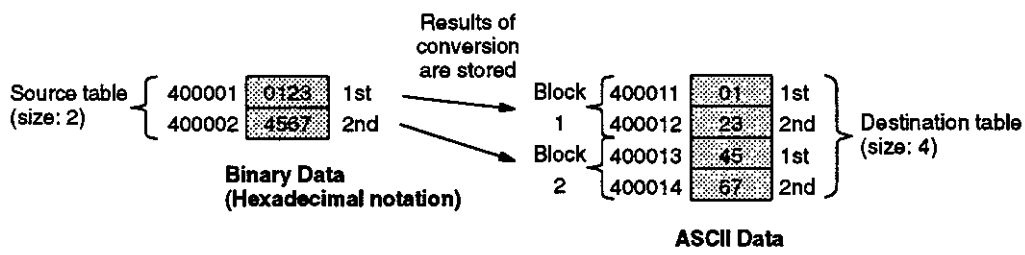


2) Conversion Process

a) Before Conversion



b) The following data conversion will be performed when input relay 100001 changes from OFF to ON. The conversion will be completed in one scan.



(1) The data in each register of the source table is separated into 4 sets of 4-bit binary data, converted into four ASCII characters, and stored in corresponding blocks of the destination table. The data converted for register n of the source table is stored in block n of the destination table.

(2) The data in each register of the source table does not change.

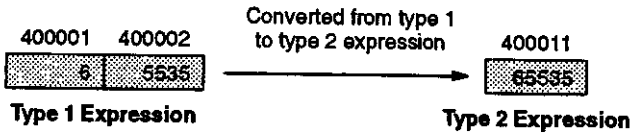
(3) Coil 000101 turns ON only for the scan in which input relay 100001 changed from OFF to ON.

7.2.5 16-BIT CONVERSION (CAST)

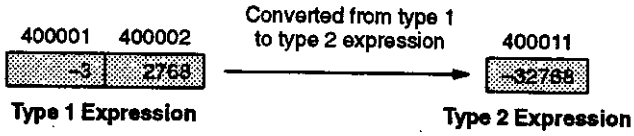
1. Function

The numeric expression of a 16-bit binary integer is changed from type 1 to type 2 or from type 2 to type 1. The conversion is completed in one scan.

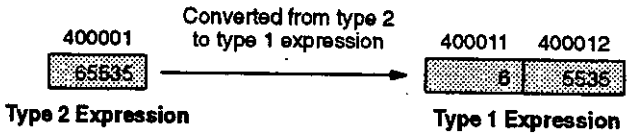
Example 1



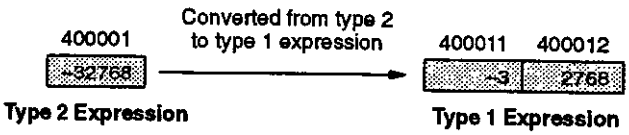
Example 2



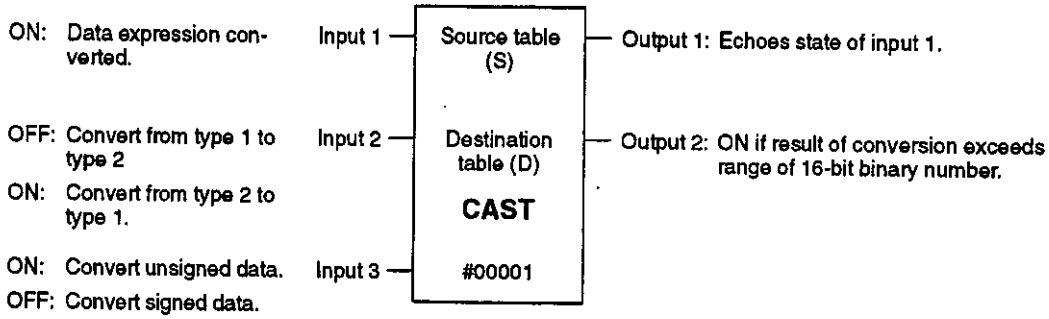
Example 3



Example 4



2. Structure



1) CAST is the symbol for 16-BIT CONVERSION.

2) CAST requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. Refer to *Table 7.10* for details on specifying constants or registers for these elements.

Example

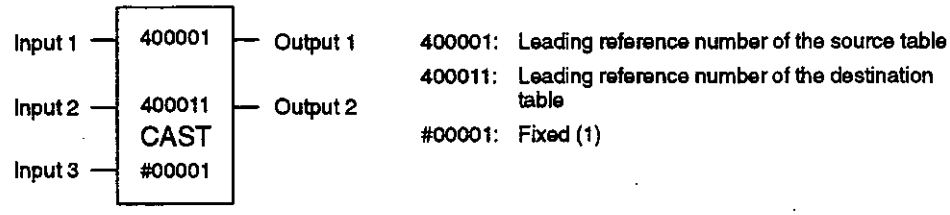


Table 7.10 Structural Elements of CAST

Element	Meaning	Possible Settings	
Top (S)	Leading reference number of source table (size:2)	Input register:	300001 to 300511 (Z00001 to Z00511)
		Holding register:	400001 to 409998 (W00001 to W09998)
		Constant register:	700001 to 704095 (K00001 to K04095)
		Link register:	R10001 to R11023 R20001 to R21023
Middle (D)	Leading reference number of destination table (size:2)	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 R20001 to R21023
Bottom	Fixed	Constant:	#00001

3. Operation

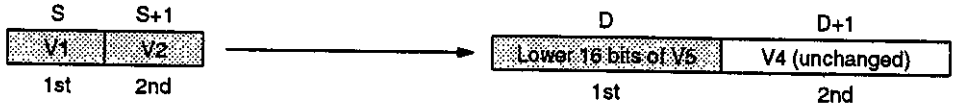
1) Status Before Execution

2) The following data conversion will be executed when input 1 turns ON and inputs 2 and 3 are OFF. The conversion will be completed in one scan.

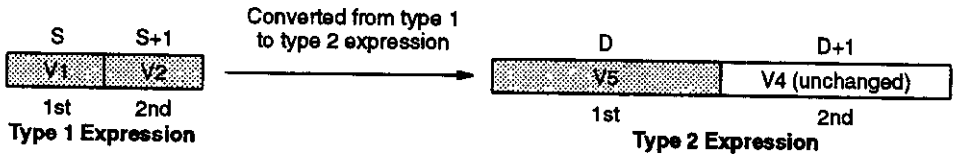
a) The data in the two registers of the source table (V1 and V2) is taken as an unsigned 5-digit decimal integer (0 to 65,535) in a type 1 expression and converted to an unsigned 9-digit decimal integer (0 to 655,415,535) (V5).

- V5 = 10,000 x V1 + V2

b) The lower 16 bits of V5 are stored in the leading register of the destination table. The upper 16 bits are discarded.



c) Thus, if the data in the two registers of the source table (V1 and V2) is an unsigned 5-digit decimal integer in a type 1 expression (0 to 65,535), it will be converted to an unsigned 9-digit decimal integer in a type 2 expression and the result will be stored in the leading register of the destination table.



d) The data in the registers of the source table and the data of the second register in the destination table does not change.

e) The status of the outputs is as follows as long as input 1 is ON:

(1) Output 1: ON

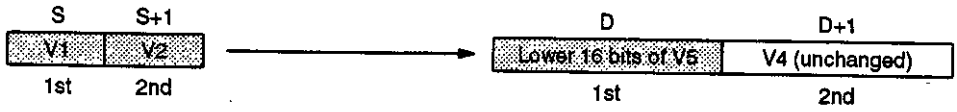
(2) Output 2: OFF if V5 is 65,535 or less; ON if V5 is greater than 65,535.

3) The following data conversion will be executed when inputs 1 and 3 turn ON and input 2 is OFF. The conversion will be completed in one scan.

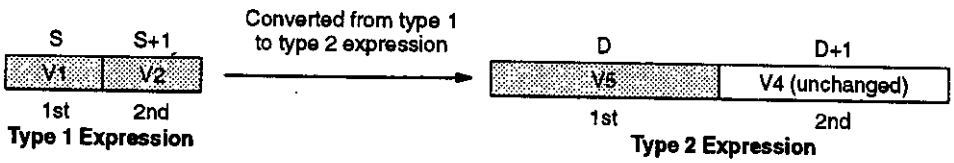
a) The data in the two registers of the source table (V1 and V2) is taken as a signed 5-digit decimal integer (–32,768 to 32,767) in a type 1 expression and converted to a signed 10-digit decimal integer (–2,147,483,648 to 2,147,483,647) (V5).

- If V1 ≥ 0, then V5 = 10,000 x |V1| + |V2|
- If V1 < 0, then V5 = –(10,000 x |V1| + |V2|)

b) The lower 16 bits of V5 are stored in the leading register of the destination table. The upper 16 bits are discarded.



c) Thus, if the data in the two registers of the source table (V1 and V2) is a signed 5-digit decimal integer in a type 1 expression (–32,768 to 32,767), it will be converted to a signed 10-digit decimal integer in a type 2 expression and the result will be stored in the leading register of the destination table.



d) The data in the registers of the source table and the second register in the destination table does not change.

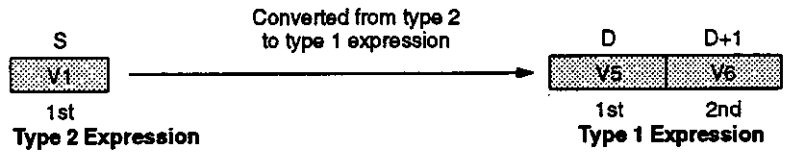
e) The status of the outputs is as follows as long as input 1 is ON:

(1) Output 1: ON

(2) Output 2: OFF if $-32{,}768 \leq V5 \leq 32{,}767$; ON otherwise.

4) The following data conversion will be executed when inputs 1 and 2 turn ON and input 3 is OFF. The conversion will be completed in one scan.

a) The data in the leading register of the source table (V1) is taken as an unsigned 5-digit decimal integer (0 to 65,535) in a type 2 expression, it is converted to a type 1 expression, and the result is stored in the two registers of the destination table (V5 and V6).



b) The following equation is used.

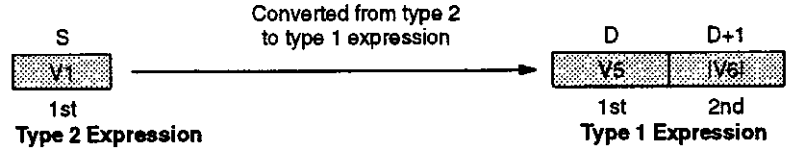
• V1 ÷ 10,000 = V5 with a remainder of V6

c) The data in the registers of the source table does not change.

d) Output 1 is ON and output 2 remains OFF as long as input 1 is ON.

5) The following data conversion will be executed when inputs 1, 2, and 3 turn ON. The conversion will be completed in one scan.

a) The data in the leading register of the source table (V1) is taken as a signed 5-digit decimal integer (–32,768 to 32,767) in a type 2 expression, it is converted to a type 1 expression, and the result is stored in the two registers of the destination table (V5 and V6).



b) The following equation is used.

• V1 ÷ 10,000 = V5 with a remainder of V6

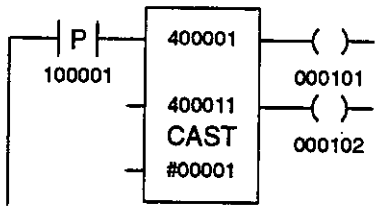
c) The data in the registers of the source table does not change.

d) Output 1 is ON and output 2 remains OFF as long as input 1 is ON.

4. Application Examples

◀EXAMPLE▶ **Example 1:**
An unsigned, 5-digit decimal integer in a type 1 expression is converted to a type 2 expression.

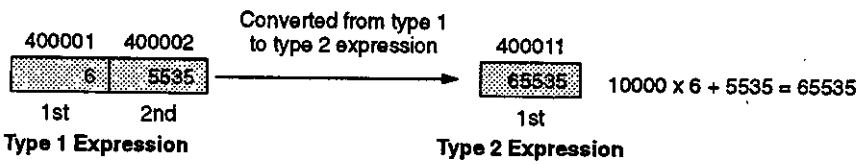
1) Ladder Programming



2) Conversion Process

a) Before Conversion

b) The following data conversion will be performed when input relay 100001 changes from OFF to ON. The conversion will be completed in one scan.



(1) The data in the two registers of the source table (V1 and V2) is taken as an unsigned 5-digit decimal integer (65,535) in a type 1 expression, it is converted to a type 2 expression, and the result is stored in the leading register of the destination table.

(2) The data in the registers of the source table and the data in the second register in the destination table does not change.

(3) The status of the coils is as follows:

Coil 000101: ON only for the scan in which input relay 100001 changed from OFF to ON

Coil 000102: OFF

c) Thus, the above program section can be used to convert an unsigned 5-digit decimal integer in a type 1 expression (0 to 65,535) to a type 2 expression.

d) If the value of the two registers in the source table exceeds 65,535 when taken as an unsigned 5-digit decimal integer in a type 1 expression, the following processing will be performed when input relay 100001 turns ON.

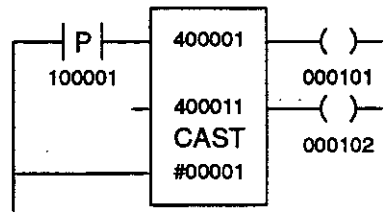
(1) The lower 16 bits of the conversion results is stored in the leading register of the destination table and the upper 16 bits is discarded.

(2) The data in the registers of the source table and the data of the second register in the destination table does not change.

(3) Coil 000101 and coil 000102 turns ON only for the scan in which input relay 100001 changed from OFF to ON.

◀EXAMPLE▶ **Example 2:**
A signed, 5-digit decimal integer in a type 1 expression is converted to a type 2 expression.

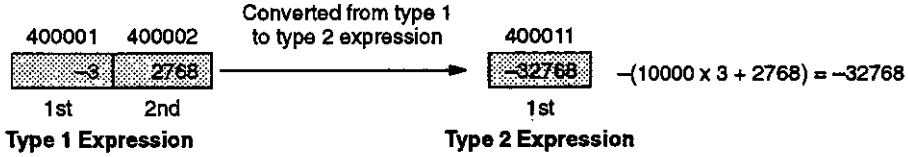
1) Ladder Programming



2) Conversion Process

a) Before Conversion

b) The following data conversion will be performed when input relay 100001 changes from OFF to ON. The conversion will be completed in one scan.



(1) The data in the two registers of the source table (V1 and V2) is taken as a signed 5-digit decimal integer (–32,768) in a type 1 expression, it is converted to a type 2 expression, and the result is stored in the leading register of the destination table.

(2) The data in the registers of the source table and the data of the second register in the destination table does not change.

(3) The status of the coils is as follows:

Coil 000101: ON only for the scan in which input relay 100001 changed from OFF to ON

Coil 000102: OFF

c) Thus, the above program section can be used to convert a signed 5-digit decimal integer in a type 1 expression (–32,768 to 32,767) to a type 2 expression.

d) If the value of the two registers of the source table is not between –32,768 and 32,767, inclusive, when taken as a signed 5-digit decimal integer in a type 1 expression, the following processing will be performed when the input relay 100001 turns ON.

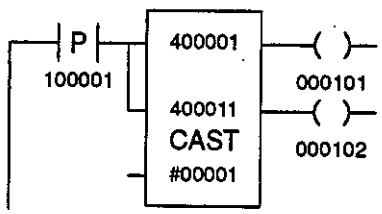
(1) The lower 16 bits of the conversion results is stored in the leading register of the destination table and the upper 16 bits is discarded.

(2) The data in the registers of the source table and the data of the second register in the destination table does not change.

(3) Coil 000101 and coil 000102 turns ON only for the scan in which input relay 100001 changed from OFF to ON.

◀EXAMPLE▶ **Example 3:**
An unsigned, 5-digit decimal integer in a type 2 expression is converted to a type 1 expression.

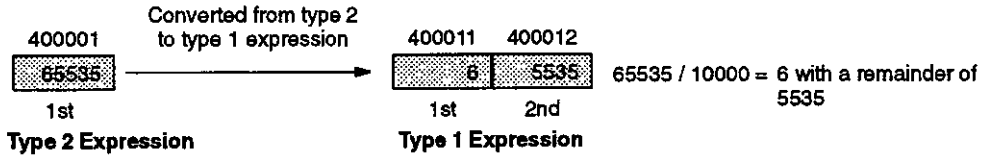
1) Ladder Programming



2) Conversion Process

a) Before Conversion

b) The following data conversion will be performed when input relay 100001 changes from OFF to ON. The conversion will be completed in one scan.



(1) The data in the leading register of the source table (V1) is taken as an unsigned 5-digit decimal integer (65,535) in a type 2 expression, it is converted to a type 1 expression, and the result is stored in the two registers of the destination table.

(2) The data in the registers of the source table does not change.

(3) The status of the coils is as follows:

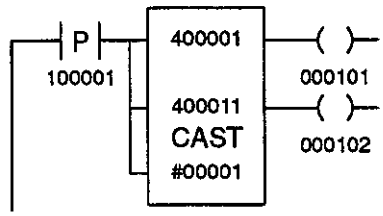
Coil 000101: ON only for the scan in which input relay 100001 changed from OFF to ON

Coil 000102: OFF

c) Thus, the above program section can be used to convert an unsigned 5-digit decimal integer in a type 2 expression (0 to 65,535) to a type 1 expression.

◀EXAMPLE▶ **Example 4:**
A signed, 5-digit decimal integer in a type 2 expression is converted to a type 1 expression.

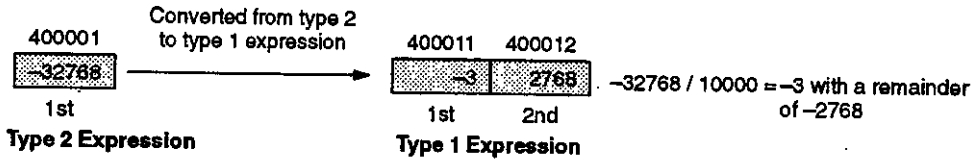
1) Ladder Programming



2) Conversion Process

a) Before Conversion

b) The following data conversion will be performed when input relay 100001 changes from OFF to ON. The conversion will be completed in one scan.



(1) The data in the leading register of the source table (V1) is taken as a signed 5-digit decimal integer (–32,768) in a type 2 expression, it is converted to a type 1 expression, and the result is stored in the two registers of the destination table.

(2) The data in the registers of the source table does not change.

(3) The status of the coils is as follows:

Coil 000101: ON only for the scan in which input relay 100001 changed from OFF to ON

Coil 000102: OFF

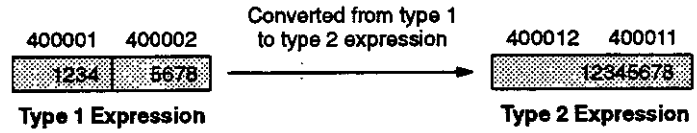
c) Thus, the above program section can be used to convert a signed 5-digit decimal integer in a type 2 expression (–32,768 to 32,767) to a type 1 expression.

7.2.6 32-BIT CONVERSION (DCST)

1. Function

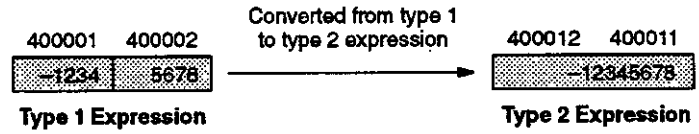
The numeric expression of a signed or unsigned 8-digit decimal integer is changed from type 1 to type 2 or from type 2 to type 1. The conversion is completed in one scan.

Example 1



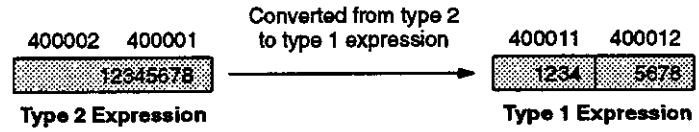
Type 1 Expression **Type 2 Expression**

Example 2



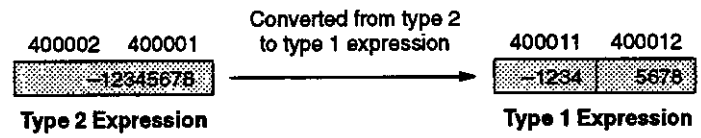
Type 1 Expression **Type 2 Expression**

Example 3



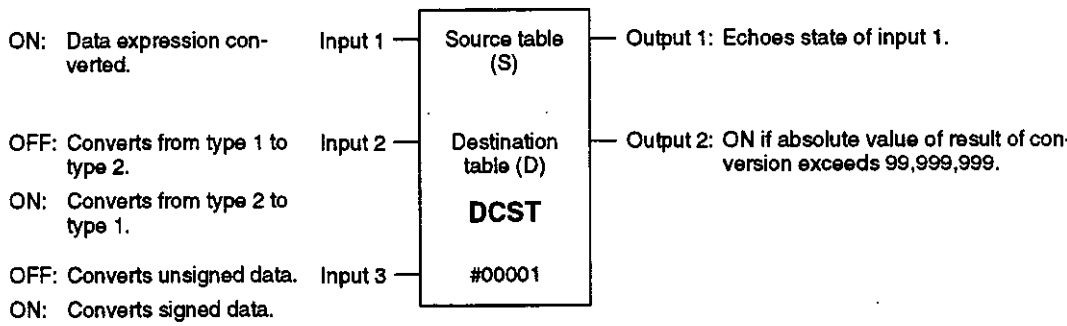
Type 2 Expression **Type 1 Expression**

Example 4



Type 2 Expression **Type 1 Expression**

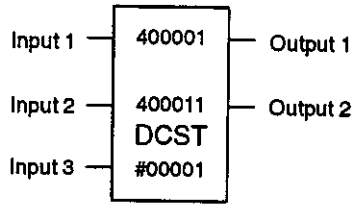
2. Structure



1) DCST is the symbol for 32-BIT CONVERSION.

2) DCST requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. Refer to *Table 7.11* for details on specifying constants or registers for these elements.

Example



400001: Leading reference number of source table
400011: Leading reference number of destination table
#00001: Fixed (1)

Table 7.11 Structural Elements of DCST

Element	Meaning	Possible Settings	
Top (S)	Leading reference number of source table (size:2)	Input register:	300001 to 300511 (Z00001 to Z00511)
		Holding register:	400001 to 409998 (W00001 to W09998)
		Constant register:	700001 to 704095 (K00001 to K04095)
		Link register:	R10001 to R11023 R20001 to R21023
Middle (D)	Leading reference number of destination table (size:2)	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 R20001 to R21023
Bottom	Fixed	Constant:	#00001

3. Operation

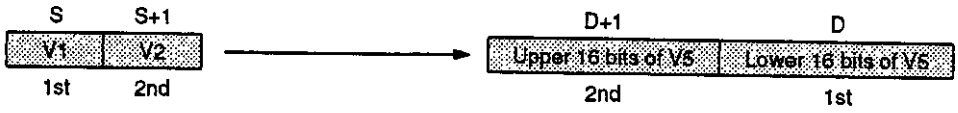
1) Status Before Execution

2) The following data conversion will be executed when input 1 turns ON and inputs 2 and 3 are OFF. The conversion will be completed in one scan.

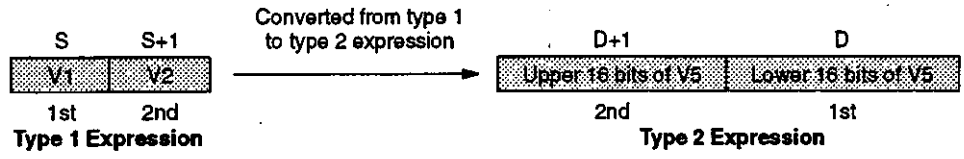
a) The data in each of the two registers of the source table (V1 and V2) is taken as an unsigned 5-digit decimal integer (0 to 65,535) in a type 1 expression and converted to an unsigned 9-digit decimal integer (0 to 655,415,535) (V5).

- V5 = 10,000 x V1 + V2

b) The result, V5, is stored in the two registers of the destination table. The upper 16 bits are stored in the second register of the destination table and the lower 16 bits of V5 are stored in the leading register of the destination table.



c) Thus, if the data in each of the two registers of the source table (V1 and V2) is between 0 and 9,999 in a type 1 expression, it will be converted to an unsigned 8-digit decimal integer (0 to 99,999,999) in a type 2 expression, and the result will be stored in the two registers of the destination table.



d) The data in the registers of the source table does not change.

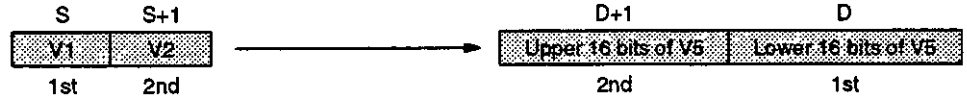
e) Output 1 is ON as long as input 1 is ON and output 2 remains OFF.

3) The following data conversion will be executed when inputs 1 and 3 turn ON and input 2 is OFF. The conversion will be completed in one scan.

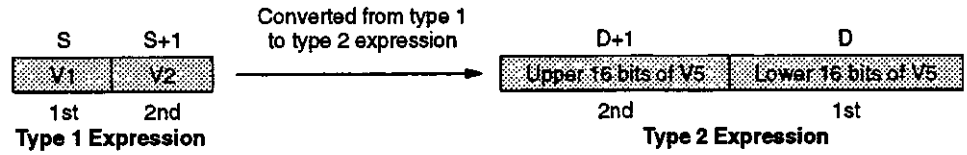
a) The data in each of the two registers of the source table (V1 and V2) is taken as a signed 5-digit decimal integer (–32,767 to 32,767) in a type 1 expression and converted to a signed 10-digit decimal integer (–2,147,483,648 to 2,147,483,647) (V5).

- If $V1 \geq 0$, then $V5 = 10{,}000 \times |V1| + |V2|$
- If $V1 < 0$, then $V5 = -(10{,}000 \times |V1| + |V2|)$

b) The result, V5, is stored in the two registers of the destination table. The upper 16 bits are stored in the second register of the destination table and the lower 16 bits of V5 are stored in the leading register of the destination table.



c) Thus, if the data in each of the two registers of the source table (V1 and V2) is $-9{,}999 \leqq 1 \leqq 9{,}999$ and $0 \leqq V2 \leqq 9{,}999$ in a type 1 expression, it will be converted to an signed 8-digit decimal integer (–99,999,999 to 99,999,999) in a type 2 expression, and the result will be stored in the two registers of the destination table.



d) The data in the registers of the source table does not change.

e) Output 1 is ON as long as input 1 is ON and output 2 remains OFF.

4) The following data conversion will be executed when inputs 1 and 2 turn ON and input 3 is OFF. The conversion will be completed in one scan.

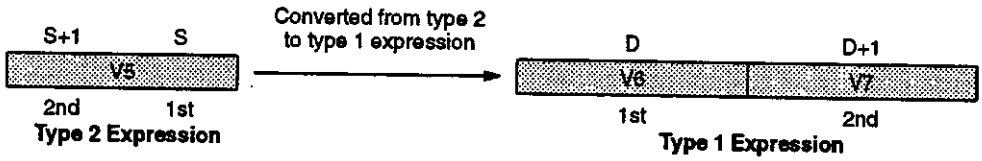
a) The data in the two registers of the source table (V5) is taken as an unsigned 9-digit decimal integer (0 to 4,294,967,295) in a type 2 expression, it is converted to a type 1 expression, and the result is stored in the two registers of the destination table (V6 and V7).

b) The following equation is used.

• V5 ÷ 10,000 = V6 with a remainder of V7

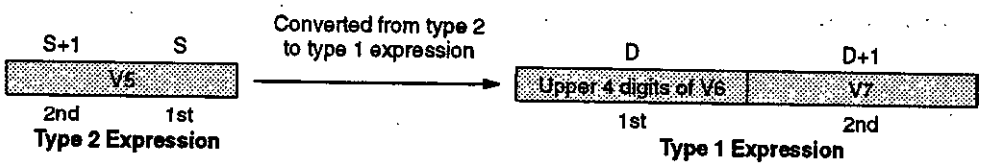
c) The result of the conversion is treated as follows:

(1) If V5 is 99,999,999 or less, V6 will be stored in the leading register and V7 will be stored in the second register of the destination table.



Output 1 is ON as long as input 1 is ON and output 2 remains OFF.

(2) If V5 is greater than 99,999,999, the upper 4 digits of V6 will be stored in the leading register and V7 will be stored in the second register. The 5th and 6th digits of V6 will be discarded.



Outputs 1 and 2 are ON as long as input 1 is ON.

d) The data in the registers of the source table does not change.

5) The following data conversion will be executed when inputs 1, 2, and 3 turn ON. The conversion will be completed in one scan.

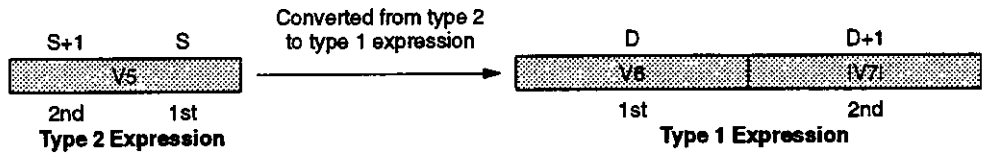
a) The data in the two registers of the source table (V5) is taken as a signed 9-digit decimal integer (–2,147,483,648 to 2,147,483,647) in a type 2 expression, it is converted to a type 1 expression, and the result is stored in the two registers of the destination table (V6 and V7).

b) The following equation is used.

- V5 ÷ 10,000 = V6 with a remainder of V7

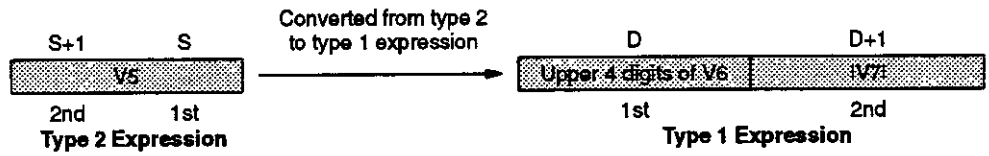
c) The result of the conversion is treated as follows:

(1) If the absolute value of V5 is 99,999,999 or less, V6 will be stored in the leading register and the absolute value of V7 will be stored in the second register.



Output 1 is ON as long as input 1 is ON and output 2 remains OFF.

(2) If the absolute value of V5 is greater than 99,999,999, the upper 4 digits of V6 will be stored in the leading register (signed) and the absolute value of V7 will be stored in the second register of the destination table. The 5th and 6th digits of V6 will be discarded.



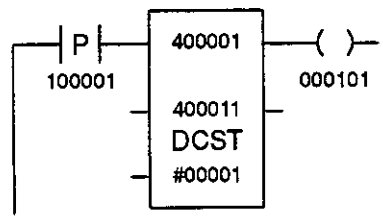
Outputs 1 and 2 is ON as long as input 1 remains ON.

d) The data in the registers of the source table does not change.

4. Application Examples

◀EXAMPLE▶ **Example 1:**
An unsigned, 8-digit decimal integer in a type 1 expression is converted to a type 2 expression.

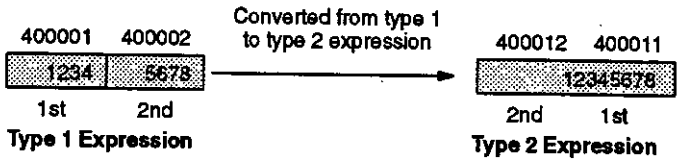
1) Ladder Programming



2) Conversion Process

a) Before Conversion

b) The following data conversion will be performed when input relay 100001 changes from OFF to ON. The conversion will be completed in one scan.



(1) The data in the two registers of the source table (V1 and V2) is taken as an unsigned 8-digit decimal integer (12,345,678) in a type 1 expression, it is converted to a type 2 expression, and the result is stored in the two registers of the destination table.

(2) The data in the registers of the source table does not change.

(3) The status of the coils is as follows:

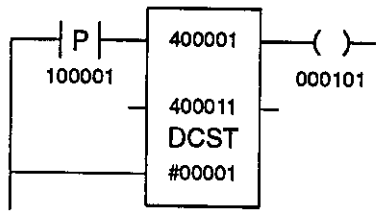
Coil 000101: ON only for the scan in which input relay 100001 changed from OFF to ON

Coil 000102: OFF

c) Thus, the above program section can be used to convert an unsigned 8-digit decimal integer in a type 1 expression (0 to 99,999,999) to a type 2 expression.

◀EXAMPLE▶ **Example 2:**
A signed, 8-digit decimal integer in a type 1 expression is converted to a type 2 expression.

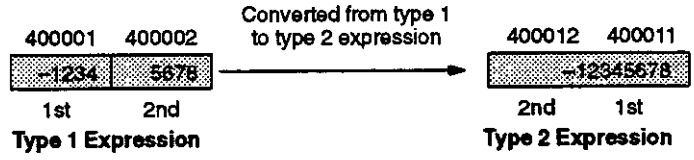
1) Ladder Programming



2) Conversion Process

a) Before Conversion

b) The following data conversion will be performed when input relay 100001 changes from OFF to ON. The conversion will be completed in one scan.



(1) The data in the two registers of the source table (V1 and V2) is taken as a signed 8-digit decimal integer (–12,345,678) in a type 1 expression, it is converted to a type 2 expression, and the result is stored in the two registers of the destination table.

(2) The data in the registers of the source table does not change.

(3) The status of the coils is as follows:

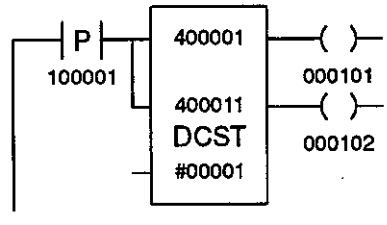
Coil 000101: ON only for the scan in which input relay 100001 changed from OFF to ON

Coil 000102: OFF

c) Thus, the above program section can be used to convert a signed 8-digit decimal integer in a type 1 expression (–99,999,999 to 99,999,999) to a type 2 expression.

◀EXAMPLE▶ **Example 3:**
An unsigned, 8-digit decimal integer in a type 2 expression is converted to a type 1 expression.

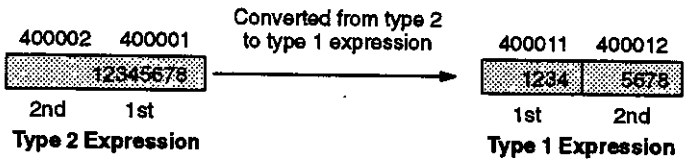
1) Ladder Programming



2) Conversion Process

a) Before Conversion

b) The following data conversion will be performed when input relay 100001 changes from OFF to ON. The conversion will be completed in one scan.



(1) The data in the two registers of the source table is taken as an unsigned 8-digit decimal integer (12,345,678) in a type 2 expression, it is converted to a type 1 expression, and the result is stored in the two registers of the destination table.

(2) The data in the registers of the source table does not change.

(3) The status of the coils is as follows:

Coil 000101: ON only for the scan in which input relay 100001 changed from OFF to ON

Coil 000102: OFF

c) Thus, the above program section can be used to convert an unsigned 8-digit decimal integer in a type 2 expression (0 to 99,999,999) to a type 1 expression.

d) If the value of the unsigned integer in the type 2 expression in the two registers of the source table is greater than 99,999,999, the following processing will be carried out when input relay 100001 turns ON.

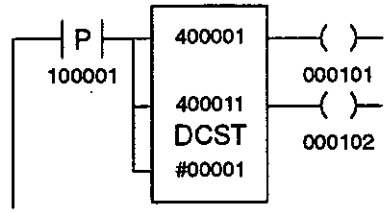
(1) The lower 8 digits of the result is stored in the two registers of the destination table. The 9th and 10th digits of V6 is discarded.

(2) The data in the registers of the source table does not change.

(3) Coils 000101 and 000102 turns ON only for the scan in which input relay 10001 changed from OFF to ON.

◀EXAMPLE▶ **Example 4:**
A signed, 8-digit decimal integer in a type 2 expression is converted to a type 1 expression.

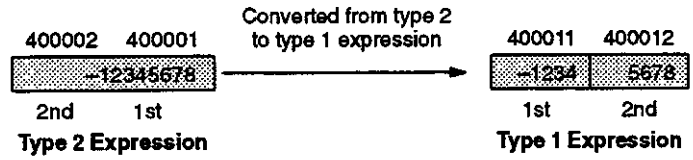
1) Ladder Programming



2) Conversion Process

a) Before Conversion

b) The following data conversion will be performed when input relay 100001 changes from OFF to ON. The conversion will be completed in one scan.



(1) The data in the two registers of the source table is taken as a signed 8-digit decimal integer (–99,999,999 to 99,999,999) in a type 2 expression, it is converted to a type 1 expression, and the result is stored in the two registers of the destination table.

(2) The data in the registers of the source table does not change.

(3) The status of the coils is as follows:

Coil 000101: ON only for the scan in which input relay 100001 changed from OFF to ON

Coil 000102: OFF

c) Thus, the above program section can be used to convert a signed 8-digit decimal integer in a type 2 expression (–99,999,999 to 99,999,999) to a type 1 expression.

d) If the absolute value of the signed integer in the type 2 expression in the two registers of the source table is greater than 99,999,999, the following processing will be carried out when input relay 100001 turns ON.

(1) The lower 8 digits of the result is stored in the two registers of the destination table. The 9th and 10th digits of V6 is discarded.

(2) The data in the registers of the source table does not change.

(3) Coils 000101 and 000102 turns ON only for the scan in which input relay 10001 changed from OFF to ON.

7.3 Building Programs

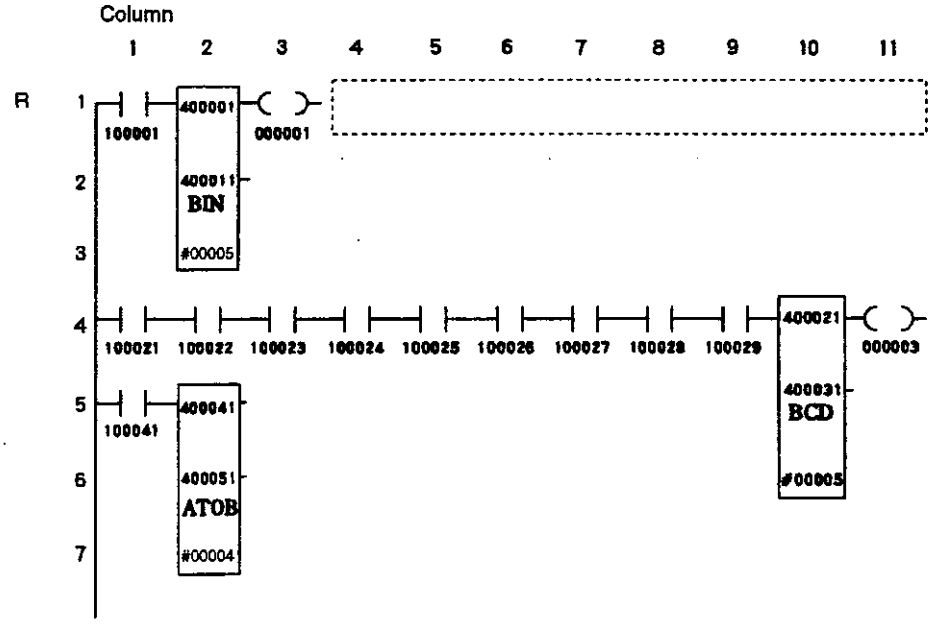
This section describes precautions that should be taken when designing circuits that contain bit manipulation instructions.

7.3.1 Storage Locations on Networks

Data conversion instructions require three vertical elements on a network, one top element, one middle element, and one bottom element. They can thus be stored anywhere on a 5-row by 10-column matrix (rows 1 through 5 and columns 1 through 10) on the network.

Note Data conversion instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

Example



7.3.2 Inputs

Inputs to data conversion instructions can be connected to relay elements (except coils) and/or outputs from timers, counters, math instructions, data manipulation instructions, other instructions, etc.

7.3.3 Outputs

Outputs from data conversion instructions can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data manipulation instructions, etc.

Other Data Manipulation Instructions

8

This chapter describes instructions used to manipulate data in various ways.

8.1 Other Data Manipulation Instructions

This section describes other data manipulation instructions and their functions (besides those outlined in Chapters 3 to 7).

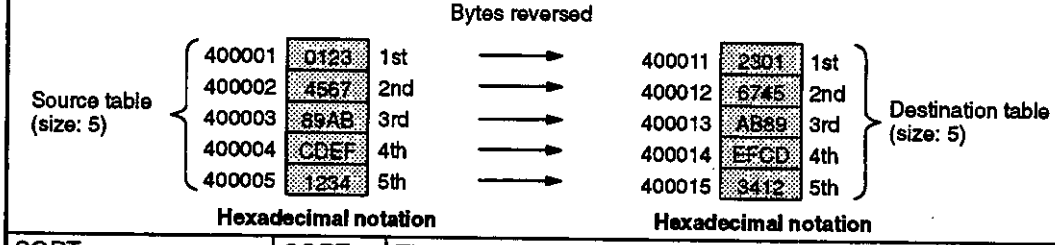
• The eleven data manipulation instructions are described in the following table.

Table 8.1 Other Data Manipulation Instructions

Name	Symbol	Function	Page
SET WORD DATA	SDAT	Word data (16-bit data) is set in a register. Execution of the instruction is completed in one scan. This instruction can be thought of as one type of data transfer instruction.	8-6
Example 1 Setting a constant Constant 65535 Source —Set→ 400011 65535 Destination **Example 2 Setting a holding register data** 400001 −32768 Source —Set→ 400011 −32768 Destination			
SET DOUBLE WORD DATA	SDDT	A 32-bit binary integer is set in two consecutive registers using a type 2 numeric expression. Execution of the instruction is completed in one scan. This instruction can be thought of as one type of data transfer instruction.	8-9
Example Setting a positive integer (1, 234, 567, 890) in holding registers 400012 and 400011 as type 2 data. Constant 18838 Source 1 Constant 722 Source 2 —Set→ 400012 400011: 1234567890 *18,838 × 65,536 + 722 = 1,234,567,890			
LOGICAL BYTE REARRANGEMENT	TWST	Each register in the destination table is separated into upper and lower bytes and then the order of the bits within each byte is placed in reverse order. The rearrangement is completed in one scan.	8-14
Example Destination table (size: 5) — Rearranged — Destination table (size: 5) 400001 0001 0010 0011 0100 1st → 400001 0100 1000 0010 1100 1st 400002 0101 0110 0111 1000 2nd → 400002 0110 1010 0001 1110 2nd 400003 1001 1010 1011 1100 3rd → 400003 0101 1001 0011 1101 3rd 400004 1101 1110 1111 0000 4th → 400004 0111 1011 0000 1111 4th 400005 0001 0010 0011 0100 5th → 400005 0100 1000 0010 1100 5th (Upper byte) (Lower byte) (Upper byte) (Lower byte)			

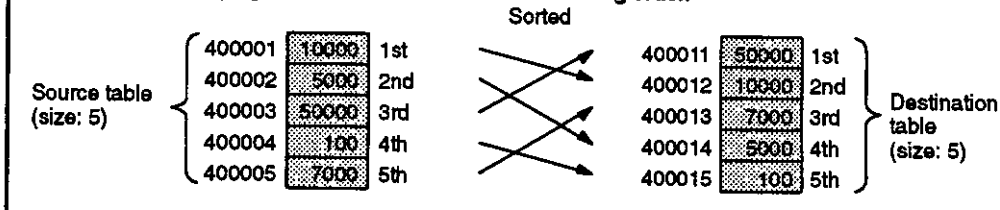
Name	Symbol	Function	Page
SWAP	SWAP	Each register in the source table is separated into upper and lower bytes and then the order of the bytes is reversed and stored in the corresponding register of the destination table. The rearrangement is completed in one scan.	8-18

Example



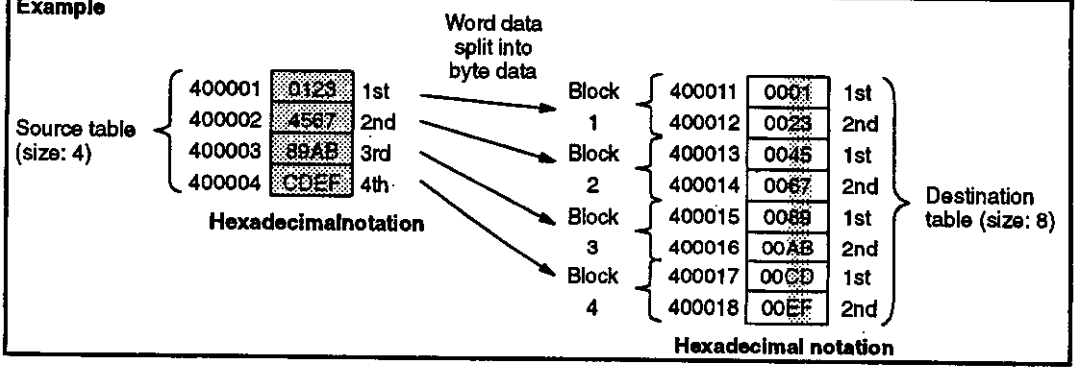
SORT	SORT	The data in each register of the source table or of the destination table is treated as 16-bit binary data (0 to 65,535), the data is sorted into ascending or descending order, and then the sorted result is stored. The sort is completed in one scan.	8-23

Example **Rearranging the source table data in descending order.**



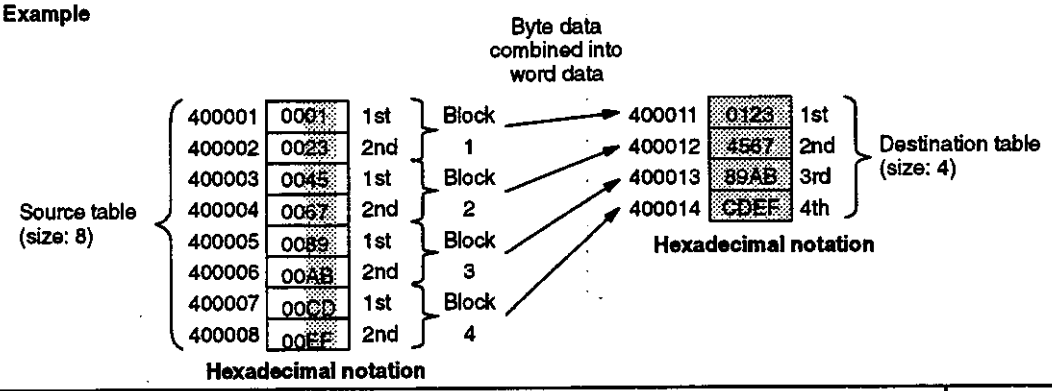
BYTE SPLIT	BYSL	The data (word data) in each register of the source table is split into bytes and then the bytes are stored in the two registers of the corresponding block in the destination table. The split is completed in one scan.	8-33

Example



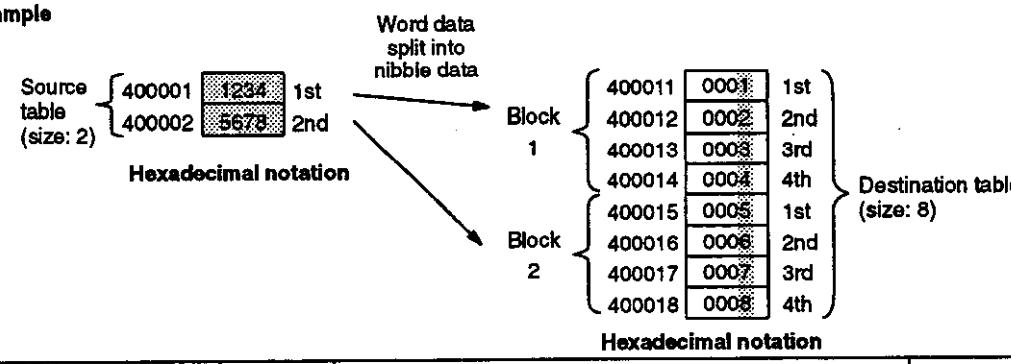
Name	Symbol	Function	Page
BYTE COMPOSITION	BYCM	The data in the lower bytes of the two registers in each block in the source table is combined into word data and the data is stored in the corresponding register in the destination table. Execution is completed in one scan.	8-37

Example



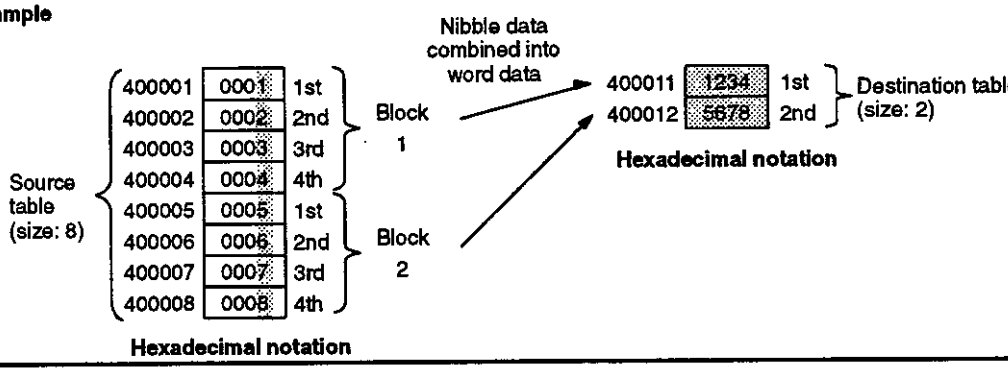
NIBBLE SPLIT	NBSL	The data word data for each register or 16 coils/relays of the source table is split into four nibbles (4-bit data) and then the nibbles are stored in the four registers of the corresponding block in the destination table. The split is completed in one scan.	8-40

Example



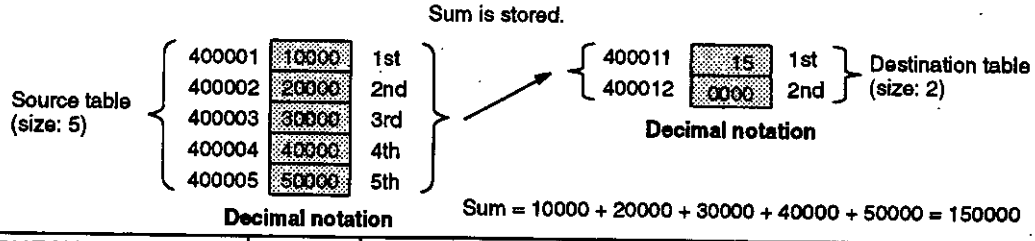
NIBBLE COMPOSITION	NBCM	The data in the lower four bits (nibbles) of the four registers in each block in the source table is combined into word data and the data is stored in the corresponding register or 16 coils/relays in the destination table. Execution is completed in one scan.	8-46

Example



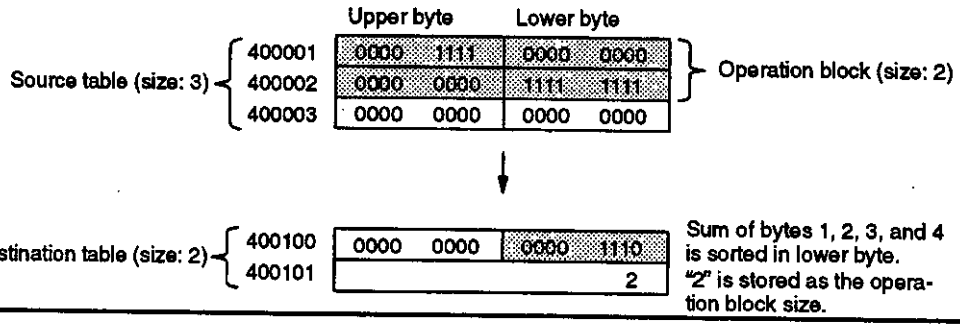
Name	Symbol	Function	Page
BLOCK ADD	BADD	The data in registers of the source table is added by word or by byte in unsigned addition and the result is stored in the two registers of the destination table. Execution is completed in one scan.	8-52

Example Word adding



CHECKSUM	CKSM	Straight check calculation, binary calculation, CRC-16 (cyclic redundancy check) calculation, or LRC (longitudinal redundancy check) calculation is performed for the data in the source table.	8-56

Example For the straight check calculation, the operation block data is added by byte and the result is stored in the lower byte of the leading register of the destination table. "00" (hexadecimal) is stored in the upper byte.



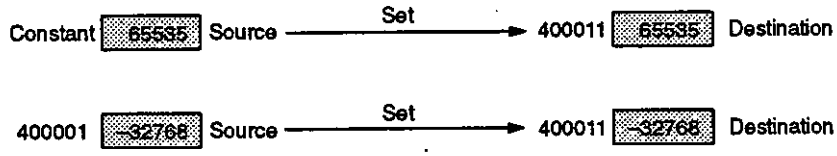
8.2 Data Setting Instructions

This section describes the functions, structures, and operations of the data setting instructions and provides simple examples of their application.

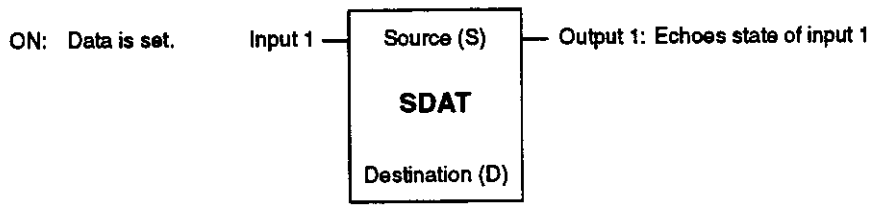
8.2.1 SET WORD DATA (SDAT)

1. Function

Word data (16-bit data) is set in a register. Execution of the instruction is completed in one scan. This instruction can be thought of as one type of data transfer instruction.



2. Structure



1) SDAT is the symbol for SET WORD DATA.

2) SDAT requires two elements, one top element and one bottom element, located vertically on the network. Refer to *Table 8.2* for details on specifying constants or registers for these elements.

Example

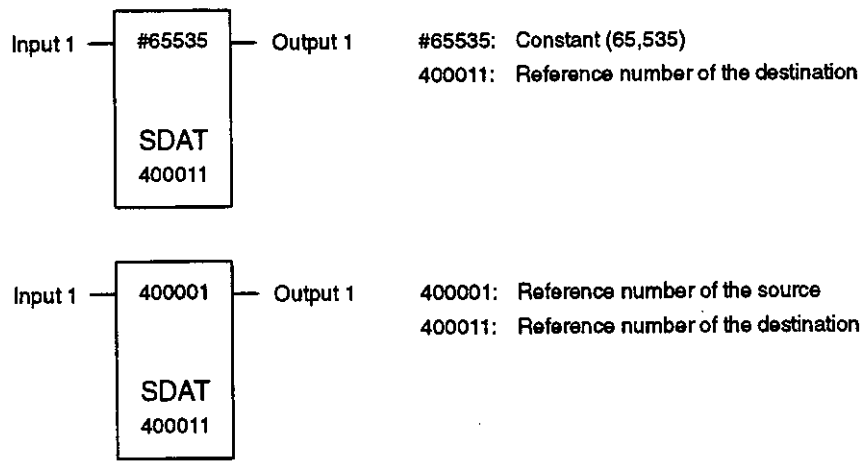


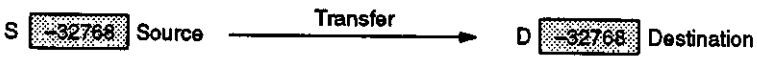
Table 8.2 Structural Elements of SDAT

Element	Meaning	Possible Settings	
Top (S)	Reference number of source	Constant:	#00000 to #65535
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 or R20001 to R21024
Bottom (D)	Reference number of destination	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 or R20001 to R21024

3. Operation

1) Status Before Execution

2) The following data transfer will be executed when input 1 turns ON.



a) The word data in the source is transferred to the destination.

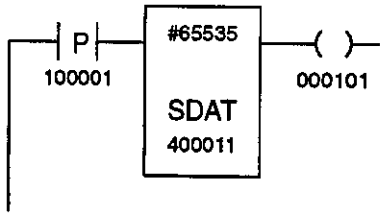
b) The data in the source does not change.

c) Output 1 is ON as long as input 1 is ON.

4. Application Examples

◀EXAMPLE▶ **Example 1**
Data is transferred from a constant.

1) Ladder Programming



2) Operation

a) Before Execution

Constant [65535] Source 400011 [0] Destination

b) The following data transfer will be performed when input relay 100001 changes from OFF to ON. The transfer will be completed in one scan.

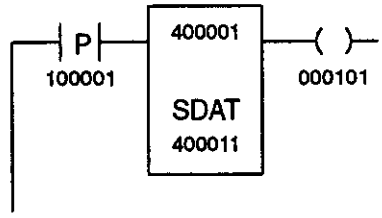
Constant [65535] Source —Transfer→ 400011 [65535] Destination

(1) The constant specified as the source (65,535) is transferred to the destination.

(2) Coil 000101 turns ON only for the scan in which input relay 100001 changed from OFF to ON.

◀EXAMPLE▶ **Example 2**
Data is transferred from a holding register.

1) Ladder Programming



2) Operation

a) Before Execution

400001 [−32768] Source 400011 [0] Destination

b) The following data transfer will be performed when input relay 100001 changes from OFF to ON. The transfer will be completed in one scan.

400001 [−32768] Source —Transfer→ 400011 [−32768] Destination

(1) The source data is transferred to the destination.

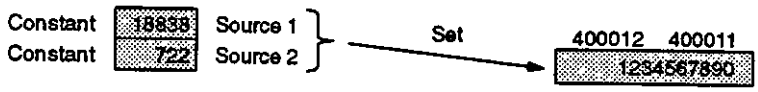
(2) The source data does not change.

(3) Coil 000101 turns ON only for the scan in which input relay 100001 changed from OFF to ON.

8.2.2 SET DOUBLE WORD DATA (SDDT)

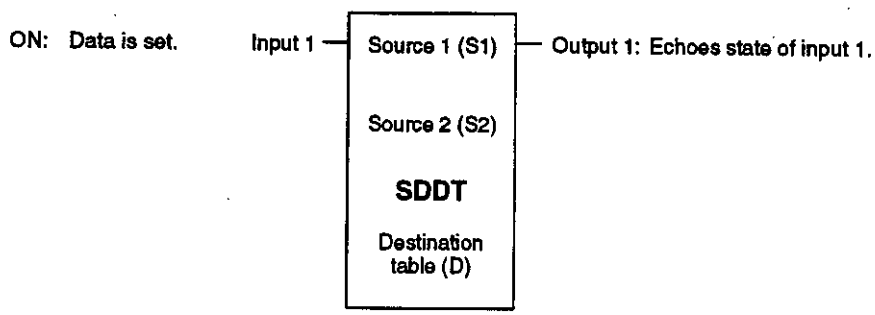
1. Function

A 32-bit binary integer is set in two consecutive registers using a type 2 numeric expression. Execution of the instruction is completed in one scan. This instruction can be thought of as one type of data transfer instruction.



*18,838 × 65,536 + 722 = 1,234,567,890

2. Structure



1) SDDT is the symbol for SET DOUBLE WORD DATA.

2) SDDT requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. Refer to *Table 8.3* for details on specifying constants or registers for these elements.

Example

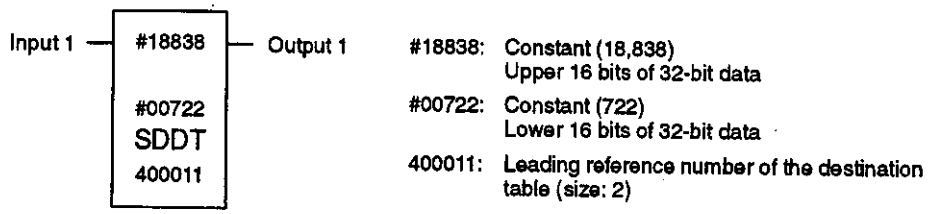


Table 8.3 Structural Elements of SDDT

Element	Meaning	Possible Settings
Top (S1)	Specify the upper 16 bits of the 32-bit data between 0 and 65,535.	Constant: #00000 to #65535
Middle (S2)	Specify the lower 16 bits of the 32-bit data between 0 and 65,535.	
Bottom (D)	Leading reference number of destination table. The 32-bit data is set as follows (table size fixed at 2): D+1 | Upper 16 bits | 1st D | Lower 16 bits | 2nd	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 or R20001 to R21023

3) The values for S1 and S2 to store as source 1 and source 2 to set for a 32-bit integer V in registers D and D+1 can be calculated as follows:

a) If V ≥ 0,

V ÷ 65,536 = S1 with a remainder of S2

Example: If V = 1,234,567,890,
V ÷ 65,536 = 18,838 with a remainder of 722.
Thus S1 = 18,838 and S2 = 722

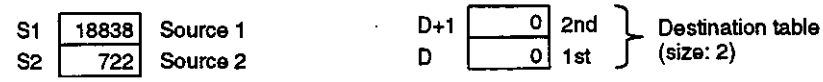
b) If V < 0,

|4294967296 + V| ÷ 65,536 = S1 with a remainder of S2

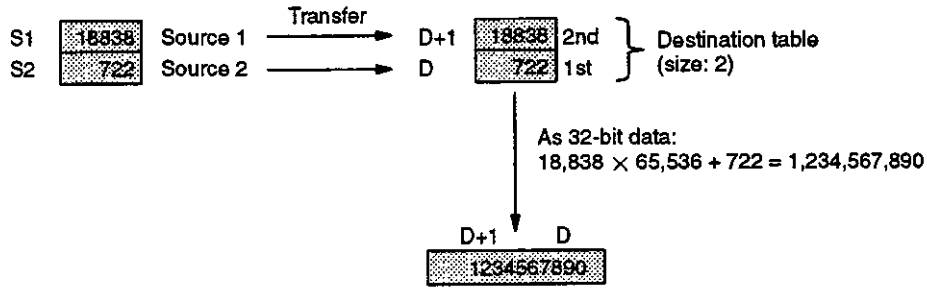
Example: If V = −1,234,567,890,
|4294967296 + V| ÷ 65,536 = 46,697 with a remainder of 64,814.
Thus S1 = 46,697 and S2 = 64,814

3. Operation

1) Status Before Execution



2) The following data transfer will be executed when input 1 turns ON.



a) The data in source 1 and 2 is transferred to the first and second registers in the destination table.

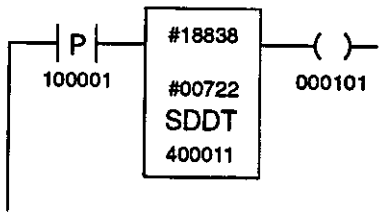
b) Thus, 32-bit data is stored in the two consecutive registers of the destination table by storing the upper 16 bits of the 32-bit data in source 1 and the lower 16 bits in source 2.

c) Output 1 is ON as long as input 1 is ON.

4. Application Examples

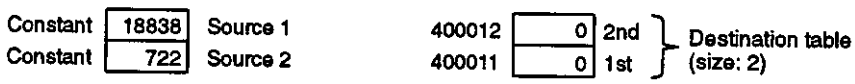
◀EXAMPLE▶ **Example 1**
Unsigned 32-bit data is set.

1) Ladder Programming

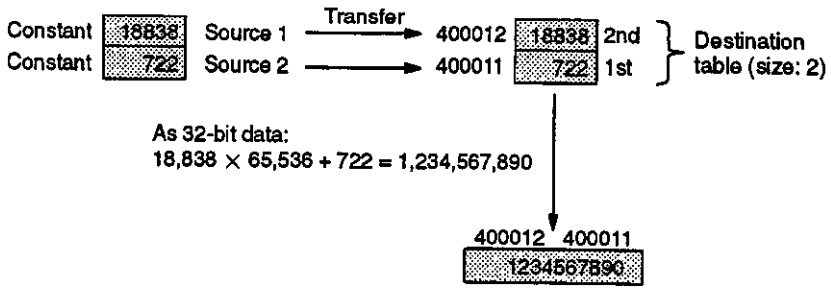


2) Operation

a) Before Execution



b) The following data transfer will be performed when input relay 100001 changes from OFF to ON. The transfer will be completed in one scan.



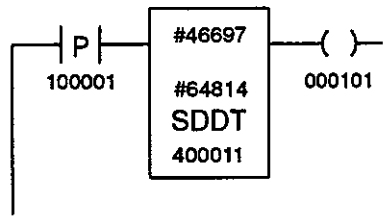
(1) The data in source 1 and source 2 is transferred to the first and second registers in the destination table.

(2) The 32-bit data (1,234,567,890) is thus set in holding registers 400012 and 400011.

(3) Coil 000101 turns ON only for the scan in which input relay 100001 changed from OFF to ON.

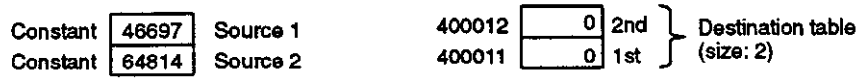
◀EXAMPLE▶ **Example 2**
Signed 32-bit data is set.

1) Ladder Programming

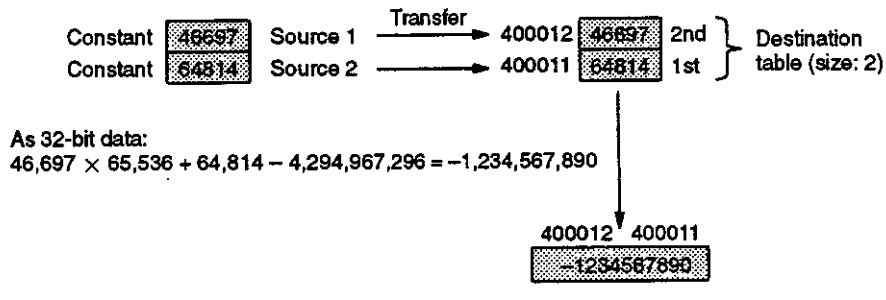


2) Operation

a) Before Execution



b) The following data transfer will be performed when input relay 100001 changes from OFF to ON. The transfer will be completed in one scan.



(1) The data in source 1 and source 2 is transferred to the first and second registers in the destination table.

(2) The 32-bit data (–1,234,567,890) is thus set in holding registers 400012 and 400011.

(3) Coil 000101 turns ON only for the scan in which input relay 100001 changed from OFF to ON.

8.2.3 Building Programs

1. Storage Locations on Networks

A. SET WORD DATA

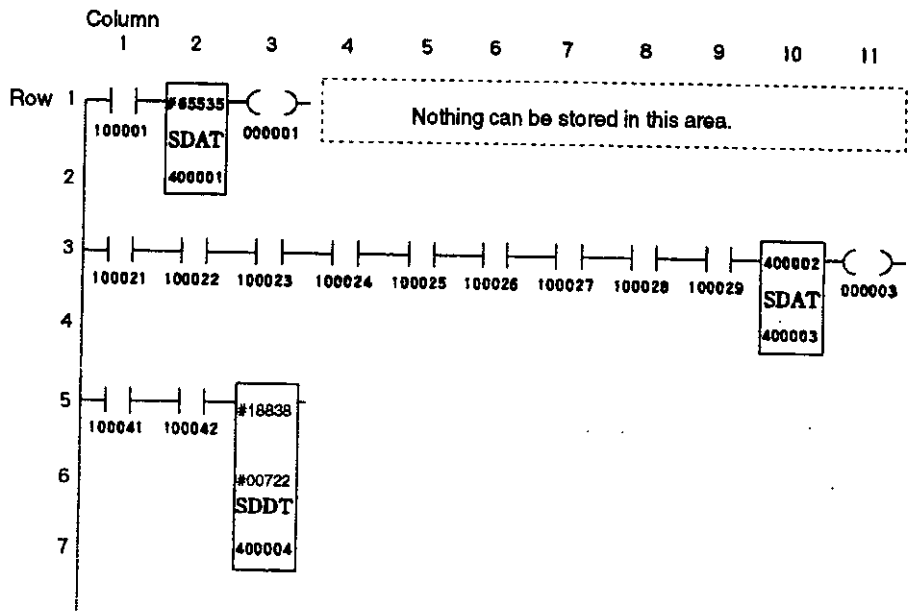
SET WORD DATA instructions require two elements (top and bottom) located vertically on the network, so they can be stored anywhere on a 6-row by 10-column matrix (rows 1 through 6 and columns 1 through 10).

B. SET DOUBLE WORD DATA

SET DOUBLE WORD DATA instructions require three vertical elements on a network, one top element, one middle element, and one bottom element. They can thus be stored anywhere on a 5-row by 10-column matrix (rows 1 through 5 and columns 1 through 10) on the network.

Note SET WORD DATA and SET DOUBLE WORD DATA instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

Example



2. Inputs

Inputs to SET WORD DATA and SET DOUBLE WORD DATA instructions can be connected to relay elements (except coils) and/or outputs from timers, counters, math instructions, data manipulation instructions, other instructions, etc.

3. Outputs

Outputs from SET WORD DATA and SET DOUBLE WORD DATA instructions can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data manipulation instructions, etc.

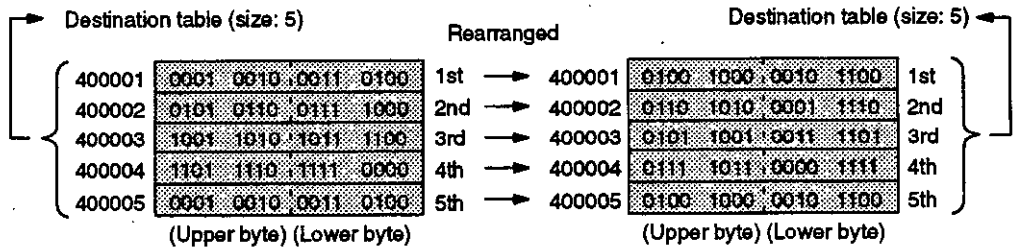
8.3 Data Rearrangement Instructions

This section describes the function, structures, and operation of the data rearrangement instructions and provides simple examples of their application.

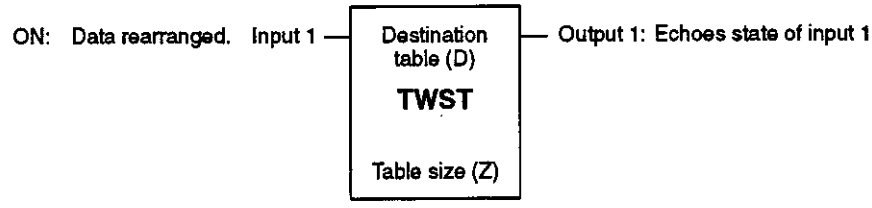
8.3.1 LOGICAL BYTE REARRANGEMENT (TWST)

1. Function

Each register in the destination table is separated into upper and lower bytes and then the order of the bits within each byte is placed in reverse order. The rearrangement is completed in one scan.



2. Structure



1) TWST is the symbol for LOGICAL BYTE REARRANGEMENT.

2) TWST requires two elements, one top element and one bottom element, located vertically on the network. Refer to *Table 8.4* for details on specifying constants or registers for these elements.

Example

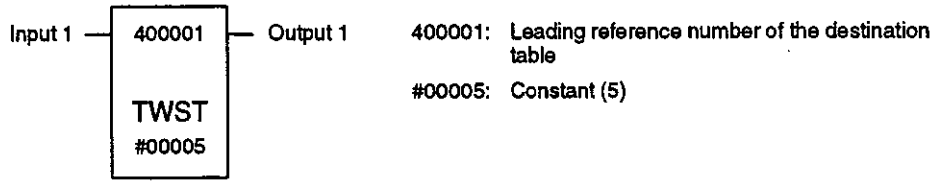
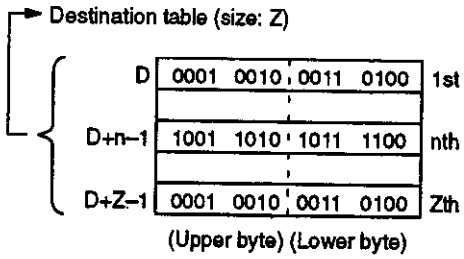


Table 8.4 Structural Elements of TWST

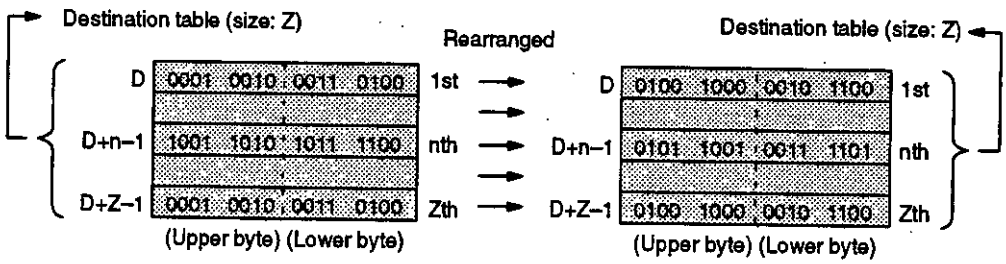
Element	Meaning	Possible Settings	
Top (D)	Leading reference number in destination table	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 or R20001 to R21024
Bottom (Z)	Size of destination table	Constant:	#00001 to #00100

3. Operation

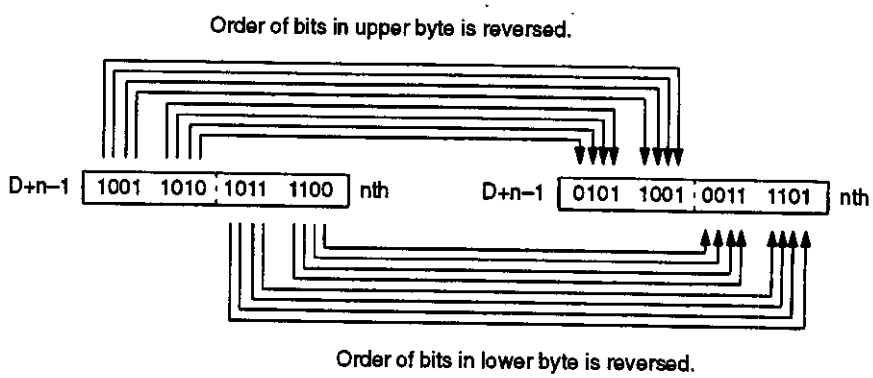
1) Status Before Execution



2) The following data rearrangement will be executed when input 1 turns ON. The rearrangement will be completed in one scan.



a) Each register n (n = 1 to Z) in the destination table is separated into upper and lower bytes and then the order of the bits within each byte is placed in reverse order as shown in the following illustration.

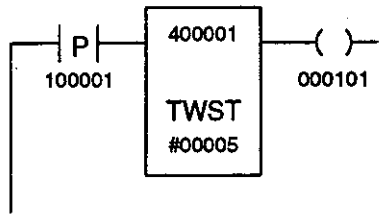


b) Output 1 is ON as long as input 1 is ON.

4. Application Examples

◀EXAMPLE▶ **Example 1**

1) Ladder Programming



2) Operation

a) Before Execution

Destination table (size: 5)

400001	0001	0010	0011	0100	1st
400002	0101	0110	0111	1000	2nd
400003	1001	1010	1011	1100	3rd
400004	1101	1110	1111	0000	4th
400005	0001	0010	0011	0100	5th

(Upper byte) (Lower byte)

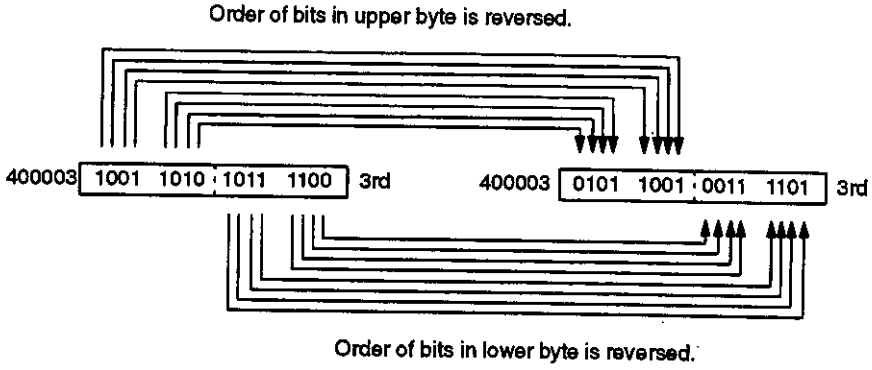
b) The following data rearrangement will be performed when input relay 100001 changes from OFF to ON. The rearrangement will be completed in one scan.

Destination table (size: 5) — Rearranged — Destination table (size: 5)

400001	0001	0010	0011	0100	1st	→	400001	0100	1000	0010	1100	1st
400002	0101	0110	0111	1000	2nd	→	400002	0110	1010	0001	1110	2nd
400003	1001	1010	1011	1100	3rd	→	400003	0101	1001	0011	1101	3rd
400004	1101	1110	1111	0000	4th	→	400004	0111	1011	0000	1111	4th
400005	0001	0010	0011	0100	5th	→	400005	0100	1000	0010	1100	5th

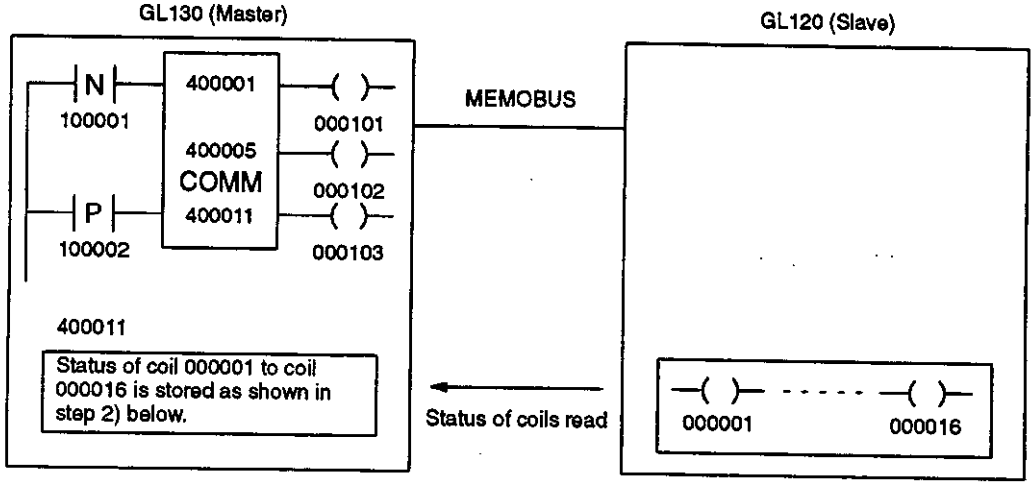
(Upper byte) (Lower byte) (Upper byte) (Lower byte)

(1) Each register n (n = 1 to 5) in the destination table is separated into upper and lower bytes and then the order of the bits within each byte is placed in reverse order. The following illustration shows how the bits is rearranged for the third register.



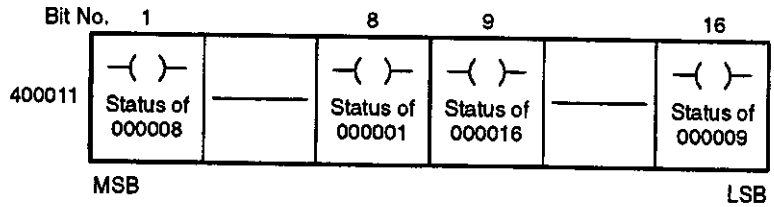
(2) Coil 000101 turns ON only for the scan in which input relay 100001 changed from ON to OFF.

◀EXAMPLE▶ **Example 2**

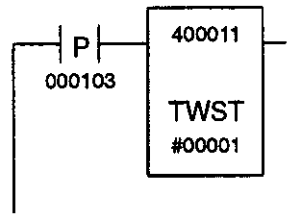


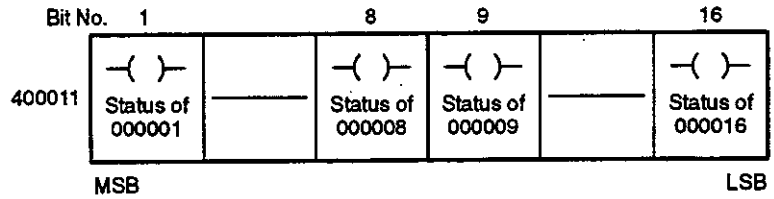
1) A GL130 PLC and a GL120 PLC are connected via MEMOBUS as in the above illustration.

2) The program in the GL130 uses the COMM instruction to read the status of coil 000001 to coil 000016 of the GL120 and store the results in holding register 400011, as shown in the following illustration. The COMM instruction is used in the automatic MEMOBUS mode.



3) TWST can be used as shown below to rearrange the contents of holding register 400011 to the order required by data transfer, matrix, and other instructions.



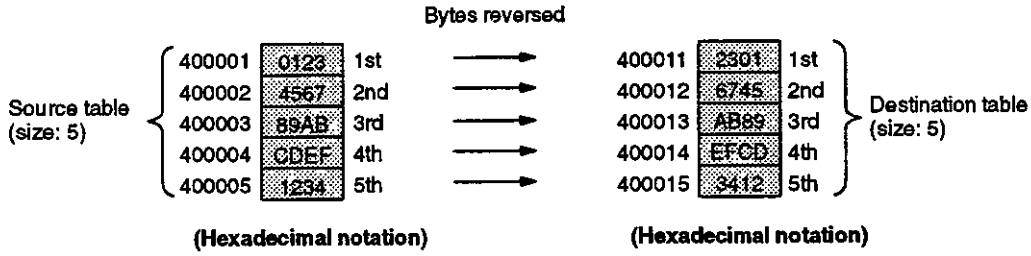


8.3.2 SWAP (SWAP)

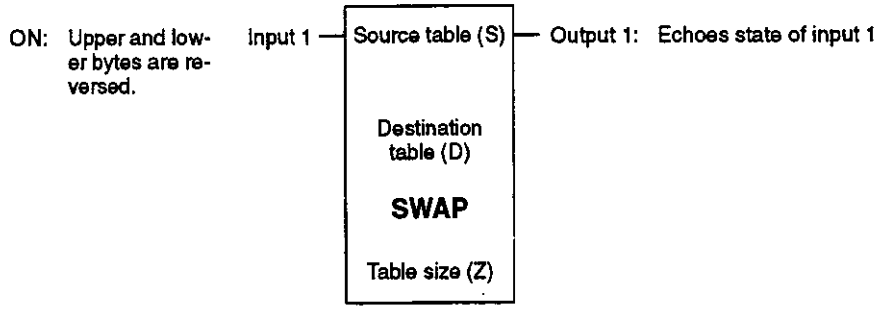
1. Function

Each register in the source table is separated into upper and lower bytes and then the order of the bytes is reversed and stored in the corresponding register of the destination table. The rearrangement is completed in one scan.

Example



2. Structure



1) SWAP is the symbol for SWAP.

2) SWAP requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. Refer to *Table 8.5* for details on specifying constants or registers for these elements.

Example

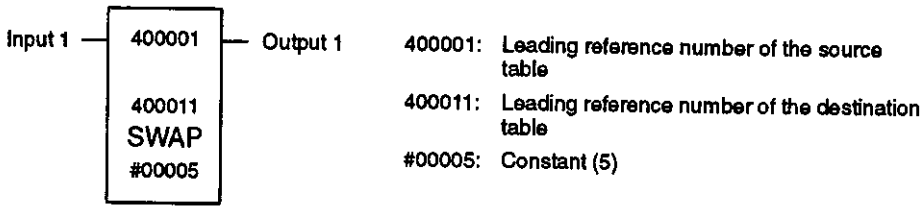
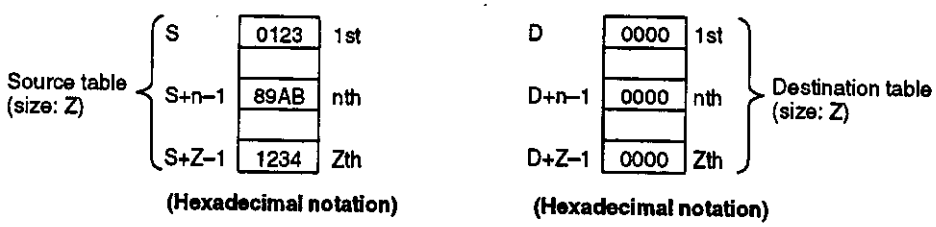


Table 8.5 Structural Elements of SWAP

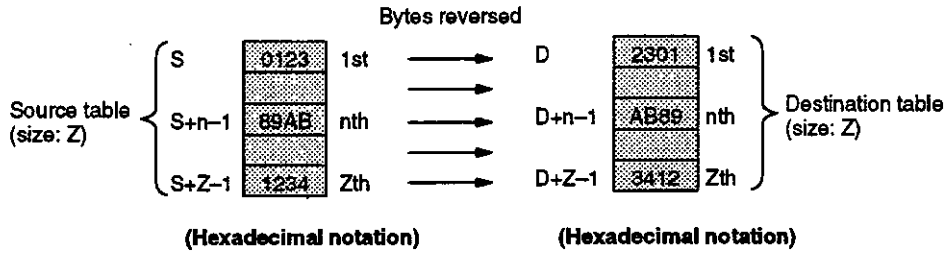
Element	Meaning	Possible Settings	
Top (S)	Leading reference number in source table	Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 or R20001 to R21024
Middle (D)	Leading reference number in destination table	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 or R20001 to R21024
Bottom (Z)	Size of source and destination tables	Constant:	#00001 to #00100

3. Operation

1) Status Before Execution



2) The following data rearrangement will be executed when input 1 turns ON. The rearrangement will be completed in one scan.



a) Each register n (n = 1 to Z) in the source table is separated into upper and lower bytes and then the order of the bytes is reversed and stored in nth register of the destination table.

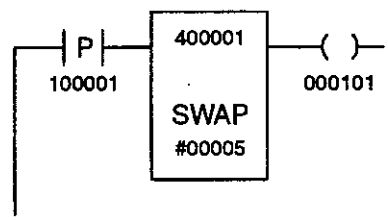
b) The data in the source table does not changed.

c) Output 1 is ON as long as input 1 is ON.

4. Application Examples

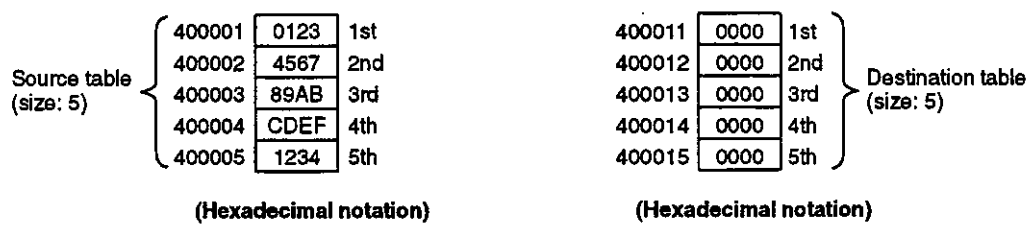
◀EXAMPLE▶ **Example 1**

1) Ladder Programming

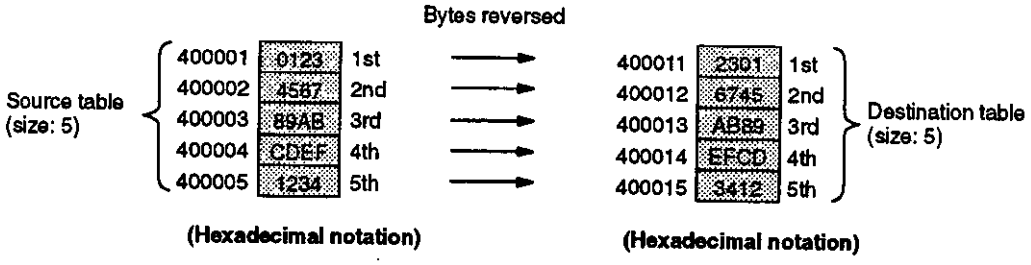


2) Operation

a) Before Execution



b) The following data rearrangement will be performed when input relay 100001 changes from OFF to ON. The rearrangement will be completed in one scan.



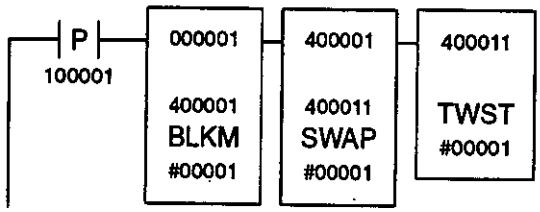
(1) Each register n (n = 1 to 5) in the source table is separated into upper and lower bytes and then the order of the bytes is reversed and stored in nth register of the destination table.

(2) The data in the source table does not changed.

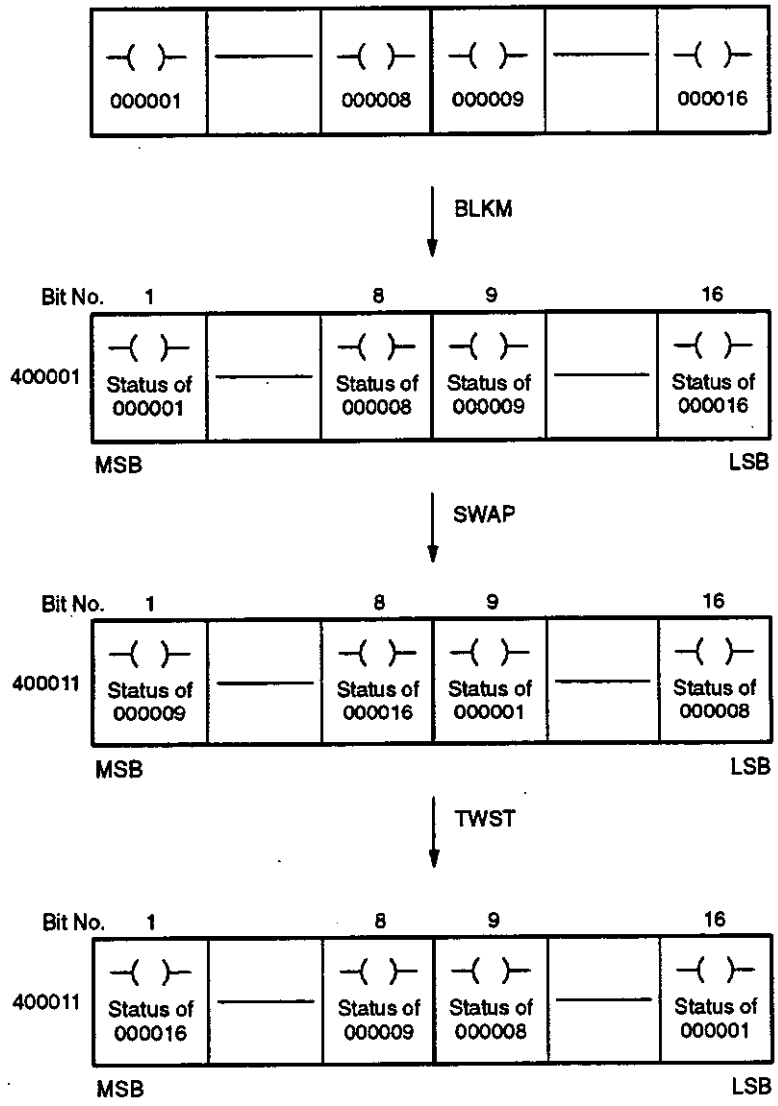
(3) Coil 000101 turns ON only for the scan in which input relay 100001 changed from ON to OFF.

◀EXAMPLE▶ **Example 2**

1) Ladder Programming



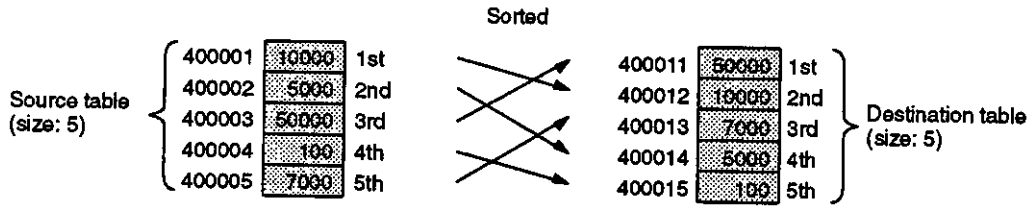
2) When input relay 100001 turns ON, the status of coil 000001 to coil 000016 will be stored in holding register 400011 in the order 000016 to 000001.



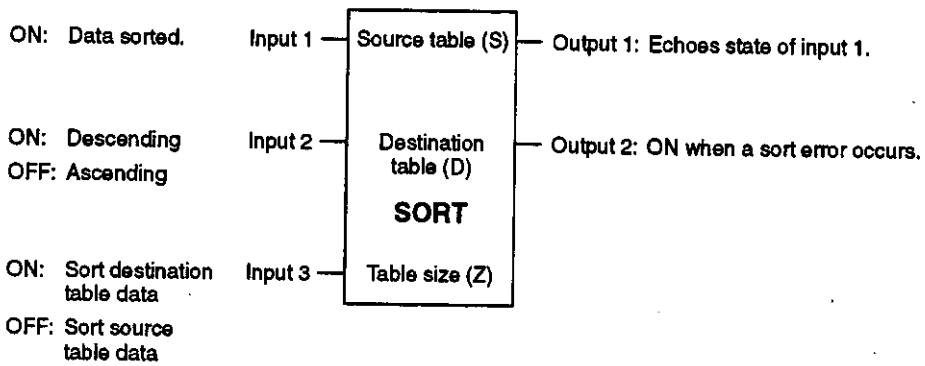
8.3.3 SORT (SORT)

1. Function

The data in each register of the source table or of the destination table is treated as 16-bit binary data (0 to 65,535), the data is sorted into ascending or descending order, and then the sorted result is stored. The sort is completed in one scan.



2. Structure



1) SORT is the symbol for SORT.

2) SORT requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. Refer to *Table 8.6* for details on specifying constants or registers for these elements.

Example

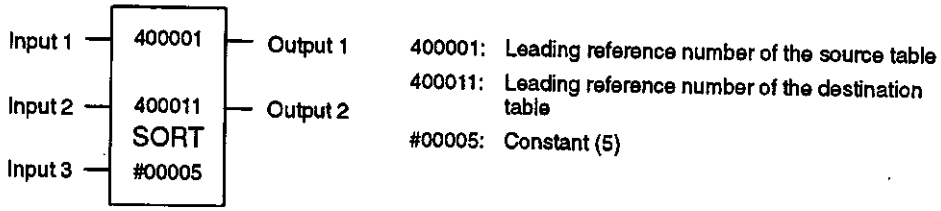
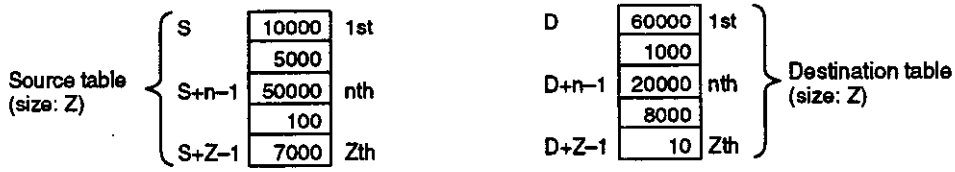


Table 8.6 Structural Elements of SORT

Element	Meaning	Possible Settings	
Top (S)	Leading reference number in source table	Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 or R20001 to R21024
Middle (D)	Leading reference number in destination table	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 or R20001 to R21024
Bottom (Z)	Size of source and destination tables	Constant:	#00001 to #00100

3. Operation

1) Status Before Execution



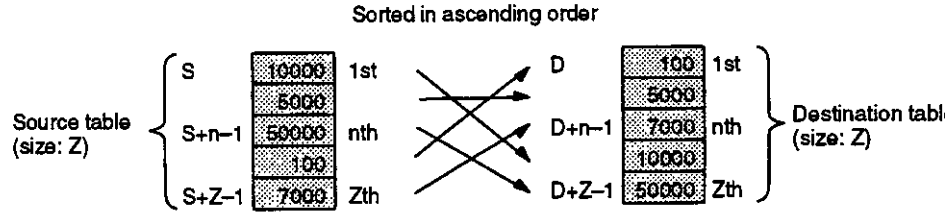
IMPORTANT The following description for items 2 to 5 assume that the source and destination tables do not overlap.

2) Sorting Source Table Data in Ascending Order

a) When only input 1 turns ON, the data in each register of the source table will be treated as 16-bit binary data (0 to 65,535), the data will be sorted into ascending order, and then the sorted result will be stored in the destination table. The sort will be completed in one scan.

b) The data in the source table does not change.

c) Output 1 is ON as long as input 1 is ON and output 2 remains OFF.

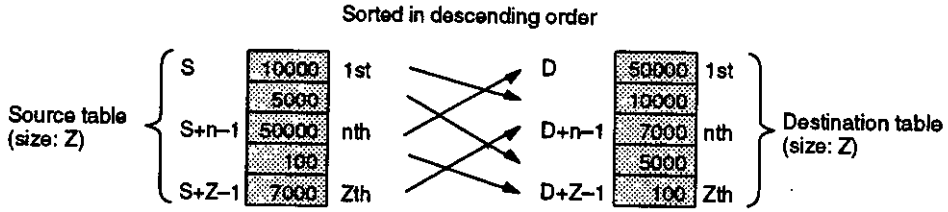


3) Sorting Source Table Data in Descending Order

a) When inputs 1 and 2 turn ON and input 3 turns OFF, the data in each register of the source table will be treated as 16-bit binary data (0 to 65,535), the data will be sorted into descending order, and then the sorted result will be stored in the destination table. The sort will be completed in one scan.

b) The data in the source table does not change.

c) Output 1 is ON as long as input 1 is ON and output 2 remains OFF.

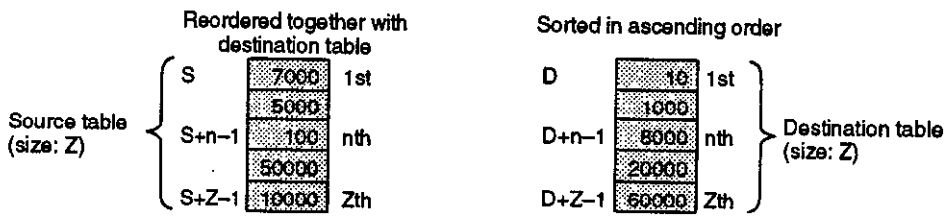


4) Sorting Destination Table Data in Ascending Order

a) When inputs 1 and 3 turn ON and input 2 turns OFF, the data in each register of the destination table will be treated as 16-bit binary data (0 to 65,535), the data will be sorted into ascending order, and then the sorted result will be stored in the destination table. The sort will be completed in one scan.

b) The data in the registers of the source table is also reordered in the same way as the registers in the destination table, i.e., data in corresponding registers moves together with the registers of the destination table.

c) Output 1 is ON as long as input 1 is ON and output 2 remains OFF.

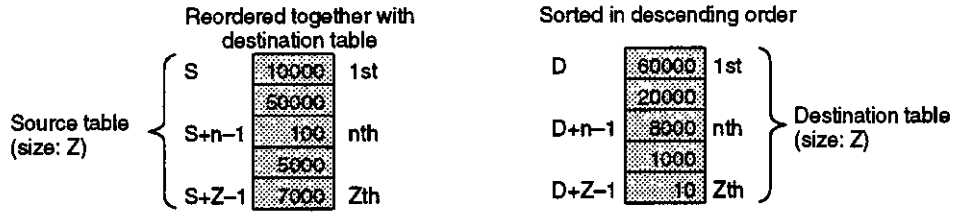


5) Sorting Destination Table Data in Descending Order

a) When inputs 1, 2, and 3 turn ON, the data in each register of the destination table will be treated as 16-bit binary data (0 to 65,535), the data will be sorted into descending order, and then the sorted result will be stored in the destination table. The sort will be completed in one scan.

b) The data in the registers of the source table is also reordered in the same way as the registers in the destination table, i.e., data in corresponding registers moves together with the registers of the destination table.

c) Output 1 is ON as long as input 1 is ON and output 2 remains OFF.



6) Sorting Overlapping Source and Destination Tables

a) If input 3 is OFF and input 1 turns ON, the data in each register of the source table will be treated as 16-bit binary data (0 to 65,535), the data will be sorted into descending or ascending order, and then the sorted result will be stored in the destination table. The data in the registers in the source table that overlap with the destination table will change after the sort has been completed, as shown in the following example.

(1) Before Execution: 400005 is in Both Tables

(2) When input 1 turns ON, the data in the source tables will be placed in ascending order in the destination table. The data in holding register 400005 of the source table will be 100 as the result of the sort.

b) If input 3 is ON, the sort will not be executed even if input 1 turns ON. The data in the source and destination tables will not change and outputs 1 and 2 are ON as long as input 1 is ON.

7) The operation of SORT is outlined in the following table.

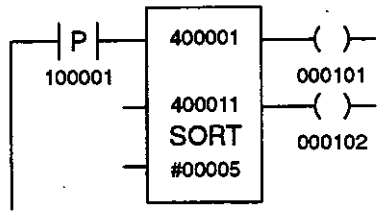
Table 8.7 Operation of SORT

Inputs			Conditions	Operation	Outputs	
1	2	2			1	2
ON	OFF	OFF	Source and destination tables do not overlap.	1) Source table data is sorted in ascending order and stored in the destination table. 2) Source table data does not change.	**ON**	OFF
	ON			1) Source table data is sorted in descending order and stored in the destination table. 2) Source table data does not change.		
	OFF		Source and destination tables overlap.	1) Source table data is sorted in ascending order and stored in the destination table. 2) Source table data changes for all registers that are also part of the destination table.		
	ON			1) Source table data is sorted in descending order and stored in the destination table. 2) Source table data changes for all registers that are also part of the destination table.		
	OFF	**ON**	Source and destination tables do not overlap.	1) Destination table data is sorted in ascending order and stored in the destination table. 2) Data in the sort table moves together with the registers of the destination table.		
	ON			1) Destination table data is sorted in descending order and stored in the destination table. 2) Data in the sort table moves together with the registers of the destination table.		
	Any		Source and destination tables overlap.	1) Nothing is sorted (sort error). 2) Source table and destination table data does not change.		**ON**
OFF	Any	Any	None	1) Nothing is sorted. 2) Source table and destination table data does not change.	OFF	OFF

4. Application Examples

◀EXAMPLE▶ **Example 1**
Source table data is sorted in ascending order.

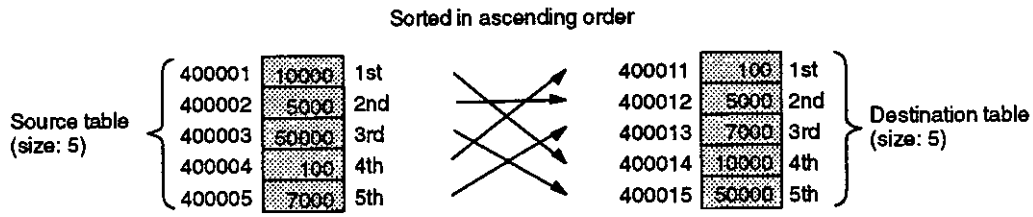
1) Ladder Programming



2) Operation

a) Before Execution

b) Data will be sorted as shown below when input relay 100001 changes from OFF to ON. The sort will be completed in one scan.



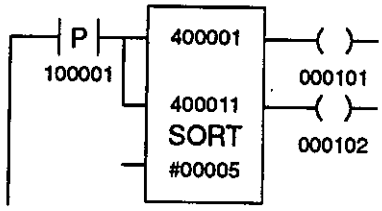
(1) The data in each register of the source table is treated as 16-bit binary data (0 to 65,535), the data is sorted into ascending order, and then the sorted result is stored in the destination table.

(2) The data in the source table does not change.

(3) Coil 000101 turns ON only for the scan in which input relay 100001 changed from OFF to ON. Coil 000102 remains OFF.

◀EXAMPLE▶ **Example 2**
Source table data is sorted in descending order.

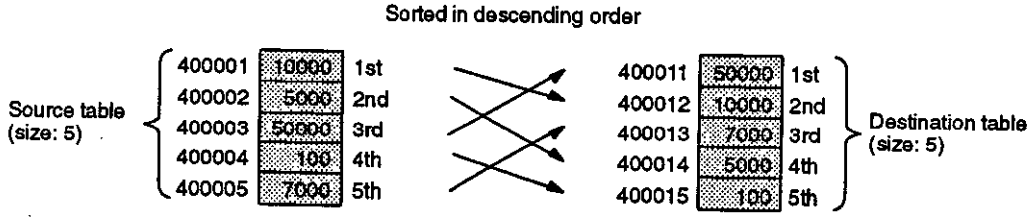
1) Ladder Programming



2) Operation

a) Before Execution

b) Data will be sorted as shown below when input relay 100001 changes from OFF to ON. The sort will be completed in one scan.



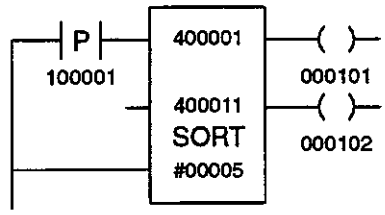
(1) The data in each register of the source table is treated as 16-bit binary data (0 to 65,535), the data is sorted into descending order, and then the sorted result is stored in the destination table.

(2) The data in the source table does not change.

(3) Coil 000101 turns ON only for the scan in which input relay 100001 changed from OFF to ON. Coil 000102 remains OFF.

◀EXAMPLE▶ **Example 3**
Destination table data is sorted in ascending order.

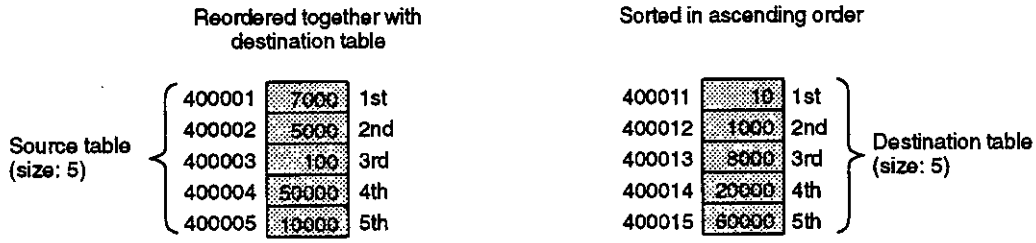
1) Ladder Programming



2) Operation

a) Before Execution

b) Data will be sorted as shown below when input relay 100001 changes from OFF to ON. The sort will be completed in one scan.



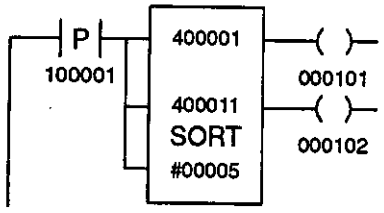
(1) The data in each register of the destination table is treated as 16-bit binary data (0 to 65,535), the data is sorted into ascending order, and then the sorted result is stored in the destination table.

(2) The data in the source table is also reordered in the same way as the registers in the destination table, i.e., data in corresponding registers moves together with the registers of the destination table.

(3) Coil 000101 turns ON only for the scan in which input relay 100001 changed from OFF to ON. Coil 000102 remains OFF.

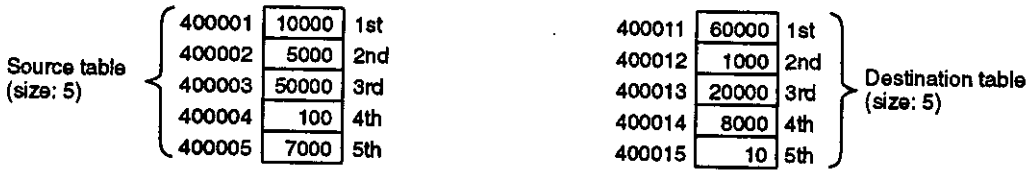
◀EXAMPLE▶ **Example 4**
Destination table data is sorted in descending order.

1) Ladder Programming

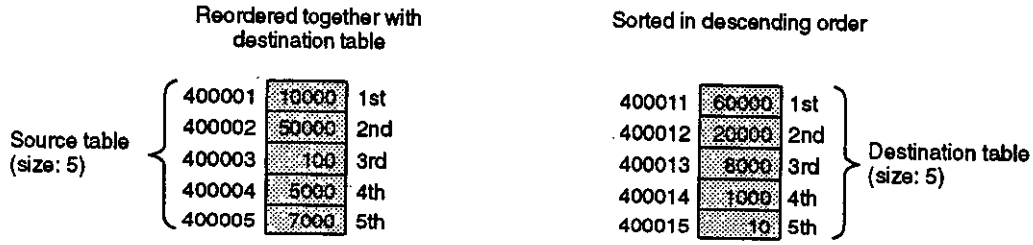


2) Operation

a) Before Execution



b) Data will be sorted as shown below when input relay 100001 changes from OFF to ON. The sort will be completed in one scan.



(1) The data in each register of the destination table is treated as 16-bit binary data (0 to 65,535), the data is sorted into descending order, and then the sorted result is stored in the destination table.

(2) The data in the source table is also reordered in the same way as the registers in the destination table, i.e., data in corresponding registers moves together with the registers of the destination table.

(3) Coil 000101 turns ON only for the scan in which input relay 100001 changed from OFF to ON. Coil 000102 remains OFF.

8.3.4 Building Programs

1. Storage Locations on Networks

A. LOGICAL BYTE REARRANGEMENT

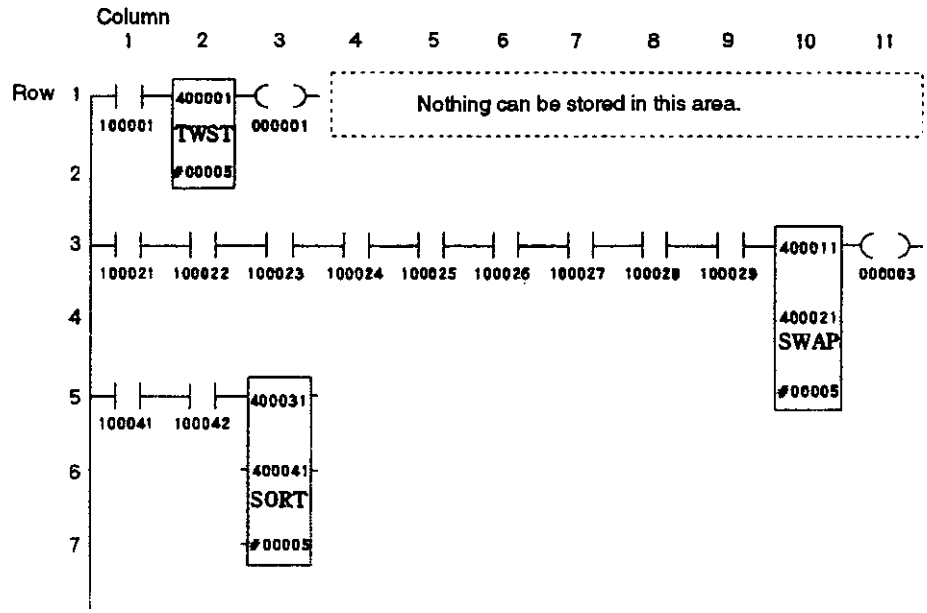
LOGICAL BYTE REARRANGEMENT instructions require two elements (top and bottom) located vertically on the network, so they can be stored anywhere on a 6-row by 10-column matrix (rows 1 through 6 and columns 1 through 10).

B. SWAP, SORT

SWAP, SORT instructions require three vertical elements on a network, one top element, one middle element, and one bottom element. They can thus be stored anywhere on a 5-row by 10-column matrix (rows 1 through 5 and columns 1 through 10) on the network.

Note LOGICAL BYTE REARRANGEMENT, SWAP and SORT instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

Example



2. Inputs

Inputs to LOGICAL BYTE REARRANGEMENT, SWAP and SORT instructions can be connected to relay elements (except coils) and/or outputs from timers, counters, math instructions, data manipulation instructions, other instructions, etc.

3. Outputs

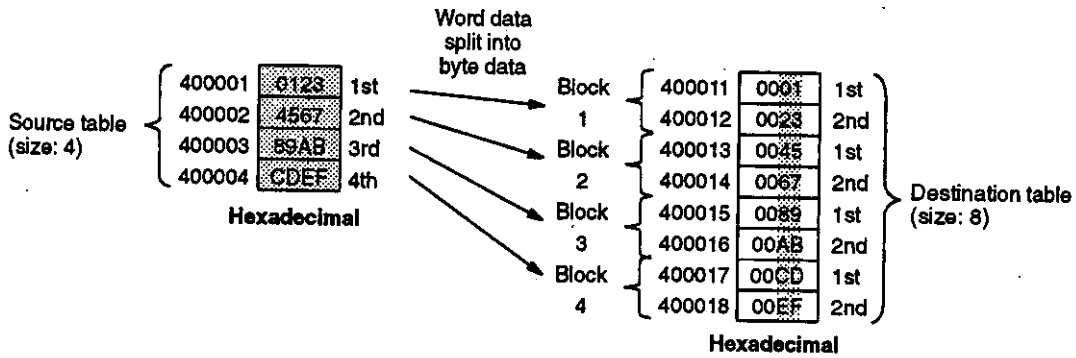
Outputs from LOGICAL BYTE REARRANGEMENT, SWAP, and SORT instructions can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data manipulation instructions, etc.

8.4 Data Split/Combine Instructions

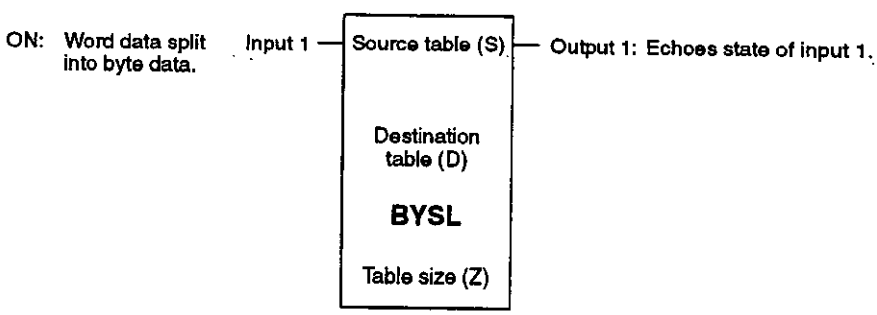
8.4.1 BYTE SPLIT (BYSL)

1. Function

The data (word data) in each register of the source table is split into bytes and then the bytes are stored in the two registers of the corresponding block in the destination table. The split is completed in one scan.



2. Structure



1) BYSL is the symbol for BYTE SPLIT.

2) BYSL requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. Refer to *Table 8.8* for details on specifying constants or registers for these elements.

Example

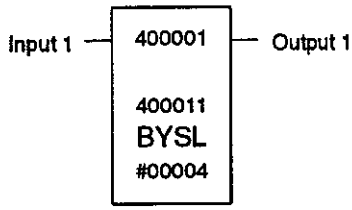


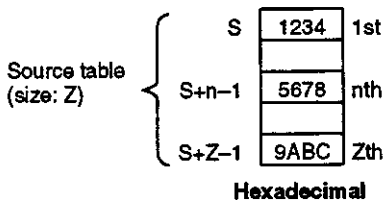
Table 8.8 Structural Elements of BYSL

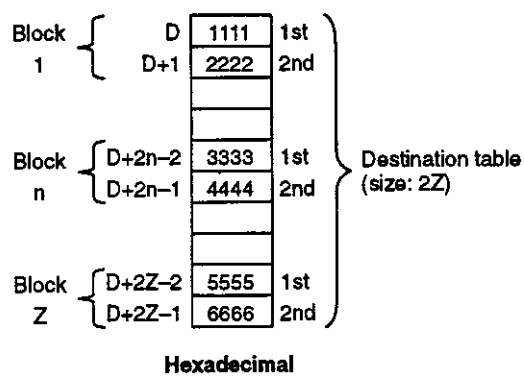
Element	Meaning	Possible Settings	
Top (S)	Leading reference number in source table	Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 or R20001 to R21024
Middle (D)	Leading reference number in destination table	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 or R20001 to R21023
Bottom (Z)	Size of source table	Constant:	#00001 to #00100

Note The destination table is twice the size of the source table.

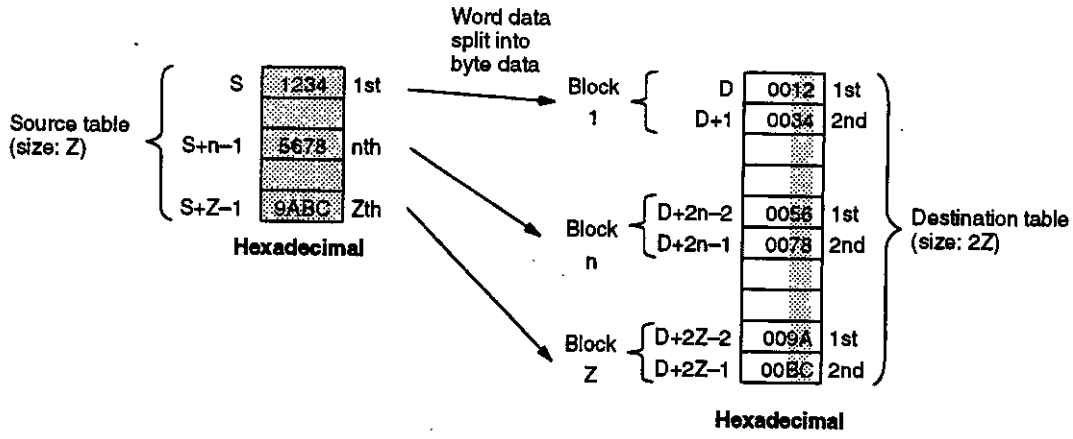
3. Operation

1) Status Before Execution





2) Data will be split as shown below when input 1 turns ON.

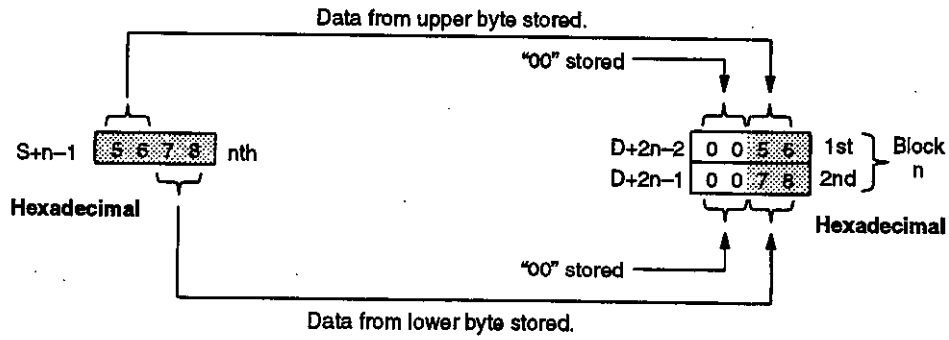


a) The word data in each register n (n = 1 to Z) of the source table is split into bytes and then the bytes are stored in the lower bytes of the two registers of block n in the destination table. "00" (hexadecimal) is stored in the upper bytes.

b) The data in the registers of the source table does not change.

c) Output 1 turns ON.

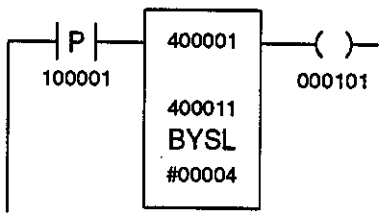
3) The bytes are split and stored as shown below.



◀EXAMPLE▶

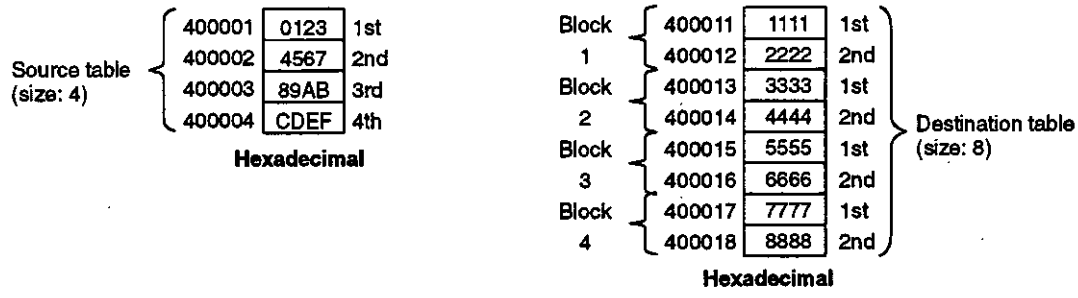
4. Application Example

1) Ladder Programming

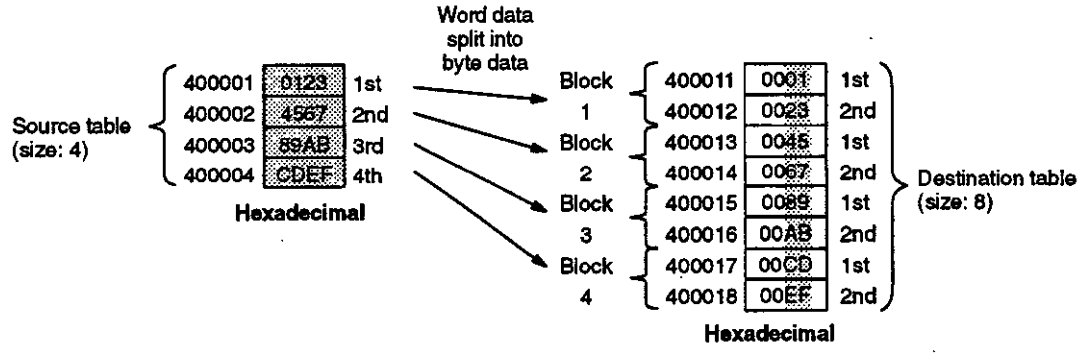


2) Operation

a) Before Execution



b) Data will be split as shown below when input relay 100001 changes from OFF to ON. The split will be completed in one scan.



(1) The word data in each register n (n = 1 to 4) of the source table is split into bytes and then the bytes is stored in the lower bytes of the two registers of block n in the destination table. "00" (hexadecimal) is stored in the upper bytes.

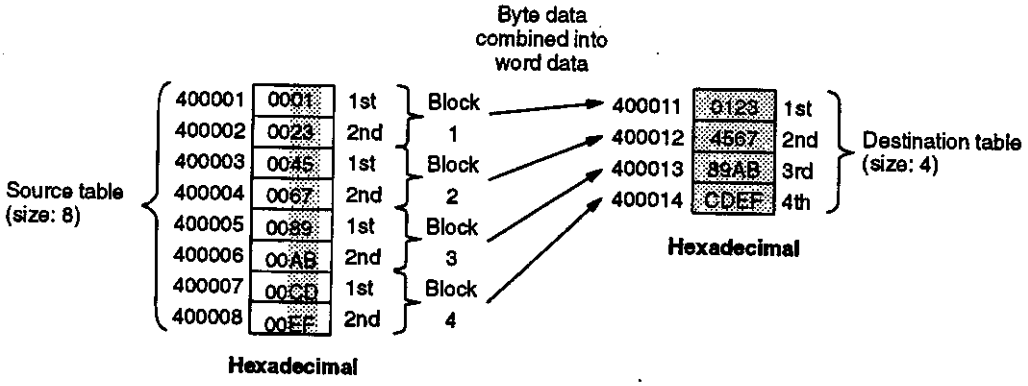
(2) The data in the source table does not change.

(3) Coil 000101 turns ON only for the scan in which input relay 100001 changed from OFF to ON.

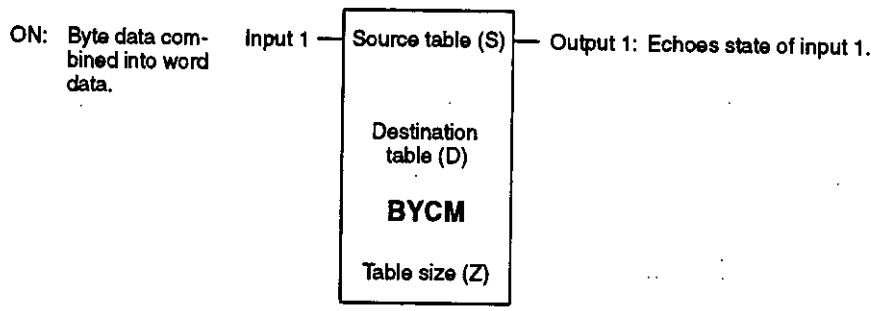
8.4.2 BYTE COMPOSITION (BYCM)

1. Function

The data in the lower bytes of the two registers in each block in the source table is combined into word data and the data is stored in the corresponding register in the destination table. Execution is completed in one scan.



2. Structure



1) BYCM is the symbol for BYTE COMPOSITION.

2) BYCM requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. Refer to *Table 8.9* for details on specifying constants or registers for these elements.

Example

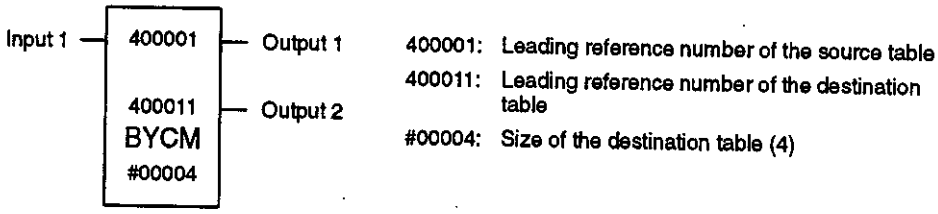


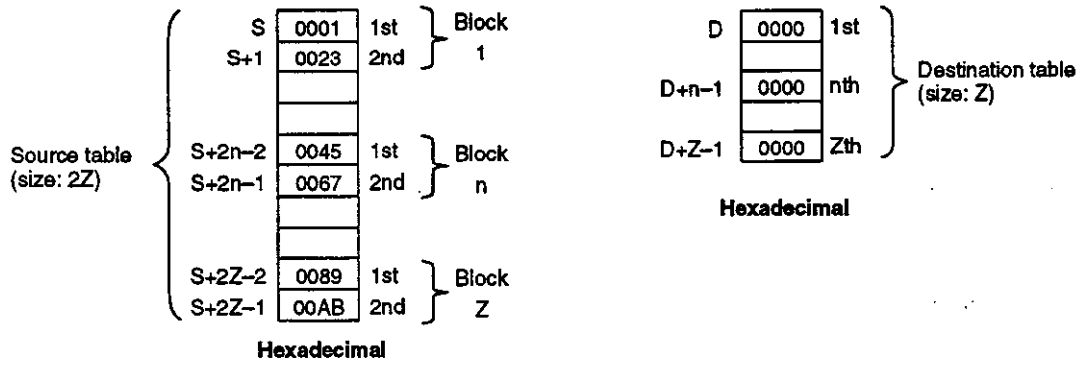
Table 8.9 Structural Elements of BYCM

Element	Meaning	Possible Settings	
Top (S)	Leading reference number in source table	Input register:	300001 to 300511 (Z00001 to Z00511)
		Holding register:	400001 to 409998 (W00001 to W09998)
		Constant register:	700001 to 704095 (K00001 to K04095)
		Link register:	R10001 to R11023 or R20001 to R21023
Middle (D)	Leading reference number in destination table	Holding register:	400001 to 409999 (W00001 to W09999)
		Link register:	R10001 to R11024 or R20001 to R21024
Bottom (Z)	Size of destination table	Constant:	#00001 to #00100

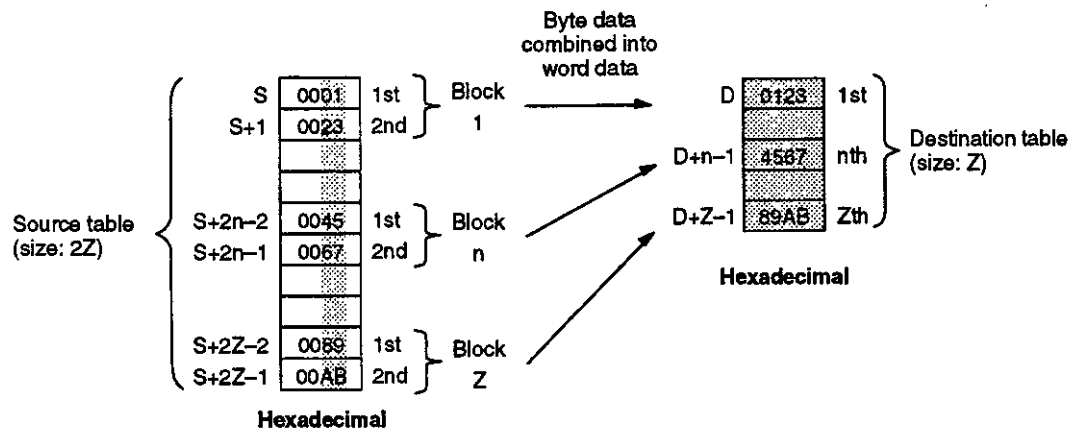
Note The source table is twice the size of the destination table.

3. Operation

1) Status Before Execution



2) Data will be combined as shown below when input 1 turns ON.

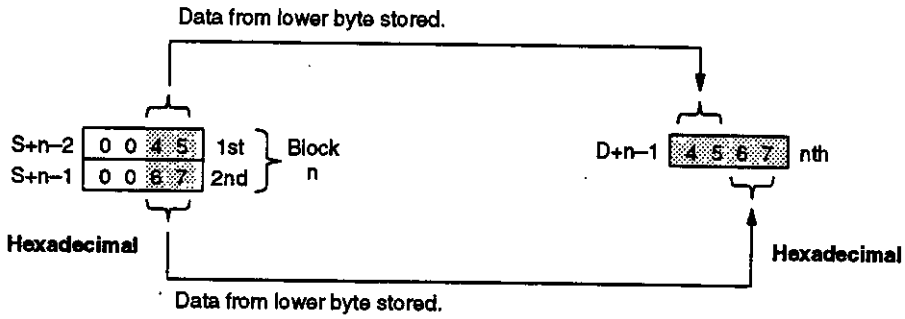


a) The data in the lower bytes of the two registers in each block n (n = 1 to Z) in the source table is combined into word data and the data is stored in nth register in the destination table.

b) The data in the registers of the source table does not change.

c) Output 1 turns ON.

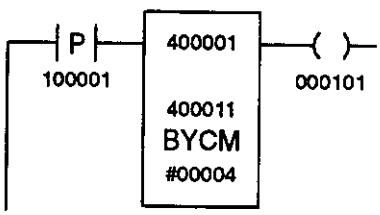
3) The bytes are combined and stored as shown below.



◀EXAMPLE▶

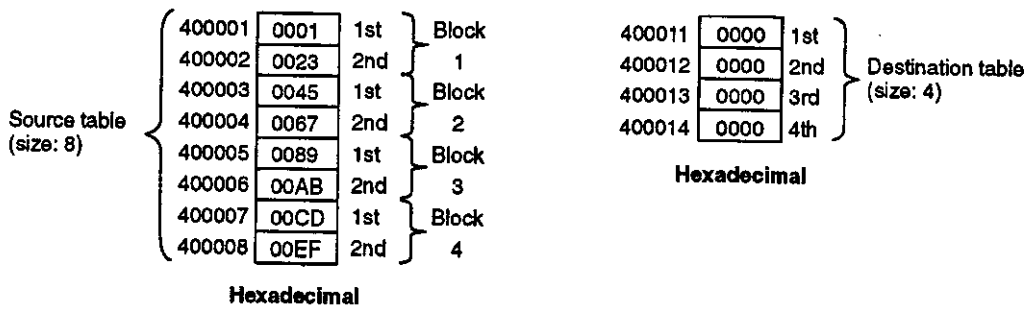
4. Application Example

1) Ladder Programming

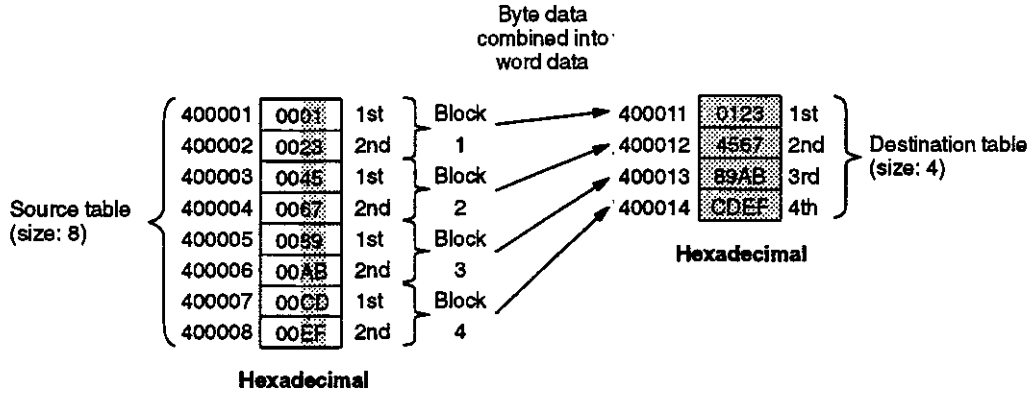


2) Operation

a) Before Execution



b) Data will be combined as shown below when input relay 100001 changes from OFF to ON. The combination will be completed in one scan.



(1) The data in the lower bytes of the two registers in each block n (n = 1 to 4) in the source table is combined into word data and the data is stored in register n in the destination table.

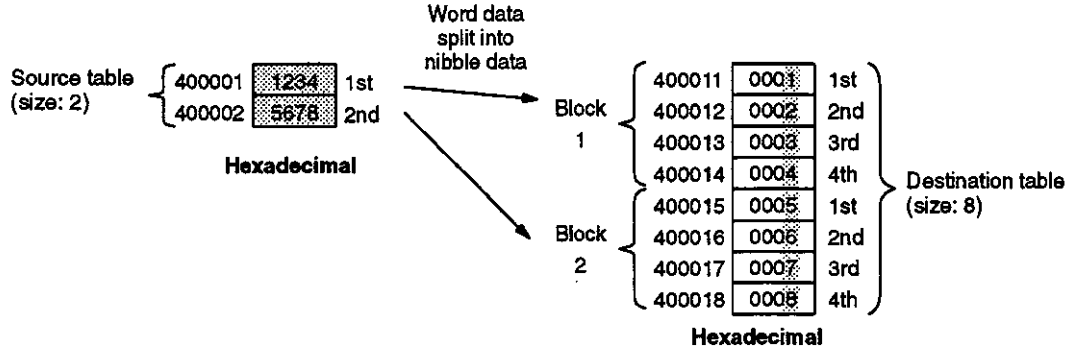
(2) The data in the source table does not change.

(3) Coil 000101 turns ON only for the scan in which input relay 100001 changed from OFF to ON.

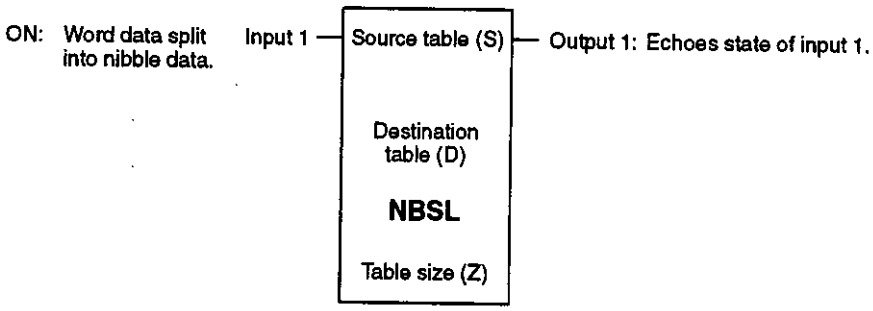
8.4.3 NIBBLE SPLIT (NBSL)

1. Function

The data (word data) for each register or 16 coils/relays of the source table is split into four nibbles (4-bit data) and then the nibbles are stored in the four registers of the corresponding block in the destination table. The split is completed in one scan.



2. Structure



1) NBSL is the symbol for NIBBLE SPLIT.

2) NBSL requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. Refer to *Table 8.10* for details on specifying constants, coils, relays, or registers for these elements.

Example

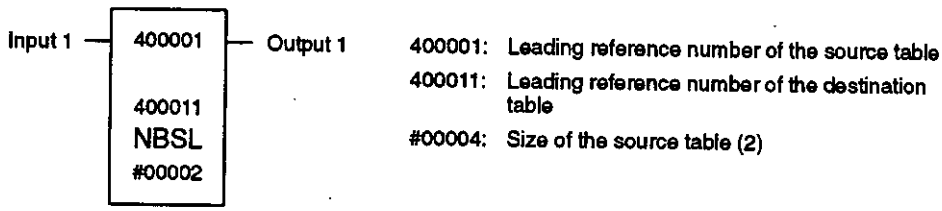


Table 8.10 Structural Elements of NBSL

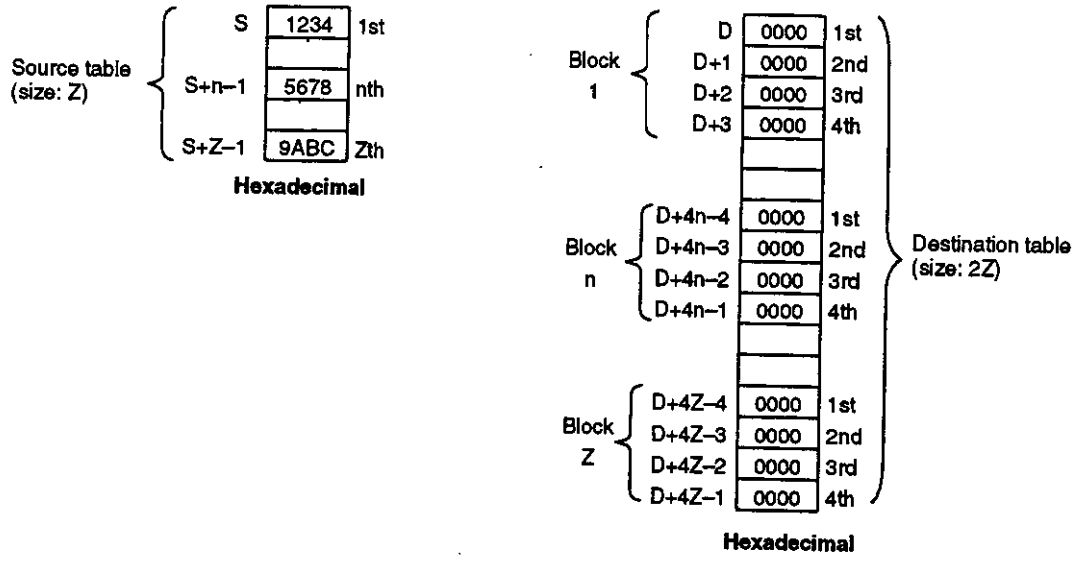
Element	Meaning	Possible Settings	
Top (S)	Leading reference number of source table	Coil:	000001 to 008177 (O00001 to O08177)
		Input relay:	100001 to 101009 (I00001 to I01009)
		Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
		MC relay:	X10001 to X10241 or X20001 to X20241
		MC control relay:	P10001 to P10241 or P20001 to P20241
		M code relay:	M10001 to M10081 or M20001 to M20081
Middle (D)	Leading reference number of destination table	Holding register:	400001 to 409996 (W00001 to W09996)
		Link register:	R10001 to R11021 or R20001 to R21021
Bottom (Z)	Size of source table	A constant must be specified. The maximum value of the constant that can be specified depends on the type of reference that is used.	
		Coil:	#00001 to #00100
		Input relay:	#00001 to #00064
		Input register, holding register or constant register:	#00001 to #00100
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00100
		MC coil, MC relay or MC control relay:	#00001 to #00016
		MC control coil:	#00001 to #00010
		M code relay:	#00001 to #00006

Note (1) When a coil or relay is being specified, the last 5 digits of the reference number must be 16n +1 (where n = 0, 1, 2, ...).

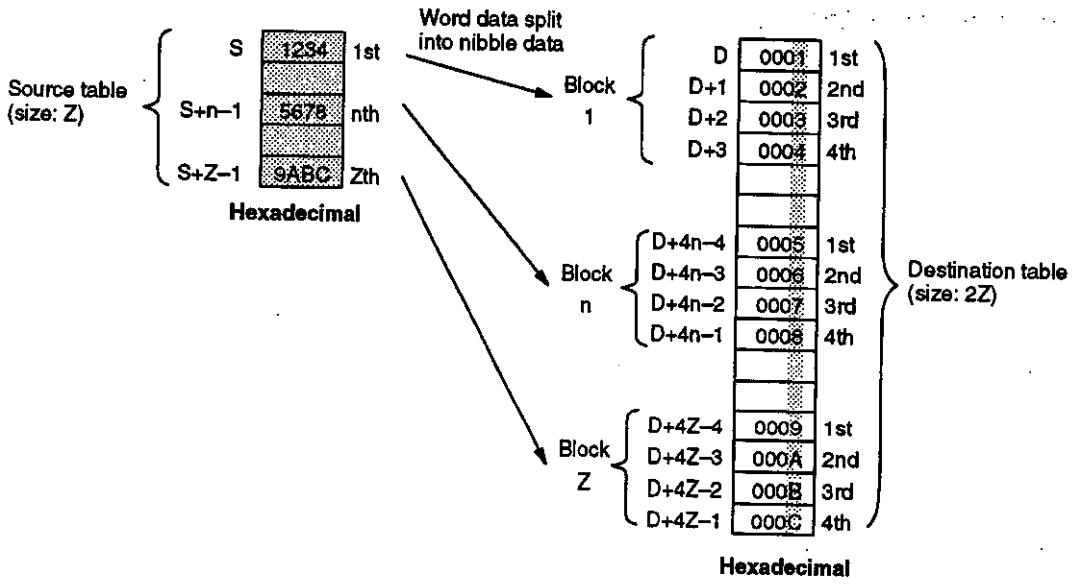
(2) The destination table is four times the size of the source table.

3. Operation

1) Status Before Execution



2) Data will be split as shown below when input 1 turns ON.

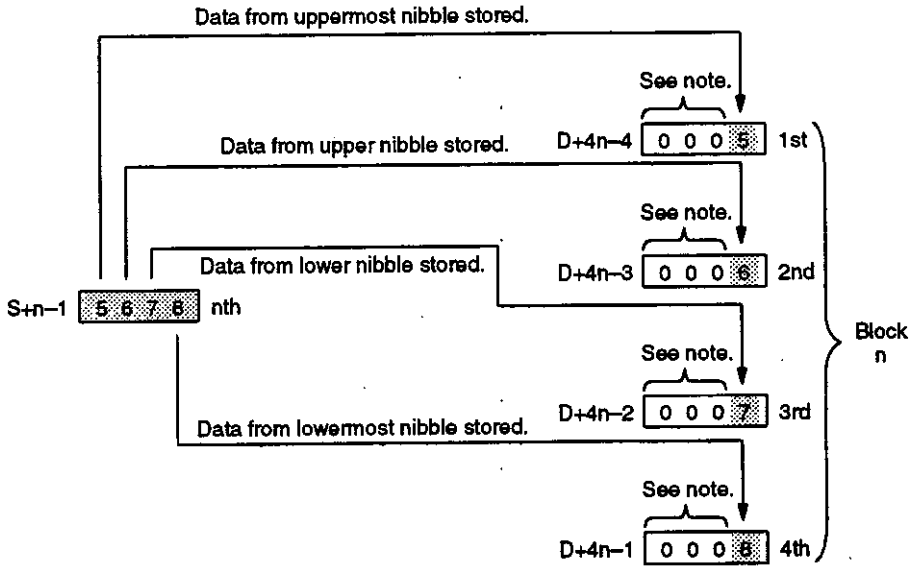


a) The word data in each nth register (n = 1 to Z) of the source table is split into four nibbles (4-bit data) and then the nibbles are stored in the lower four bits of the four registers of block n in the destination table. "000" (hexadecimal) is stored in the upper 12 bits.

b) The data in the registers of the source table does not change.

c) Output 1 turns ON.

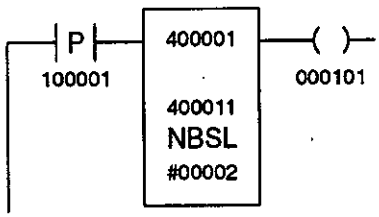
3) The nibbles are split and stored as shown below.



Note "000" (hexadecimal) is stored.

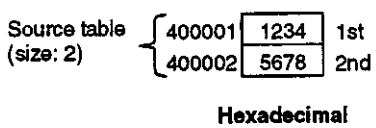
◀EXAMPLE▶ 4. Application Example

1) Ladder Programming

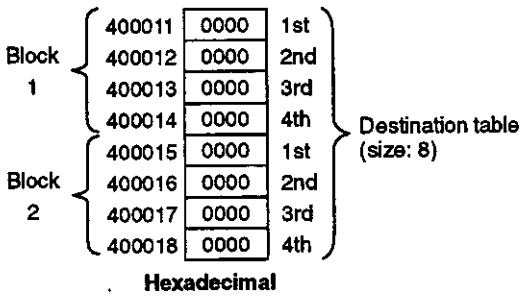


2) Operation

a) Before Execution

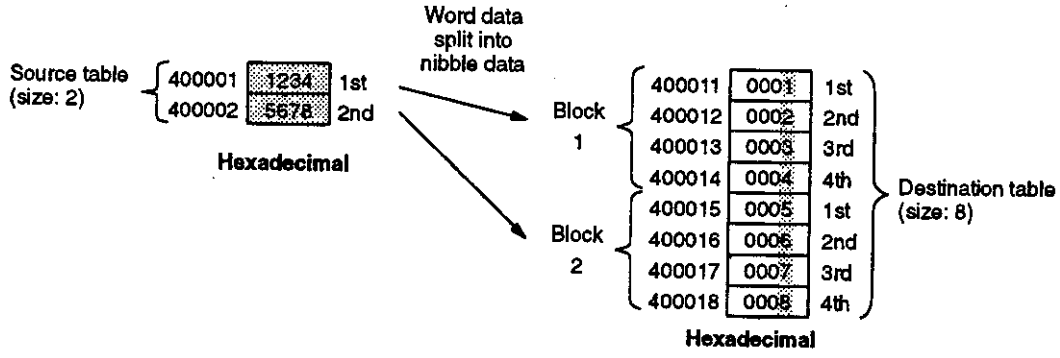


Hexadecimal



Hexadecimal

b) Data will be split as shown below when input relay 100001 changes from OFF to ON. The split will be completed in one scan.



(1) The word data in each nth register (n = 1 to 4) of the source table is split into four nibbles (4-bit data) and then the nibbles is stored in the lower four bits of the four registers of block n in the destination table. "00" (hexadecimal) is stored in the upper nibbles.

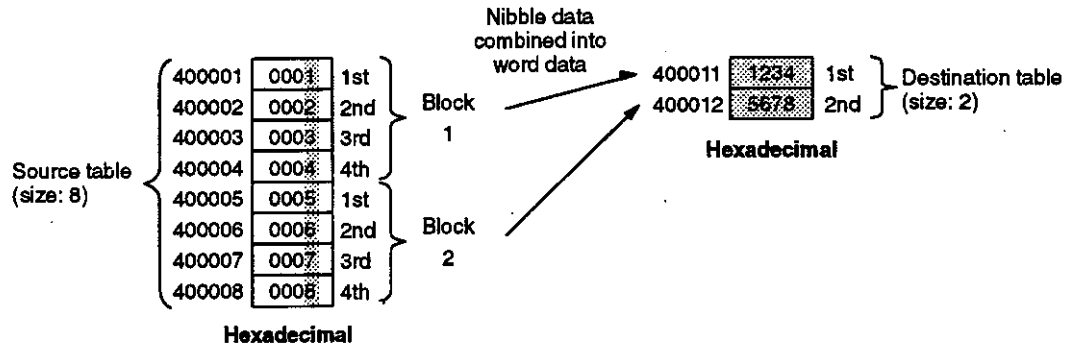
(2) The data in the source table does not change.

(3) Coil 000101 turns ON only for the scan in which input relay 100001 changed from OFF to ON.

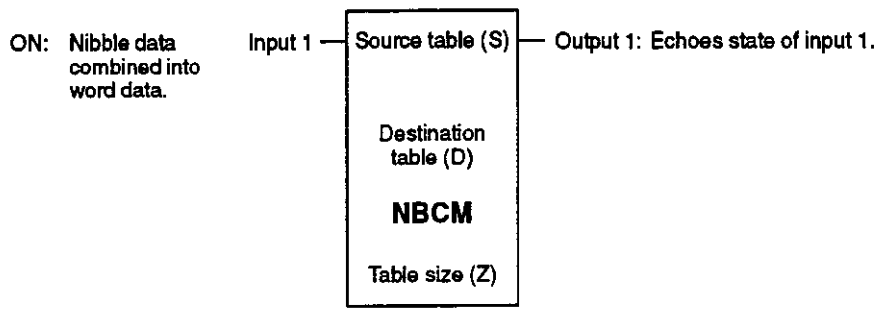
8.4.4 NIBBLE COMPOSITION (NBCM)

1. Function

The data in the lower four bits (nibbles) of the four registers in each block in the source table is combined into word data and the data is stored in the corresponding register or 16 coils/relays in the destination table. Execution is completed in one scan.



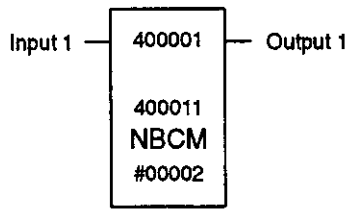
2. Structure



1) NBCM is the symbol for NIBBLE COMPOSITION.

2) NBCM requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. Refer to *Table 8.11* for details on specifying constants, coils, relays, or registers for these elements.

Example



400001: Leading reference number of the source table
400011: Leading reference number of the destination table
#00004: Size of the destination table (2)

Table 8.11 Structural Elements of NBCM

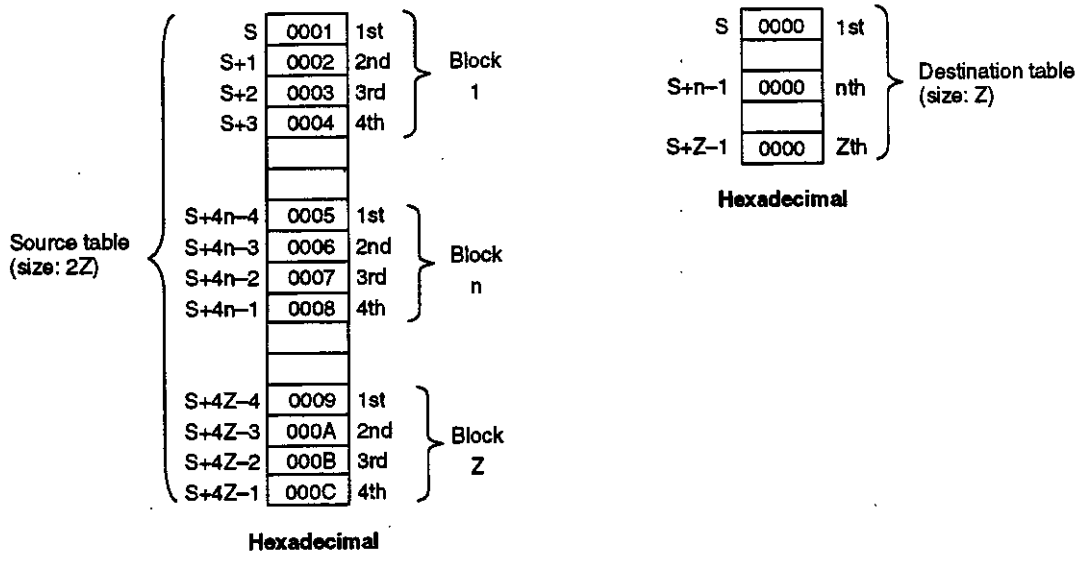
Element	Meaning	Possible Settings	
Top (S)	Leading reference number of source table	Input register:	300001 to 300509 (Z00001 to Z00509)
		Holding register:	400001 to 409996 (W00001 to W09996)
		Constant register:	700001 to 704093 (K00001 to K04093)
		Link register:	R10001 to R11021 or R20001 to R21021
Middle (D)	Leading reference number of destination table	Coil:	000001 to 008177 (O00001 to O08177)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Link coil:	D10001 to D11009 or D20001 to D21009
		Link register:	R10001 to R11024 or R20001 to R21024
		MC coil:	Y10001 to Y10241 or Y20001 to Y20241
		MC control coil:	Q10001 to Q10145 or Q20001 to Q20145
Bottom (Z)	Size of destination table	A constant must be specified. maximum value of constant that can be specified depends on type of reference that is used.	
		Coil:	#00001 to #00100
		Holding register:	#00001 to #00100
		Link coil:	#00001 to #00064
		Link register:	#00001 to #00100
		MC coil:	#00001 to #00016
		MC control coil:	#00001 to #00010

Note (1) When a coil or relay is being specified, the last 5 digits of the reference number must be 16n +1 (where n = 0, 1, 2, ...).

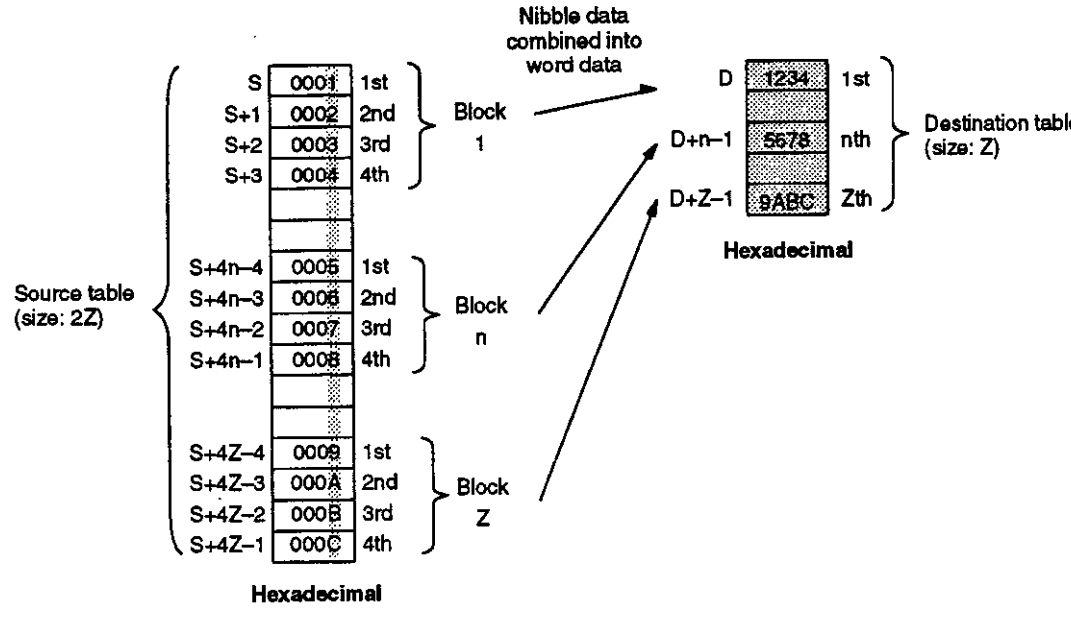
(2) The source table is four times the size of the destination table.

3. Operation

1) Status Before Execution



2) Data will be combined as shown below when input 1 turns ON.

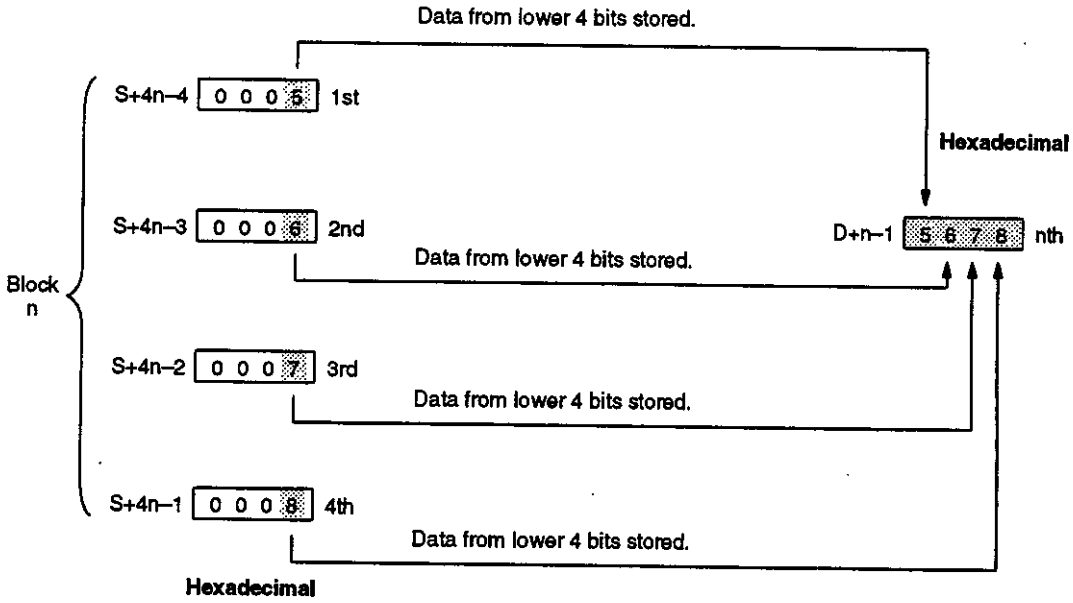


a) The data in the lower four bits (nibbles) of the four registers in each block n (n = 1 to Z) in the source table is combined into word data and the data is stored in register n in the destination table.

b) The data in the registers of the source table does not change.

c) Output 1 turns ON.

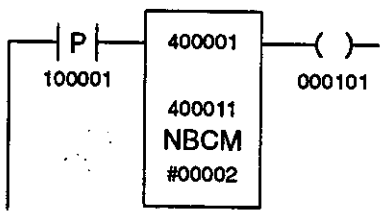
3) The nibbles are combined and stored as shown below.



◀EXAMPLE▶

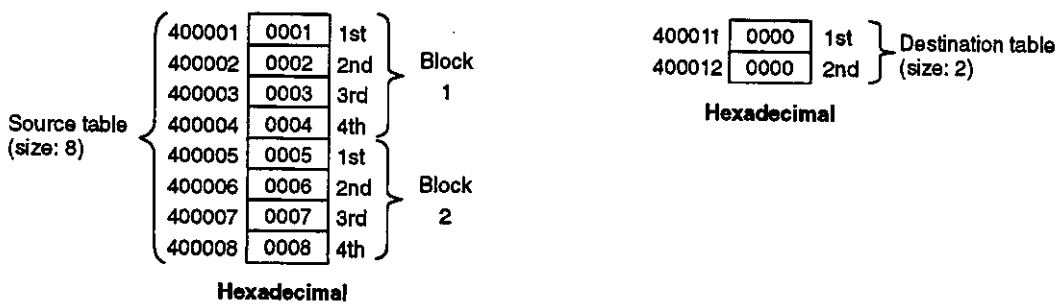
4. Application Example

1) Ladder Programming

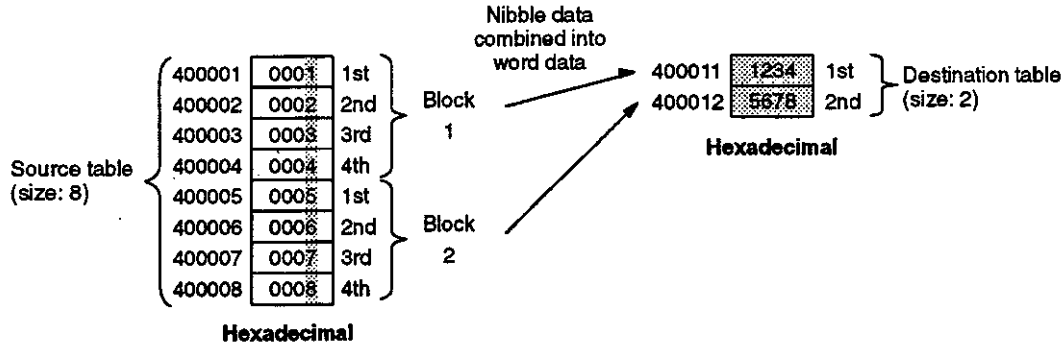


2) Operation

a) Before Execution



b) Data will be combined as shown below when input relay 100001 changes from OFF to ON. The combination will be completed in one scan.



(1) The data in the lower four bits (nibbles) of the four registers in each block n (n = 1 and 2) in the source table is combined into word data and the data is stored in nth register in the destination table.

(2) The data in the source table does not change.

(3) Coil 000101 turns ON only for the scan in which input relay 100001 changed from OFF to ON.

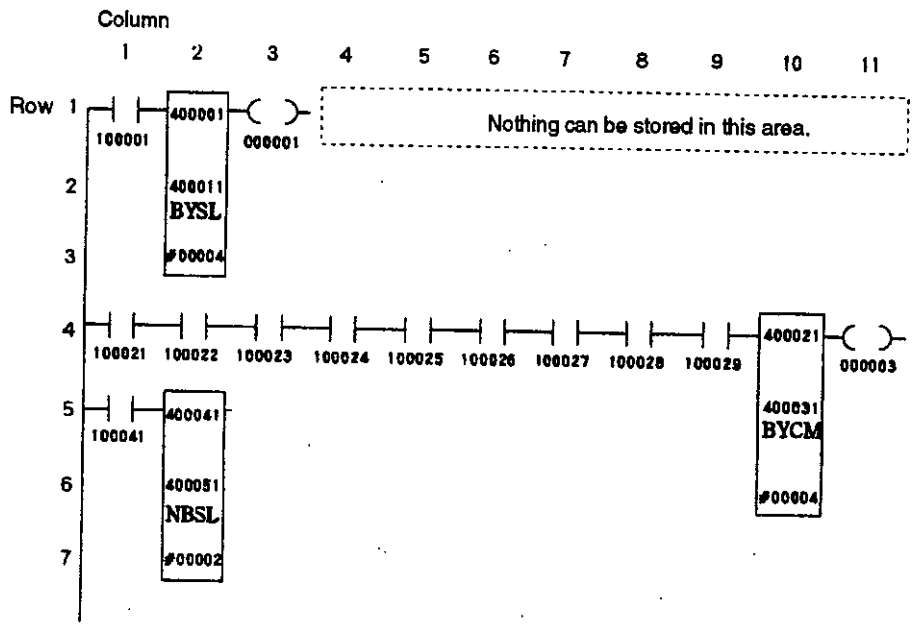
8.4.5 Building Programs

1. Storage Locations on Networks

BYTE SPLIT, BYTE COMPOSITION, NIBBLE SPLIT and NIBBLE COMPOSITION instructions require three vertical elements on a network, one top element, one middle element, and one bottom element. They can thus be stored anywhere on a 5-row by 10-column matrix (rows 1 through 5 and columns 1 through 10) on the network.

Note BYTE SPLIT, BYTE COMPOSITION, NIBBLE SPLIT and NIBBLE COMPOSITION instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

Example



2. Inputs

Inputs to BYTE SPLIT, BYTE COMPOSITION, NIBBLE SPLIT and NIBBLE COMPOSITION instructions can be connected to relay elements (except coils) and/or outputs from timers, counters, math instructions, data manipulation instructions, other instructions, etc.

3. Outputs

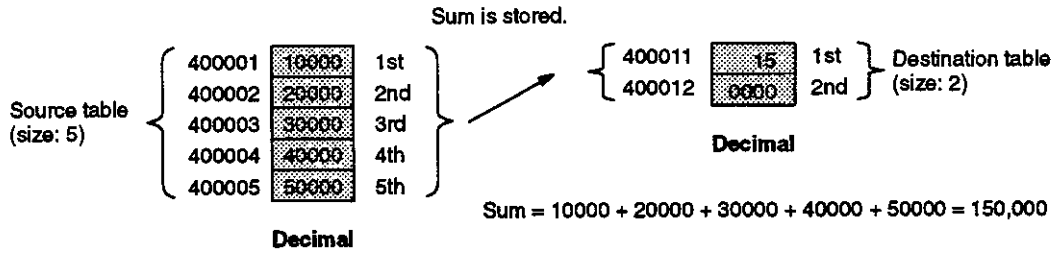
Outputs from BYTE SPLIT, BYTE COMPOSITION, NIBBLE SPLIT and NIBBLE COMPOSITION instructions can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data manipulation instructions, etc.

8.5 Block Addition and Check Calculation Instructions

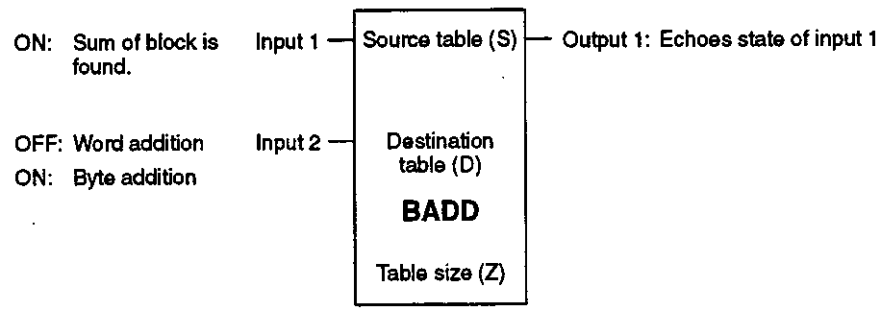
8.5.1 BLOCK ADD (BADD)

1. Function

The data in registers of the source table is added by word or by byte in unsigned addition and the result is stored in the two registers of the destination table. Execution is completed in one scan.



2. Structure



1) BADD is the symbol for BLOCK ADD.

2) BADD requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. Refer to *Table 8.12* for details on specifying constants or registers for these elements.

Example

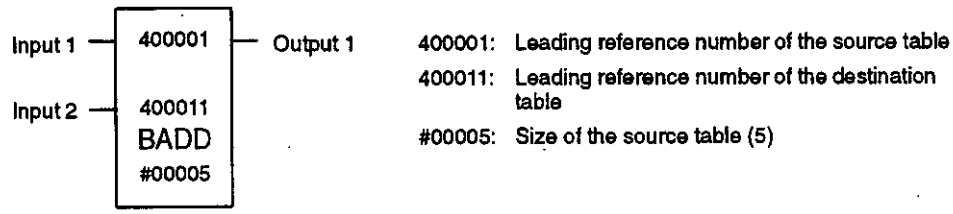


Table 8.12 Structural Elements of BADD

Element	Meaning	Possible Settings	
Top (S)	Leading reference number in source table	Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 or R20001 to R21024
Middle (D)	Leading reference number in destination table	Holding register:	400001 to 409998 (W00001 to W09998)
		Link register:	R10001 to R11023 or R20001 to R21023
Bottom (Z)	Size of source table	Constant:	#00001 to #00100

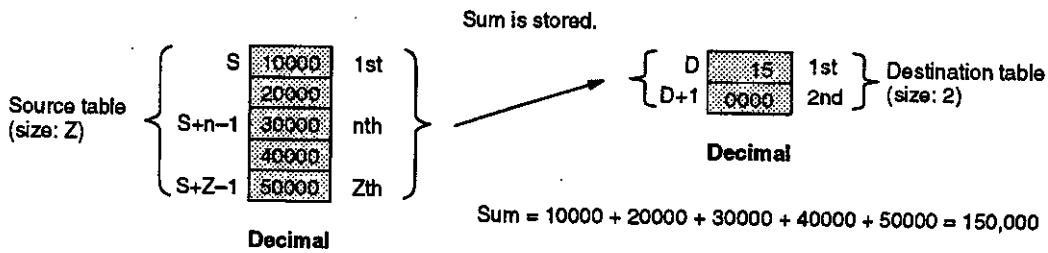
Note The destination table is always two registers in size.

3. Operation

1) Status Before Addition

2) Data will be added as shown below when input 1 turns ON.

a) Word data will be added if input 2 is OFF.



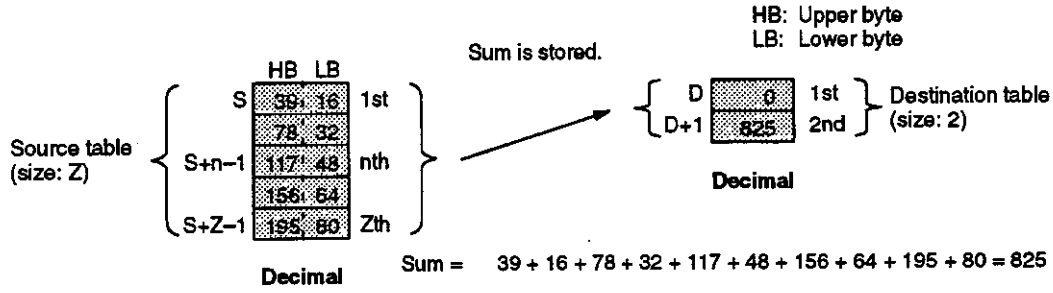
(1) The word data in the each register of the source table will be taken as an unsigned integer (0 to 65,535 when expressed in decimal) and the data in all of the registers will be added.

(2) The upper four digits of the sum are stored in register D and the lower four digits are stored in D+1.

(3) The data in the registers of the source table does not change.

(4) Output 1 is ON as long as input 1 is ON.

b) Byte data will be added if input 2 is ON.



(1) Each register in the source table is split into two bytes.

(2) Each byte data is taken as an unsigned integer (0 to 255 when expressed in decimal) and all the bytes from the source table are added.

(3) The upper four digits of the sum are stored in register D and the lower four digits are stored in D+1.

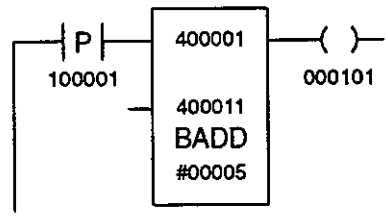
(4) The data in the registers of the source table does not change.

(5) Output 1 is ON as long as input 1 is ON.

4. Application Example

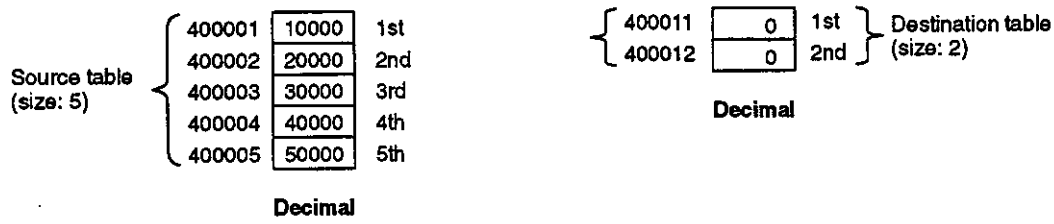
◀EXAMPLE▶ **Example 1: Word Addition**

1) Ladder Programming

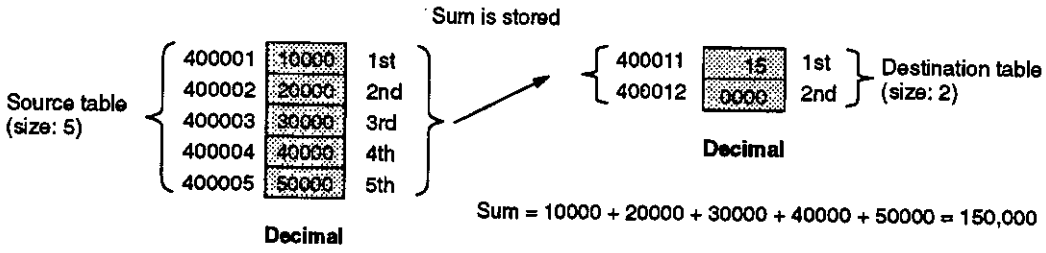


2) Operation

a) Before Execution



b) The following block addition will be performed when input relay 100001 changes from OFF to ON. The addition will be completed in one scan.



(1) The word data in the each register of the source table is taken as an unsigned integer (0 to 65,535 when expressed in decimal) and the data in all of the registers is added.

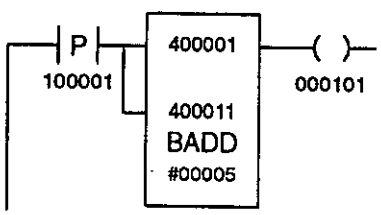
(2) The upper four digits of the sum is stored in first register of the destination table and the lower four digits is stored in the second register.

(3) The data in the source table does not change.

(4) Coil 000101 turns ON only for the scan in which input relay 100001 changed from OFF to ON.

◀EXAMPLE▶ **Example 2: Byte Addition**

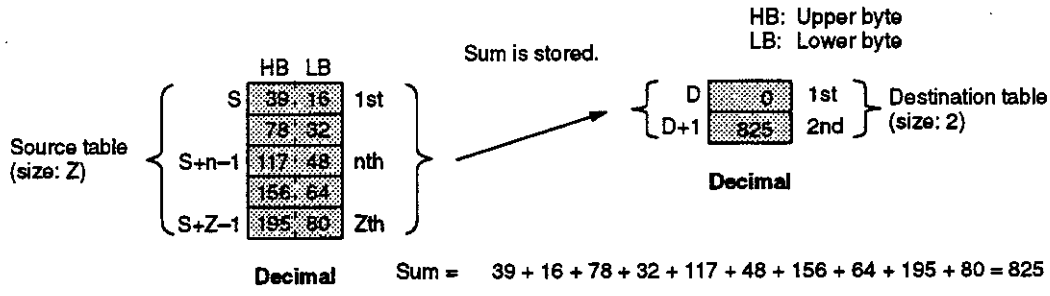
1) Ladder Programming



2) Operation

a) Before Execution

b) The following block addition will be performed when input relay 100001 changes from OFF to ON. The addition will be completed in one scan.



(1) The word data in the each register of the source table is split into two bytes.

(2) Each byte data from the source table is taken as an unsigned integer (0 to 255 when expressed in decimal) and all the bytes is added.

(3) The upper four digits of the sum is stored in first register of the destination table and the lower four digits is stored in the second register.

(4) The data in the source table does not change.

(5) Coil 000101 turns ON only for the scan in which input relay 100001 changed from OFF to ON.

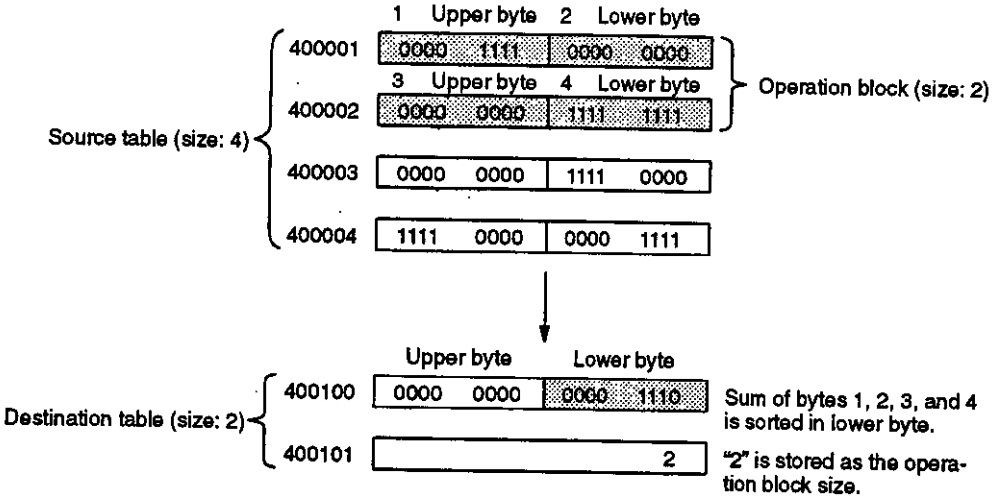
8.5.2 CHECKSUM (CKSM)

1. Function

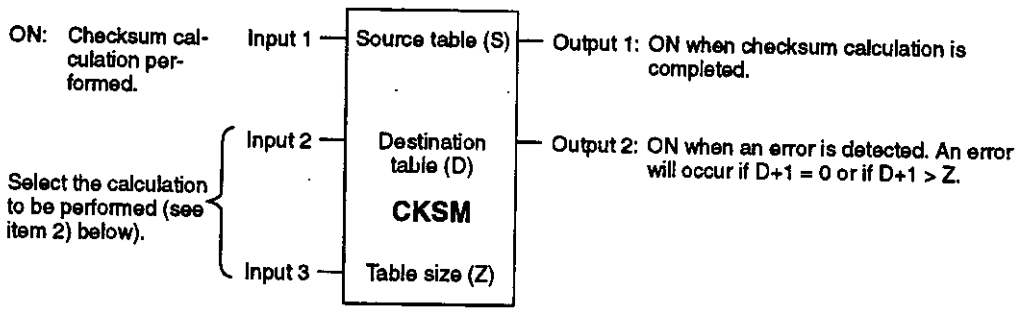
Straight check calculation, binary Addition check calculation, CRC-16 (cyclic redundancy check) calculation, or LRC (longitudinal redundancy check) calculation is performed for the data in the source table. Execution is completed in one scan.

Example:
For the straight check calculation, the operation block data is added by byte and the result is

stored in the lower byte of the leading register of the destination table. "00" (hexadecimal) is stored in the upper byte.



2. Structure



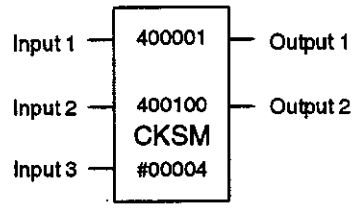
1) CKSM is the symbol for CHECKSUM.

2) The type of checksum calculation to be performed is specified by combining inputs 2 and 3.

Input 2	Input 3	Checksum Calculation
OFF	ON	Straight check
ON	ON	Binary addition check
ON	OFF	CRC-16 (cyclic redundancy)
OFF	OFF	LRC (longitudinal redundancy)

3) CKSM requires three elements, one top element, one middle element, and one bottom element, located vertically on the network. Refer to *Table 8.13* for details on specifying constants or registers for these elements.

Example



400001: Leading reference number of the source table
400100: Leading reference number of the destination table
#00004: Size of the source table (4)

Table 8.13 Structural Elements of CKSM

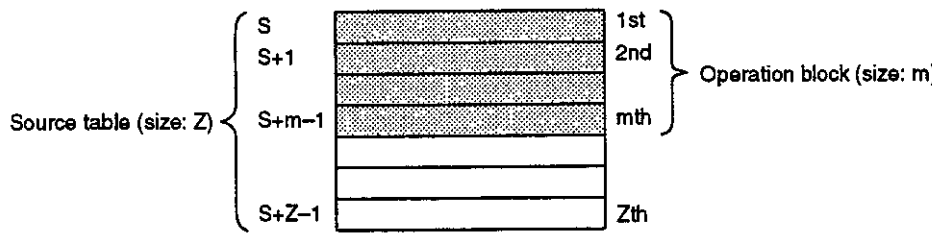
Element	Meaning	Possible Settings
Top (S)	Leading reference number in source table	Holding register: 400001 to 409999 (W00001 to W09999) Constant register: 700001 to 704096 (K00001 to K04096) Link register: R10001 to R11024 or R20001 to R21024
Middle (D)	Leading reference number in destination table (size: 2)	Holding register: 400001 to 409998 (W00001 to W09998) Link register: R10001 to R11023 or R20001 to R21023
Bottom (Z)	Size of source table	Constant: #00001 to #00255

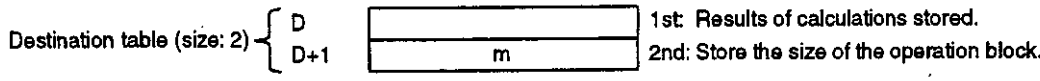
4) Destination Table

a) The destination table is a register table consisting of 2 registers.

b) The results of the checksum calculation is stored in the leading register.

c) Store the size (m) of the block for the checksum (called the operation block) in the second register. The value of m must be between 1 and Z.

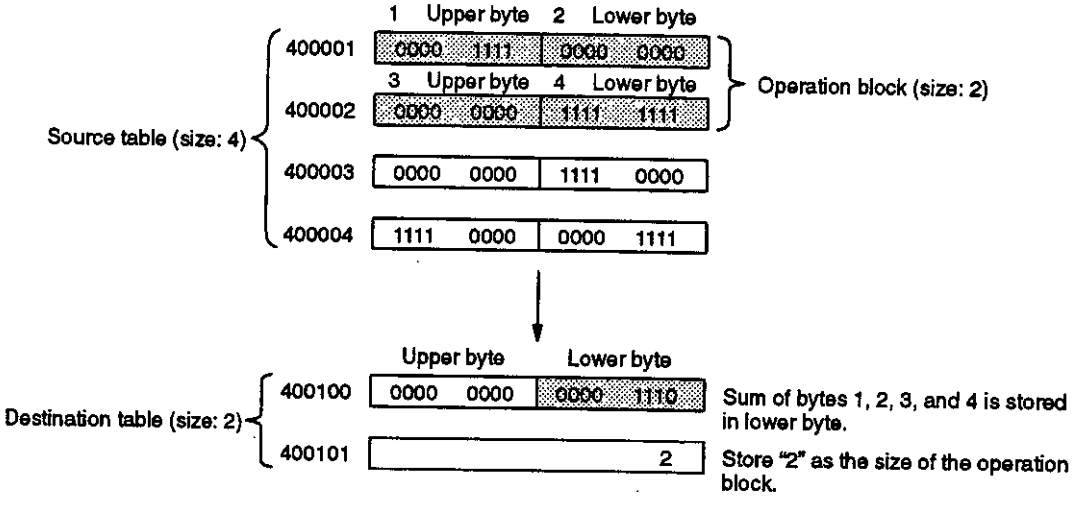




3. Operation

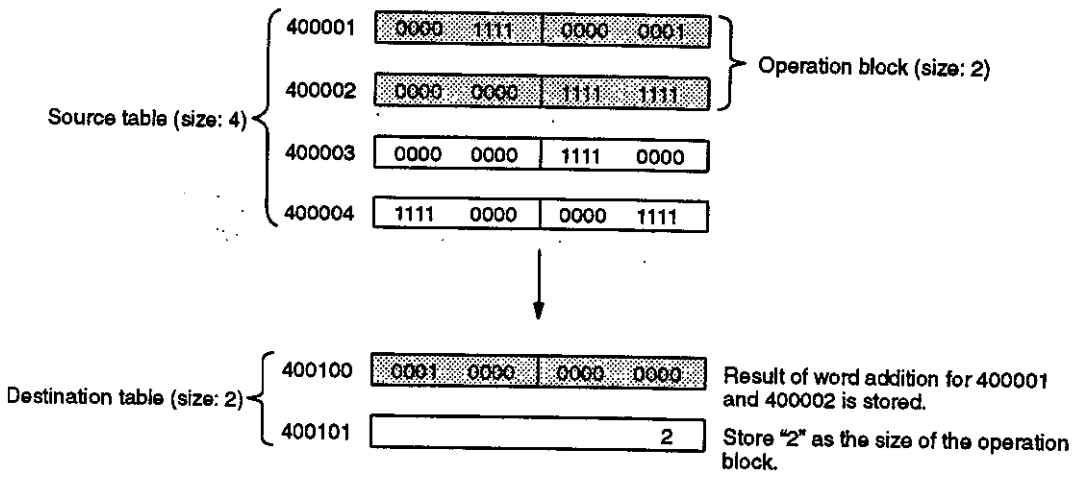
1) Straight Check Calculation (Input 2 OFF and Input 3 ON)

The operation block is added by byte and the result is stored in the lower byte of the leading register of the destination table. "00" (hexadecimal) is stored in the upper byte. The data in the source table does not change.



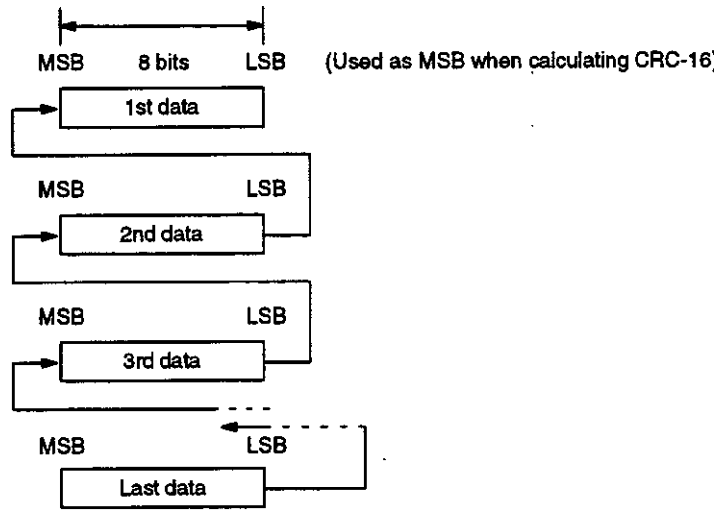
2) Binary Check Calculation (Input 2 ON and Input 3 ON)

The operation block is added by word and the result is stored in the leading register of the destination table. The data in the source table does not change.



3) CRC-16 (Input 2 ON and Input 3 OFF)

a) The CRC-16 (cyclic redundancy check) value is found, i.e., the data in the operation block is formed into one series of 8-bit bytes, which is divided by a specific 17-bit binary number (1 1000 0000 0000 0101) to find a 16-bit remainder. The calculation method used for CRC-16 is given in item b), below, and an example is given in item c), below.



b) CRC-16 Calculation Method

(1) The 16 bits of the remainder are all set to 1 to initialize the value.

(2) An exclusive OR is taken between the first data and the remainder.

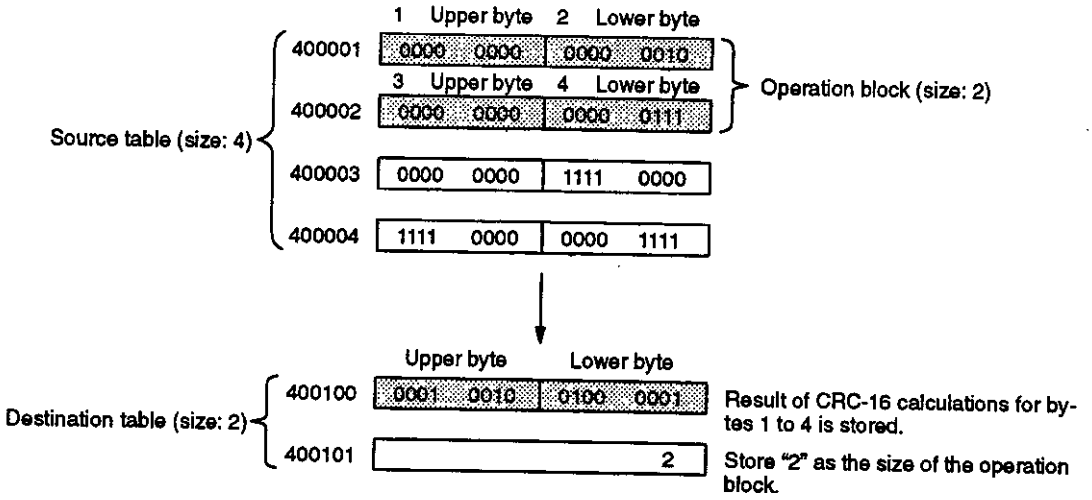
(3) The result of the exclusive OR is shifted 1 digit at a time to the right until a 1 is shifted out.

(4) An exclusive OR is taken between the result of the above shift operation and the lower 16 bits of the constant defined by CRC-16 (1 1000 0000 0000 0101).

(5) After 8 shifts to the right (and if 1 is shifted out, the exclusive OR in step (4) will be performed), exclusive OR is taken between the current results and the next 8 bits (function code).

(6) The above procedure is repeated through the last data.

c) CRC-16 Calculation Example

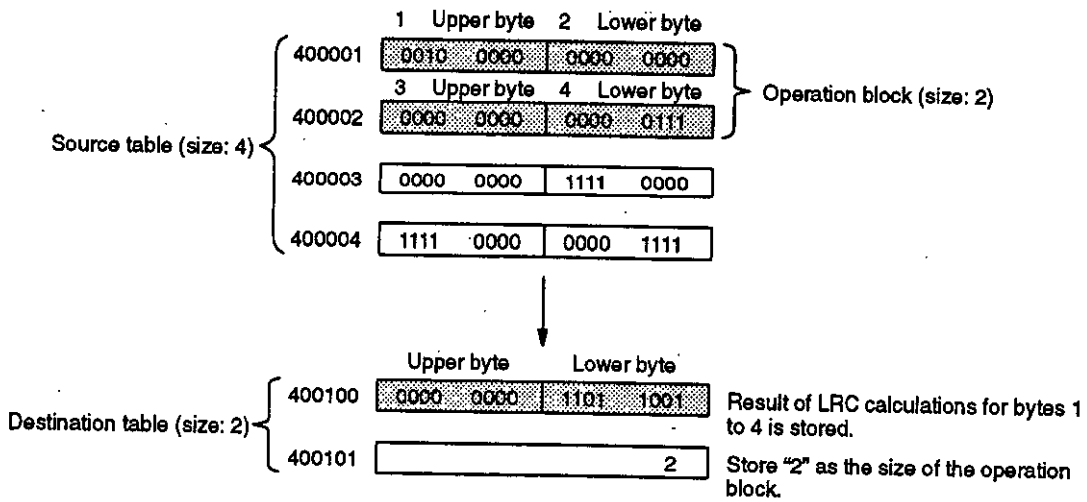


4) LRC (Input 2 OFF and Input 3 OFF)

a) The LRC (longitudinal redundancy check) value is found, i.e., the data in the operation block is added by byte (8 bits) and then the two's complement of the result is taken. Any overflows are ignored.

b) The results of adding all bytes in the operation block and the bytes of the LRC will result in 0.

c) LRC Calculation Example



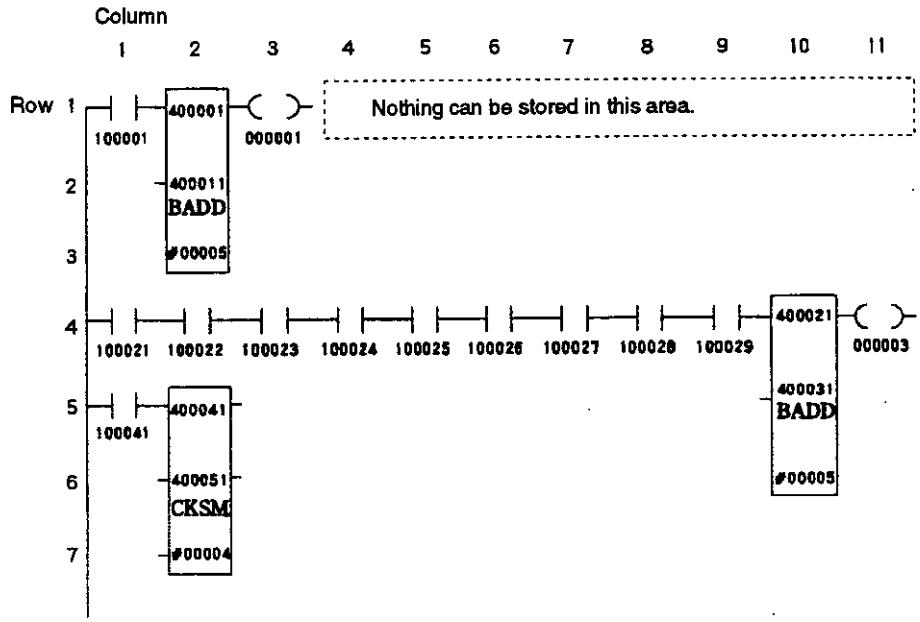
8.5.3 Building Programs

1. Storage Locations on Networks

BLOCK ADD and CHECKSUM instructions require three vertical elements on a network, one top element, one middle element, and one bottom element. They can thus be stored anywhere on a 5-row by 10-column matrix (rows 1 through 5 and columns 1 through 10) on the network.

Note BLOCK ADD and CHECKSUM instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

Example



2. Inputs

Inputs to BLOCK ADD and CHECKSUM instructions can be connected to relay elements (except coils) and/or outputs from timers, counters, math instructions, data manipulation instructions, other instructions, etc.

3. Outputs

Outputs from BLOCK ADD and CHECKSUM instructions can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data manipulation instructions, etc.

System Status Monitoring Instruction

9

This chapter describes the instruction used to read system status.

9.1 System Status Monitoring Instruction

This section describes the function, structure, and operation of SYSTEM STATUS MONITORING and provides simple examples of its application. It also includes details on Building Programs and Precautions.

9.1.1 SYSTEM STATUS MONITORING (STAT)

1. Function

1) The data stored in the system status table in the CPU Module is read. The system status table is part of system memory.

2) The system status table contains data on the following Modules.

   a) CPU Module

   b) Communications Modules

   c) I/O Modules

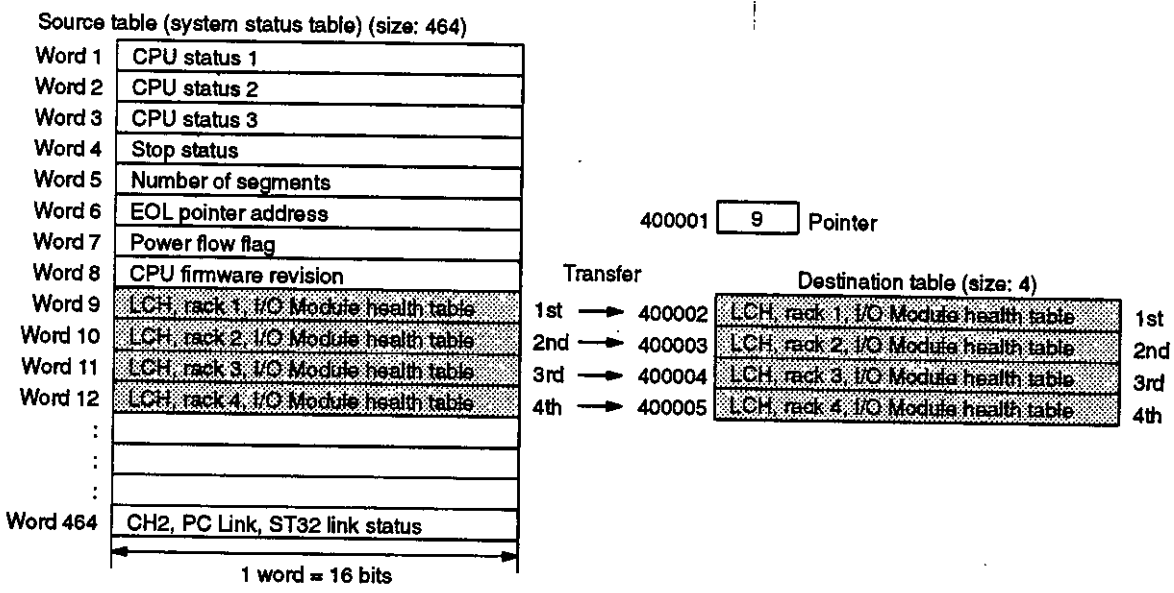
   d) Special-purpose Modules

   e) Motion Modules

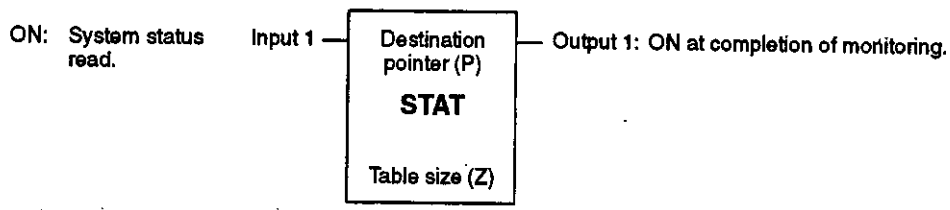
3) SYSTEM STATUS MONITORING can be thought of as a type of data transfer instructions, similar to those introduced earlier in this manual.

4) The value of the pointer and the size of the destination table can be set so that the status data from any particular block of the source table (system status table) can be transferred to the destination table. The transfer is completed in one scan.

Example:
When the pointer value and destination table shown in the following illustration are used and SYSTEM STATUS MONITORING is executed, the status data from words 9 to 12 of the system status table will be transferred to the destination table.



2. Structure



1) STAT is the symbol for SYSTEM STATUS MONITORING.

2) STAT requires two elements, one top element and one bottom element, located vertically on the network. Refer to *Table 9.1* for details on specifying constants or registers for these elements.

Example

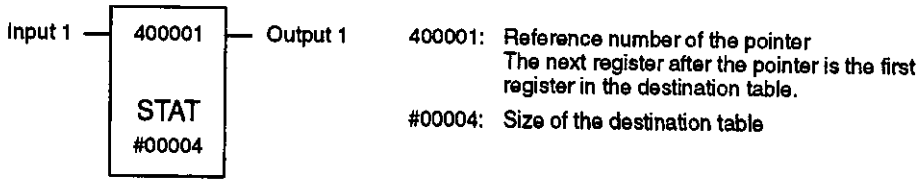
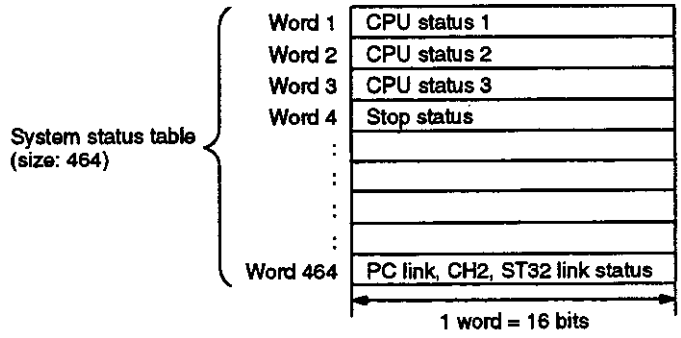


Table 9.1 Structural Elements of STAT

Element	Meaning	Possible Settings
Top (P)	• Reference number of the pointer • The next register after the pointer is the first register in the destination table.	Holding register: 400001 to 409998 (W00001 to W09998) Link register: D10001 to D11023, D20001 to D21023
Bottom (Z)	Size of the destination table	Constant: #00001 to #00256

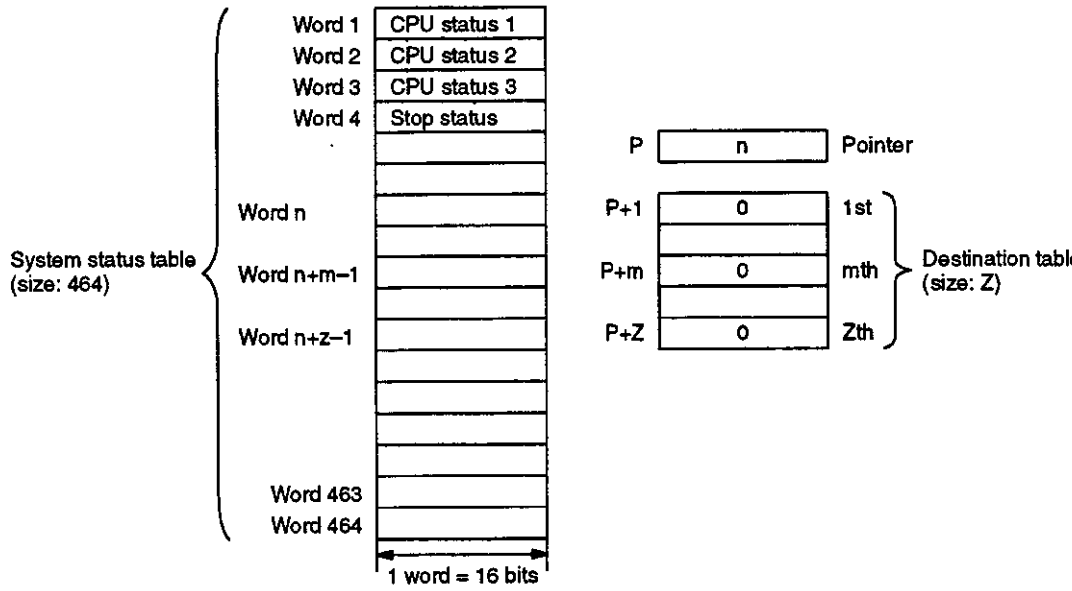
3) Source

The source for STAT is the system status table, which is part of the system memory. The system status table is a data table consisting of 464 words, as shown in the following illustration, and it contains data on the CPU Module, Communications Modules, I/O Modules, and Motion Modules. Each word has been allocated a word number between 1 and 464. Refer to *9.2 System Status Table* for details on the status information recorded in each word.

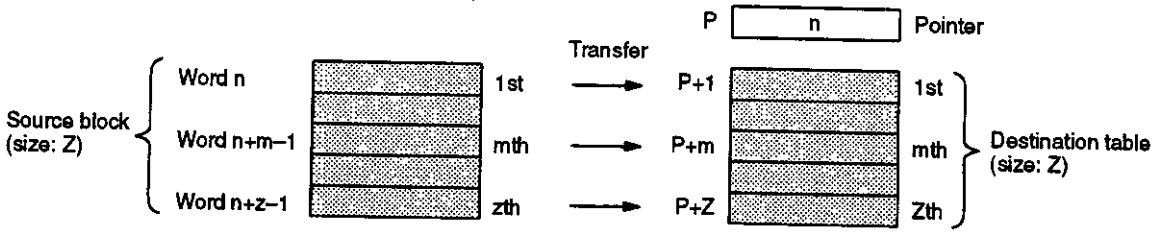


3. Operation

1) Status Before Execution



2) The following data transfer will be executed when input 1 turns ON and the pointer value (n) is within the valid range (1 to 464). The transfer will be completed in one scan.



a) Words n to n+Z–1 in the system status table are selected as the source block. For example, if n = 9, the source block will be word 9 to word 12.

b) The data in the selected source block is transferred to the destination table.

c) The pointer value and the data in the source block do not change.

d) Output 1 turns ON.

3) If the pointer value (n) is not within the valid range (0 to 465), the source block will not exist and nothing will be done even if input1 turns ON.

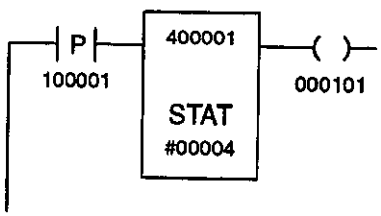
a) Data is not transferred, i.e., the system status is not read.

b) Output 1 remains OFF.

◀EXAMPLE▶

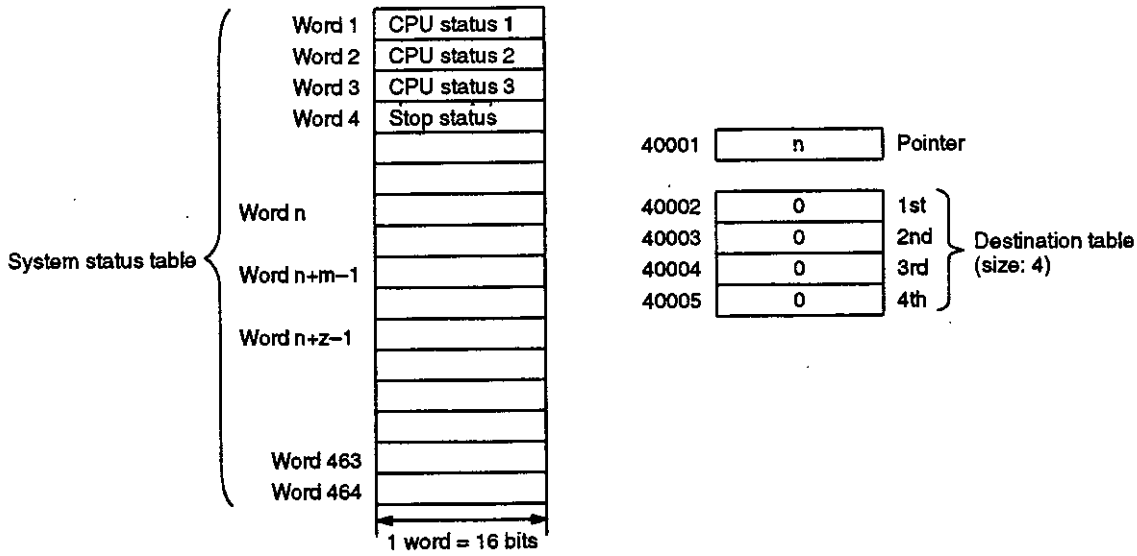
4. Application Example

1) Ladder Programming

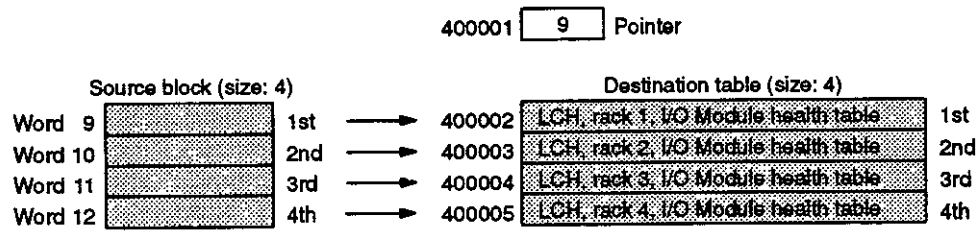


2) Operation

a) Before Execution



b) If the pointer value (n) is 9, the following data transfer will be performed when input relay 100001 changes from OFF to ON. The transfer will be completed in one scan.



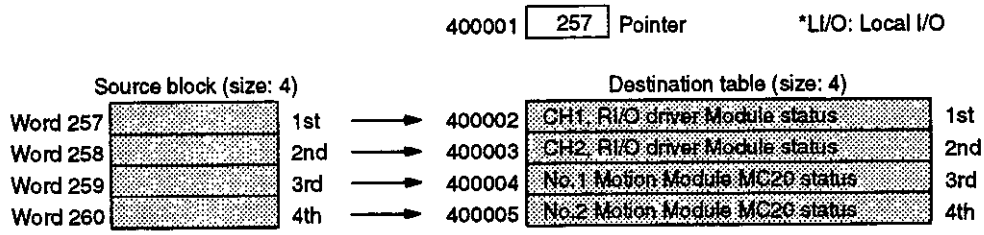
(1) A pointer value of 9 and a destination table size of 4 will cause word 9 to word 12 to be selected as the source block.

(2) The data in the selected source block will be transferred to the destination table.

(3) The pointer value and the data in the source block will not change.

(4) Coil 000101 will be ON only during the scan in which input relay 100001 changed from OFF to ON.

c) If the pointer value (n) is 257, the following data transfer will be performed when input relay 100001 changes from OFF to ON. The transfer will be completed in one scan.



(1) A pointer value of 257 and a destination table size of 4 will cause Word 257 to Word 260 to be selected as the source block.

(2) The data in the selected source block is transferred to the destination table.

(3) The pointer value and the data in the source block do not change.

(4) Coil 000101 turns ON only for the scan in which input relay 100001 changed from OFF to ON.

d) If the pointer value (n) is 0, then the pointer value will not be within the valid range (1 to 464) and data will not be transferred even when input relay 100001 changes from OFF to ON. Coil 000101 will remain OFF.

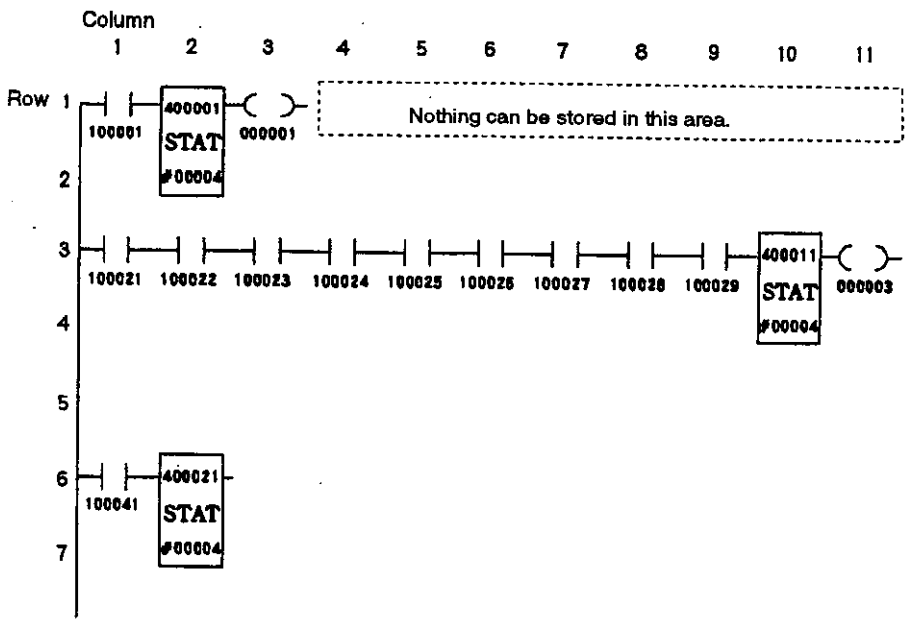
9.1.2 Building Programs

1. Storage Locations on Networks

Read Controller System Status instructions require two elements (top and bottom) located vertically on the network, so they can be stored anywhere on a 6-row by 10-column matrix (rows 1 through 6 and columns 1 through 10).

Note Read Controller System Status instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

Example



2. Inputs

Inputs to Read Controller System Status instructions can be connected to relay elements (except coils) and/or outputs from timers, counters, math instructions, data processing instructions, other instructions, etc.

3. Outputs

Outputs from Read Controller System Status instructions can be connected to any of the following: coils, contacts, inputs to math instructions, inputs to data processing instructions, etc.

9.2 System Status Table

This section describes the contents of the System Status Table.

9.2.1 Word No. and Items

The following table shows the relationship between the word numbers and the items to which they are assigned. Refer to the page number indicated in the table for details on each item.

Table 9.2 Word No. and Items

Word No.	Items	Page
1	CPU status 1	9-14
2	CPU status 2	9-14
3	CPU status 3	9-15
4	Stop status	9-15
5	Number of segments	9-15
6	EOL pointer address	9-16
7	Power flow flag	9-16
8	CPU firmware revision	9-16
9	I/O Module health table, local channel, rack 1	9-16
10	I/O Module health table, local channel, rack 2	9-16
11	I/O Module health table, local channel, rack 3	9-16
12	I/O Module health table, local channel, rack 4	9-16
13	I/O Module health table, remote channel 1, station 1, rack 1	9-16
14	I/O Module health table, remote channel 1, station 1, rack 2	9-16
15	I/O Module health table, remote channel 1, station 1, rack 3	9-16
16	I/O Module health table, remote channel 1, station 1, rack 4	9-16
17	I/O Module health table, remote channel 1, station 2, rack 1	9-16
–	—	
21	I/O Module health table, remote channel 1, station 3, rack 1	9-16
–	—	
25	I/O Module health table, remote channel 1, station 4, rack 1	9-16
–	—	
29	I/O Module health table, remote channel 1, station 5, rack 1	9-16
–	—	
33	I/O Module health table, remote channel 1, station 6, rack 1	9-16
–	—	
37	I/O Module health table, remote channel 1, station 7, rack 1	9-16
–	—	
41	I/O Module health table, remote channel 1, station 8, rack 1	9-16
–	—	

9.2.1 Word No. and Items cont.

Word No.	Items	Page
45	I/O Module health table, remote channel 1, station 9, rack 1	9-16
–	—	
49	I/O Module health table, remote channel 1, station 10, rack 1	9-16
–	—	
53	I/O Module health table, remote channel 1, station 11, rack 1	9-16
–	—	
57	I/O Module health table, remote channel 1, station 12, rack 1	9-16
–	—	
61	I/O Module health table, remote channel 1, station 13, rack 1	9-16
–	—	
65	I/O Module health table, remote channel 1, station 14, rack 1	9-16
–	—	
69	I/O Module health table, remote channel 1, station 15, rack 1	9-16
–	—	
73	I/O Module health table, remote channel 2, station 1, rack 1	9-16
–	—	
77	I/O Module health table, remote channel 2, station 2, rack 1	9-16
–	—	
81	I/O Module health table, remote channel 2, station 3, rack 1	9-16
–	—	
85	I/O Module health table, remote channel 2, station 4, rack 1	9-16
–	—	
89	I/O Module health table, remote channel 2, station 5, rack 1	9-16
–	—	
93	I/O Module health table, remote channel 2, station 6, rack 1	9-16
–	—	
97	I/O Module health table, remote channel 2, station 7, rack 1	9-16
–	—	
101	I/O Module health table, remote channel 2, station 8, rack 1	9-16
–	—	
105	I/O Module health table, remote channel 2, station 9, rack 1	9-16
–	—	
109	I/O Module health table, remote channel 2, station 10, rack 1	9-16
–	—	
113	I/O Module health table, remote channel 2, station 11, rack 1	9-16
–	—	
117	I/O Module health table, remote channel 2, station 12, rack 1	9-16
–	—	
121	I/O Module health table, remote channel 2, station 13, rack 1	9-16
–	—	
125	I/O Module health table, remote channel 2, station 14, rack 1	9-16
–	—	
129	I/O Module health table, remote channel 2, station 15, rack 1	9-16
–	—	
133	Status, remote channel 1, station 1	9-17
–	—	
147	Status, remote channel 1, station 15	

Word No.	Items	Page
148	Status, remote channel 2, station 1	9-17
–	——	
162	Status, remote channel 2, station 15	
163	I/O error counter (new error counter), local channel	9-17
164	I/O error counter (error counter), local channel	
165	I/O error counter (bus error counter), local channel	
166	I/O error counter (new error counter), remote channel 1, station 1	9-17
167	I/O error counter (error counter), remote channel 1, station 1	
168	I/O error counter (bus error counter), remote channel 1, station 1	
169	I/O error counter (new error counter), remote channel 1, station 2	9-17
–	——	
172	I/O error counter (new error counter), remote channel 1, station 3	9-17
–	——	
175	I/O error counter (new error counter), remote channel 1, station 4	9-17
–	——	
178	I/O error counter (new error counter), remote channel 1, station 5	9-17
–	——	
181	I/O error counter (new error counter), remote channel 1, station 6	9-17
–	——	
184	I/O error counter (new error counter), remote channel 1, station 7	9-17
–	——	
187	I/O error counter (new error counter), remote channel 1, station 8	9-17
–	——	
190	I/O error counter (new error counter), remote channel 1, station 9	9-17
–	——	
193	I/O error counter (new error counter), remote channel 1, station 10	9-17
–	——	
196	I/O error counter (new error counter), remote channel 1, station 11	9-17
–	——	
199	I/O error counter (new error counter), remote channel 1, station 12	9-17
–	——	
202	I/O error counter (new error counter), remote channel 1, station 13	9-17
–	——	
205	I/O error counter (new error counter), remote channel 1, station 14	9-17
–	——	
208	I/O error counter (new error counter), remote channel 1, station 15	9-17
–	——	
211	I/O error counter (new error counter), remote channel 2, station 1	9-17
–	——	
214	I/O error counter (new error counter), remote channel 2, station 2	9-17
–	——	
217	I/O error counter (new error counter), remote channel 2, station 3	9-17
–	——	
220	I/O error counter (new error counter), remote channel 2, station 4	9-17
–	——	
223	I/O error counter (new error counter), remote channel 2, station 5	9-17
–	——	

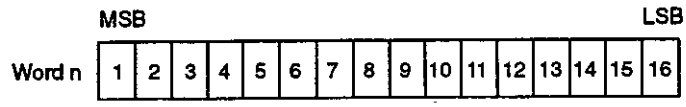
Word No.	Items	Page
226	I/O error counter (new error counter), remote channel 2, station 6	9-17
–	——	
229	I/O error counter (new error counter), remote channel 2, station 7	9-17
–	——	
232	I/O error counter (new error counter), remote channel 2, station 8	9-17
–	——	
235	I/O error counter (new error counter), remote channel 2, station 9	9-17
–	——	
238	I/O error counter (new error counter), remote channel 2, station 10	9-17
–	——	
241	I/O error counter (new error counter), remote channel 2, station 11	9-17
–	——	
244	I/O error counter (new error counter), remote channel 2, station 12	9-17
–	——	
247	I/O error counter (new error counter), remote channel 2, station 13	9-17
–	——	
250	I/O error counter (new error counter), remote channel 2, station 14	9-17
–	——	
253	I/O error counter (new error counter), remote channel 2, station 15	9-17
–	——	
256	Option Module installation status	9-19
257	Channel 1 Remote I/O Driver Module status	9-19
258	Channel 2 Remote I/O Driver Module status	
259	No.1 MC20 Module status	9-20
260	No.2 MC20 Module status	
261	For future expansion	
262	For future expansion	
263	Channel 1 PC Link Module status	9-20
264	Channel 2 PC Link Module status	
265	For future expansion	
266	For future expansion	
267	No. 1 MEMOBUS Module status	9-20
268	No. 2 MEMOBUS Module status	
269	For future expansion	
–	——	
274	For future expansion	
275	Channel 1 Remote I/O Driver Module revision	9-21
276	Channel 2 Remote I/O Driver Module revision	
277	No. 1 MC20 Module main revision	9-21
278	No. 1 MC20 Module servo revision	
279	No. 2 MC20 Module main revision	
280	No. 2 MC20 Module servo revision	
281	For future expansion	
–	——	
284	For future expansion	
285	Channel 1 PC Link Module revision	9-21
286	Channel 2 PC Link Module revision	

Word No.	Items	Page
287	For future expansion	
288	For future expansion	
289	No. 1 MEMOBUS Module revision	9-21
290	No. 2 MEMOBUS Module revision	
291	For future expansion	
–	—	
296	For future expansion	
297	I/O Module status history, previous location, local channel	9-21
298	I/O Module status history, previous status, local channel	
–	—	
335	I/O Module status history, 20th previous location, local channel	
336	I/O Module status history, 20th previous status, local channel	
337	Stop status history, current stop status	9-22
338	Stop status history, current error code	
339	Stop status history, previous stop status	
340	Stop status history, previous error code	
–	—	
375	Stop status history, 19th previous stop status	
376	Stop status history, 19th previous error code	
377	No. 1 MC20 Module error status history, current error status	9-22
378	No. 1 MC20 Module error status history, previous error status	
–	—	
386	No. 1 MC20 Module error status history, 9th previous error status	
387	No. 2 MC20 Module error status history, current error status	
388	No. 2 MC20 Module error status history, previous error status	
–	—	
396	No. 2 MC20 Module error status history, 9th previous error status	
397	Channel 1 PC Link bit status for stations 1 to 16	9-23
398	Channel 1 PC Link bit status for stations 17 to 32	
399	Channel 2 PC Link bit status for stations 1 to 16	
400	Channel 2 PC Link bit status for stations 17 to 32	
401	Channel 1 PC Link station 1, link status	9-24
–	—	
432	Channel 1 PC Link station 32, link status	
433	Channel 2 PC Link station 1, link status	9-24
–	—	
464	Channel 2 PC Link station 32, link status	

9.2.2 System Status Table Details

1) Bit No.

Each word in the System Status Table has 16 bits and each bit has a number allocated to it as indicated in the following table below.



2) Word 1: CPU Status 1

Word 1 stores the following data on the CPU Module and the Power Supply Module.

Table 9.3 Word 1

Bit No.	Name	Meaning	
		Bit at "1"	Bit at "0"
1	High-speed scan, single sweep mode	CPU is set for a high-speed scan in single sweep mode.	CPU is not set for a high-speed scan in single sweep mode.
6	Constant sweep mode	CPU is set for constant sweep mode.	CPU not is set for constant sweep mode.
7	Normal scan, single sweep mode	CPU is set for a normal scan in single sweep mode.	CPU is not set for a normal scan in single sweep mode.
9	Power Supply Module normal	The Power Supply Module is operating normally.	The Power Supply Module is not operating normally.
10	RUN indicator	The RUN indicator is not lit, i.e., the CPU Module is stopped.	The RUN indicator is lit, i.e., the CPU Module is running.
11	Memory protection	Memory is not protected.	Memory is protected.
12	Battery output voltage error	The voltage output from the battery is low; replace the battery.	The voltage output from the battery is normal.

3) Word 2: CPU Status 2

Word 2 stores the following data on the CPU Module.

Table 9.4 Word 2

Bit No.	Name	Bit at "1"
1	First normal scan	The first normal scan is being performed since operation started.
3	Constant sweep not possible	The scan time is longer than the time set for the constant sweep.
4	System configulation load required	Operation is not possible unless the system configuration is loaded into the CPU Module.

4) Word 3: CPU Status 3

Word 3 stores the following data on the CPU Module.

Table 9.5 Word 3

Bit No.	Name	Bit at "1"
1	First high-speed scan	The first high-speed scan is being performed since operation started.

5) Word 4: Stop Status

Word 4 stores the cause of operational stops for the CPU Module.

Table 9.6 Word 4

Bit No.	Name	Bit at "1"
1	Controller stop	The CPU Module has stopped operation due to Programming Panel operation.
2	Allocation memory error	Invalid data has been found in allocation memory.
3	Load required	The system configuration table needs to be loaded.
5	Segment scheduler error	There is an error in the contents of the segment scheduler table.
7	Power down checksum error	• There is a difference between the status of the system configuration memory when power was interrupted and after power was resupplied. • There is an error in the calendar circuit. • A hardware error has been discovered.
8	EOL error	There is an error in the data at the last address in the ladder program memory.
9	Watchdog timer error	The normal scan time has exceeded 240 ms.
12	Remote I/O Driver error	There is an error in the Remote I/O Driver Module.
13	Node type error	There is invalid data in the ladder program memory.
14	Ladder program checksum error	The ladder program memory has been changed by means other than the Programming Panel.
16	System configuration error	There is invalid data in the system configuration memory.

Note The power must be cycled when a hardware error has been discovered. The contents of the stop status will be A001 (hex).

6) Word 5: Number of Segments

Word 5 contains the total number of high-speed, normal, and subroutine segments between 1 and 32.

Word 5 | 1 to 32 | **Decimal**

7) Word 6: EOL Pointer Address

Word 6 contains the address in memory that contains the last address in the ladder program. This word always contains 7FEE (hex).

Word 6 | 7FEE | **Hexadecimal**

8) Word 7: Power Flow Flag

Word 7 contains data on the current status display mode when monitoring the program on the Programming Panel or other devices.

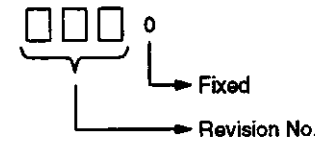
Word 7 | 0 to 2 | **Decimal**

0 = No display
1 = State flow display
2 = Power flow display

9) Word 8: CPU Firmware Revision

Word 8 contains the revision number of the CPU firmware.

Word 8 | 0000 to FFF0 | **Hexadecimal**



10) Words 9 to 132: I/O Module Health Tables

a) Words 9 to 132 are each allocated to a slot on the Mounting Base and contain health tables that indicate the status (normal/error) of the following Modules.

- Digital I/O Modules
- Analog I/O Modules
- Special-purpose Modules
- One-axis Motion Modules

b) Each word stores the health status of one rack (16 slots) of Modules. Refer to the following table for details.

Table 9.7 Words 9 to 132

Bit No.	Name	Meaning
1	Health status of I/O Module in slot 1	1) "1" indicates that the Module is operating normally. 2) "0" indicates that either an error has occurred in the Module or that an I/O Module is not mounted.
2	Health status of I/O Module in slot 2	
3	Health status of I/O Module in slot 3	
4	Health status of I/O Module in slot 4	
5	Health status of I/O Module in slot 5	
6	Health status of I/O Module in slot 6	
7	Health status of I/O Module in slot 7	
8	Health status of I/O Module in slot 8	
9	Health status of I/O Module in slot 9	
10	Health status of I/O Module in slot 10	
11	Health status of I/O Module in slot 11	
12	Health status of I/O Module in slot 12	
13	Health status of I/O Module in slot 13	
14	Health status of I/O Module in slot 14	
15	Health status of I/O Module in slot 15	
16	Health status of I/O Module in slot 16	

11) Words 133 to 162: Remote Station Status Tables

Words 133 to 162 contain the status of the stations on the remote channels.

Table 9.8 Words 133 to 162

Bit No.	Name	Bit at "1"
1	Transmission error or no response from remote station	An error occurred between the Remote I/O Driver Module and the Remote I/O Receiver Module or the remote station does not respond.
2	Allocation data error	There is invalid data in the allocation data.
3	Output data length error	There is an error in the output data length from the Remote I/O Driver Module.
5	MEMOBUS port transmission parameter error	There is an error in the transmission parameters for the MEMOBUS port on the Remote I/O Receiver Module.
8	ASIC error	There is an error in the ASIC for the bus controller.
9	Allocation required	There is no allocation data at the remote station.
13	I/O service timeout error	I/O servicing was not finished in the allotted time.

12) Words 163 to 255: I/O Error Counters

a) Status words 163 to 255 contain the numbers of errors that have occurred by station for the following Modules mounted on local and remote stations.

- Digital I/O Modules
- Analog I/O Modules
- Special-purpose Modules
- One-axis Motion Modules

b) There are three types of I/O Error Counters, as detailed below. All three are cleared to zero during the power-up sequence of the CPU Module Operation Start (RUN).

c) New Error Counter

Example: Word 163 contains the number of new errors that have occurred in local channel modules.

	Upper byte	Lower byte
Word 163	80 or 0 (hex)	0 to 255 (decimal)

Upper byte data on the New Error Counter have the following meanings.

80 (hexadecimal) . . I/O processing for all Modules on the local channel is normal.

00 (hexadecimal) . . I/O processing for a local channel Module is not normal.

Lower byte data on the New Error Counter has the following meaning.

The byte data will increase by one each time I/O processing for any Module on the local channel changes from normal to abnormal. When an error occurs in I/O processing for the Module and the byte data is 255, the byte data will return to 0.

d) Error Counter

Example: Word 164 contains the number of I/O errors that have occurred in the local channel.

Word 164	0 to 65, 535	(decimal)

The counter increases by one when an I/O processing error is detected for a Module on the local channel. When an error occurs in I/O processing for the Module and the byte data is 65,535, the data will return to 0.

e) Bus Error Counter

Example: Word 165 contains the number of bus errors that have occurred in the local channel.

Word 165	0 to 65, 535	(decimal)

Data increases by one each time a bus error is detected in a Module on a local channel. When a bus error occurs and the byte data is 65,535, the data will return to 0.

13) Word 256: Option Module Installation Status

Word 256 contains the installation status of Option Modules.

Table 9.9 Word 256

Bit No.	Bit at "1"
3	Channel 2 Remote I/O Driver Module is mounted.
4	Channel 1 Remote I/O Driver Module is mounted.
5	No. 2 MEMOBUS Module is mounted.
6	No. 1 MEMOBUS Module is mounted.
11	Channel 2 PC Link Module is mounted.
12	Channel 1 PC Link Module is mounted.
15	No. 2 MC20 Module is mounted.
16	No. 1 MC20 Module is mounted.

14) Words 257 to 258: Remote I/O Driver Module Status

Words 257 to 258 contain the error status of the Remote I/O Driver Modules.

Table 9.10 Words 257 to 258

Bit No.	Bit at "1"
2	Watchdog timer error has occurred.
13	Common memory check error has occurred.
14	RAM check error has occurred.
15	ROM total checksum error has occurred.
16	Not ready

15) Words 259 to 260: MC20 Module Status

Words 259 to 260 contain the error status of the Four-axis Motion Modules (MC20).

Table 9.11 Words 259 and 260

Bit No.	Name	Bit at "1"	Remarks
1	ASIC error	Error indicated on the left has occurred.	Main firmware error status
2	ROM total checksum error		
3	Common memory check error		
4	RAM check error		
5	For future expansion		
6	Power error		
7	System error		
8	Division by 0 or overflow error		
9	EEPROM error		Servo firmware error status
10	ROM total checksum error		
11	Common memory check error		
12	RAM check error		
13	Watchdog timer error		
14	Power error		
15	System error		
16	Division by 0 or overflow error		

16) Words 263 to 264: PC Link Module Status

Words 263 to 264 contain the error status of the PC Link Modules.

Table 9.12 Words 263 to 264

Bit No.	Bit at "1"
2	Watchdog timer error has occurred.
13	Common memory check error has occurred.
14	RAM check error has occurred.
15	ROM total checksum error has occurred.
16	Not ready

17) Words 267 to 268: MEMOBUS Module Status

Words 267 to 268 contain the error status of the MEMOBUS Modules.

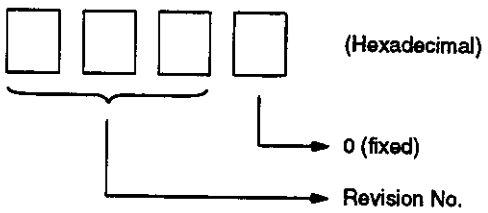
Table 9.13 Words 267 to 268

Bit No.	Bit at "1"
2	Watchdog timer error has occurred.
13	Common memory check error has occurred.
14	RAM check error has occurred.
15	ROM total checksum error has occurred.
16	Not ready

18) Status Words 275 to 290: Option Module Revisions

Words 275 to 290 contain the firmware revision numbers for the following Modules. The revision numbers are separated into main numbers and servo numbers.

- Remote I/O Driver Modules
- Four-axis Motion Modules MC20
- PC Link Modules
- MEMOBUS Modules



19) Words 297 to 336: I/O Module Status History, Local Channel

a) Words 297 to 336 contain the most recent 20 error status histories for the following Modules mounted to the local channel.

- Digital I/O Modules
- Analog I/O Modules
- Special-purpose Modules
- One-axis Motion Modules

b) Example: Word 297 and Word 298 contain the previous error status history.

	Upper byte	Lower byte
Word 297	1 to 4 (decimal)	1 to 16 (decimal)

Upper byte data . . Displays the rack number of the Module where the error occurred.

Lower byte data . . Displays the slot number of the Module where the error occurred.

Word 298	0, 2, 3, 4, 6, 7, FE, FF (hexadecimal)

0 .... A Module that has not been allocated I/O has been installed.

2 .... I/O has been allocated, but a Module has not been installed, or the Module to which I/O has been allocated has been installed, but the configuration for it has not been completed.

3 .... A Module to which I/O has been allocated has been installed and the Module configuration has been completed, but the parameters have not yet been set.

4 .... A Module to which I/O has been allocated has been installed and the Module configuration has been completed, but parameters are currently being set.

6 .... Preparations to begin normal operation from the next scan have been completed.

7 .... A bus error occurred during I/O servicing.

FE ... There is no answer from the Module about parameter settings.

FF ... The Module could not be recognized.

20) Words 337 to 376: Stop Status Histories

Words 337 to 376 contain the status of the CPU Module when it stops (excluding 8000 (hex) generated when the CPU Module is force-stopped from the Programming Panel) and the error code for the most recent 19 stops. The first two words provide the current stop status. The error codes indicate the cause of the stop. The histories are updated when operation is started from a stopped status. Error codes are stored with the stop status as shown in the following table.

Table 9.14 Words 337 to 376

Bit No.	Bit at "1"	Error Code (Decimal)
2	Allocation memory error	1 and higher
7	Power down checksum error	3001 and higher (not for hardware errors)
12	Remote I/O Driver Module error	4001 and higher for channel 1 Remote I/O Driver Module 5001 and higher for channel 2 Remote I/O Driver Module
13	Node type error	2001 and higher
14	ladder program checksum error	1001 and higher
16	System configuration error	1 and higher

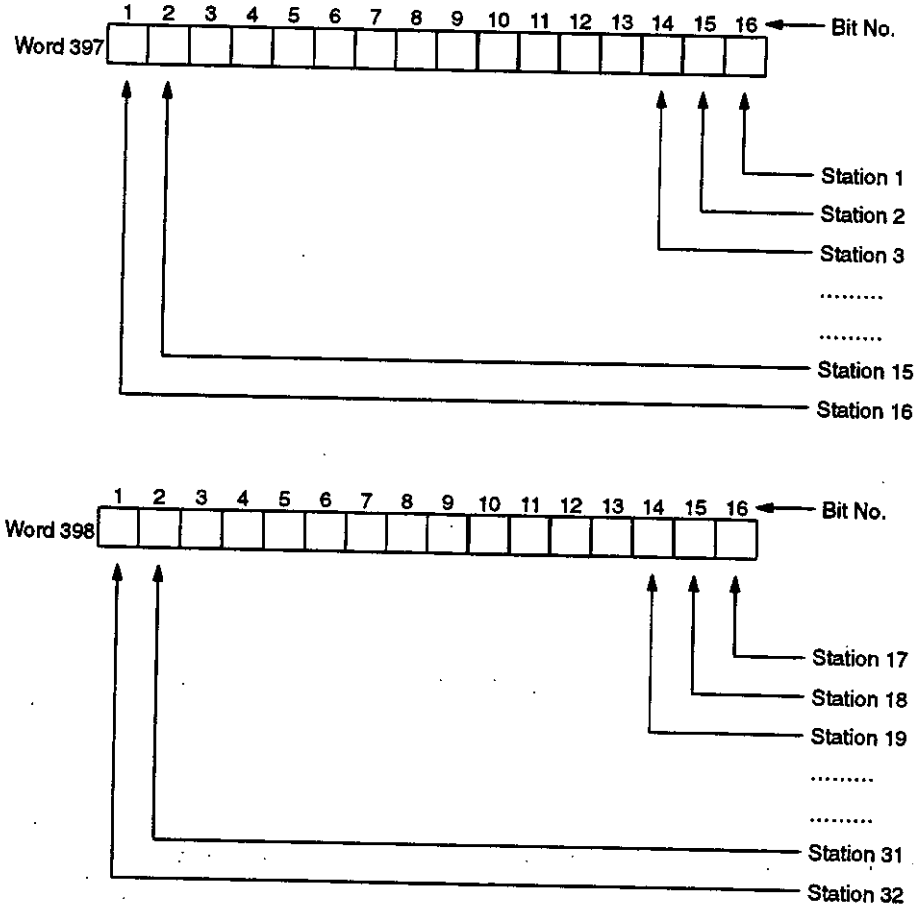
21) Words 377 to 396: MC20 Module Error Status History

Words 377 to 396 contain the most recent 9 error status history records (except for 0 for normal) for the Four-axis Motion Modules (MC20). The history is updated whenever the error status changes. The first two words contain the current error status.

22) Words 397 to 400: PC Link Module Bit Status

a) Words 397 to 400 contain the PC Link Module Status.

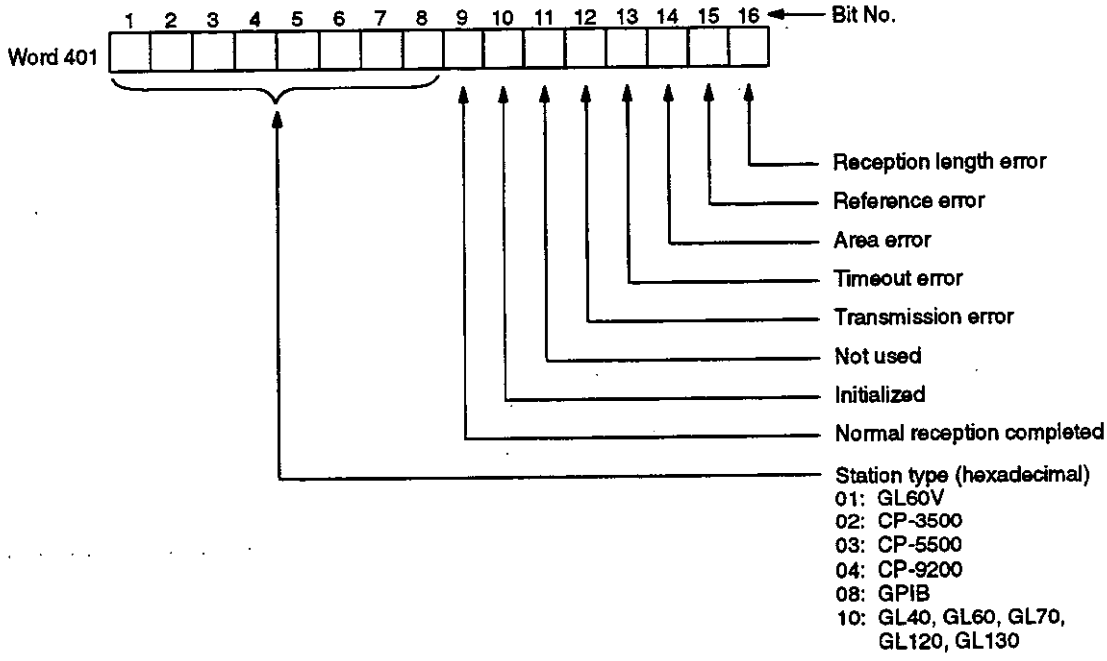
b) Example: The PC Link Module Status for the 32 Modules on channel 1 is stored as follows in words 397 and 398 ("1" is normal and "0" is abnormal):



23) Words 401 to 464: PC Link Module Link Status

a) Words 401 to 464 contain the link transmission status for the PC Link Modules (one word per station).

b) Example: The link transmission status for the PC Link Module at station 1 of channel 1 is stored as follows in word 401:



Reception Length Error
Set when the number of bytes for all link registers and link coils allocated to a single station exceeds 512 bytes.

Reference Error
Set when the total number of references of link registers and link coils exceeds 2,048.

Area Error
Set when the number of link registers allocated to a single station exceeds 256 words or when the number of link coils exceeds 4,096 points.

Transmission Error
The transmission error bit is set to "1" whenever a reception length, reference, area, or time out error occurs.

Sequence Control Instructions 10

This chapter describes the instructions used to control sequences.

10.1 Sequencers

This section describes the function, structure, and operation of sequencers and provides simple examples of their application.

10.1.1 Sequencers

1. Function

1) Sequencer Numbers

Thirty-two stepping sequencers are provided. The sequencers are identified by sequencer numbers running from 1 to 32, e.g., sequencer 1, sequencer 2, etc.

2) Step Numbers

Each stepping sequencer contains a maximum of 99 steps. The steps are identified by sequencer numbers running from 1 to 99, e.g., step 1, step 2, etc.

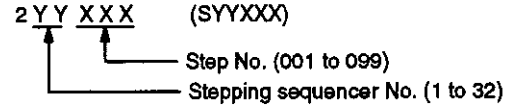
3) Sequencer Control Registers

Stepping sequencer 1 is controlled by the content of holding register 402001; sequencer 2, by the content of holding register 402032, etc. These registers are called stepping sequencer control registers. The contents of these registers can be manipulated with timer, counter, arithmetic, or other instructions to advance the stepping sequencers, to implement returns, jumps, etc.

2. Structure

1) Reference Numbers

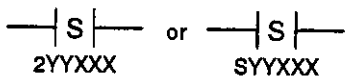
Reference numbers are allocated to stepping sequencers as shown below. For example, 201032 is the reference number allocated to step 32 of sequencer 1. The reference numbers in parentheses are used for alphanumeric numbers.



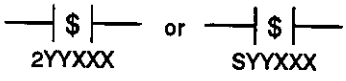
2) Contacts

Stepping sequencers are used in ladder programming through N.O. and N.C. contacts. Transitional contacts cannot be used. The structure of the N.O. and N.C. contacts for stepping sequencers is shown below.

a) N.O. Contacts for Stepping Sequencers



b) N.C. Contacts for Stepping Sequencers



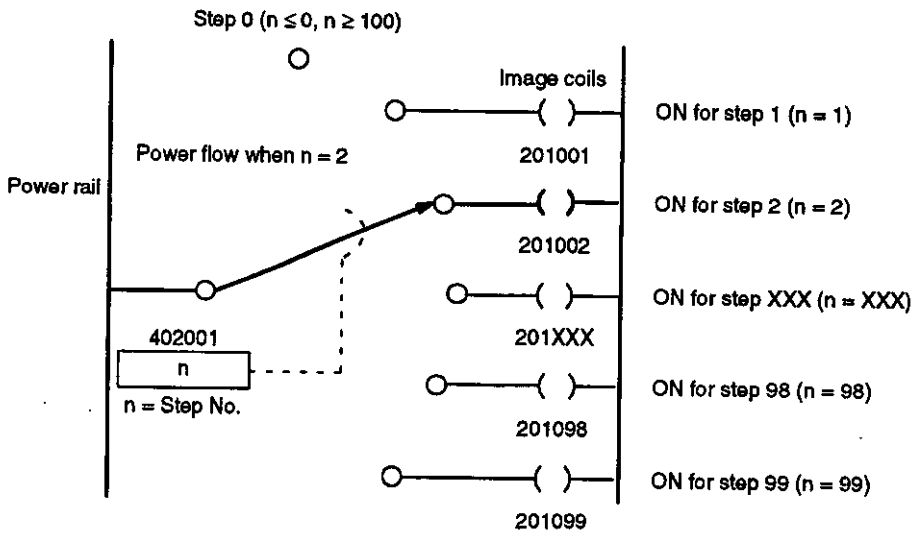
3. Operation

1) Stepping sequencers is described using sequencer 1 as an example. Stepping sequencer 1 operates as described below based on the contents (n) of holding register 402001.

   a) When n is between 1 and 99, one of the image coils between 201001 and 201099 will be ON depending on the value of n. N.O. contacts will be energized and N.C. coils will be de-energized. For example, if n = 1, image coil 201001 will be turned ON, N.O. contacts for 201001 will be energized and N.C. coils for it will be de-energized.

   b) When n is not between 1 and 99, all image coils between 201001 and 201099 will be OFF.

2) The equivalent sequencer circuit for stepping sequencer 1 is shown in the following illustration.



3) The following table outlines the operation of the 32 stepping sequencers

Table 10.1 Operation of Stepping Sequencer

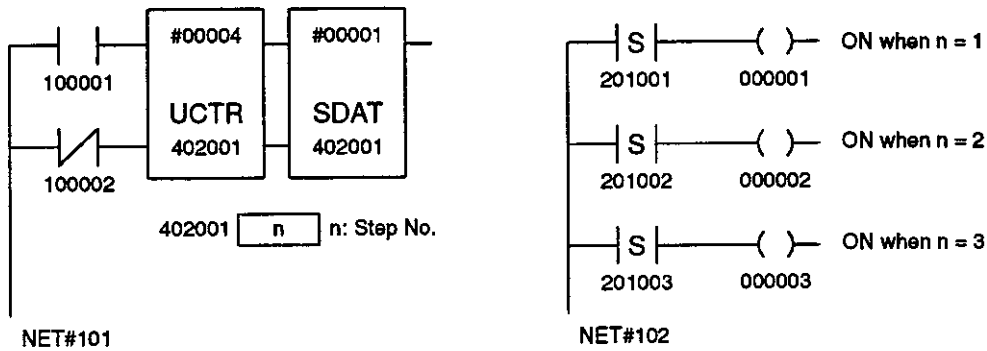
Stepping Sequencer No.	Control Register Reference No. (See Note.)	Contents of Stepping Sequencer Control Register				
		1	2	3	—	99
1	402001	Only 201001 ON	Only 201002 ON	Only 201003 ON	—	Only 201099 ON
2	402002	Only 202001 ON	Only 202002 ON	Only 202003 ON	—	Only 202099 ON
3	402003	Only 203001 ON	Only 203002 ON	Only 203003 ON	—	Only 203099 ON
–	—	—	—	—	—	—
32	402032	Only 232001 ON	Only 232002 ON	Only 232003 ON	—	Only 232099 ON

Note Holding registers 402001 to 402032 can be used as normal holding registers when they are not being used to control stepping sequencers.

10.1.2 Stepping Sequencer Application Example

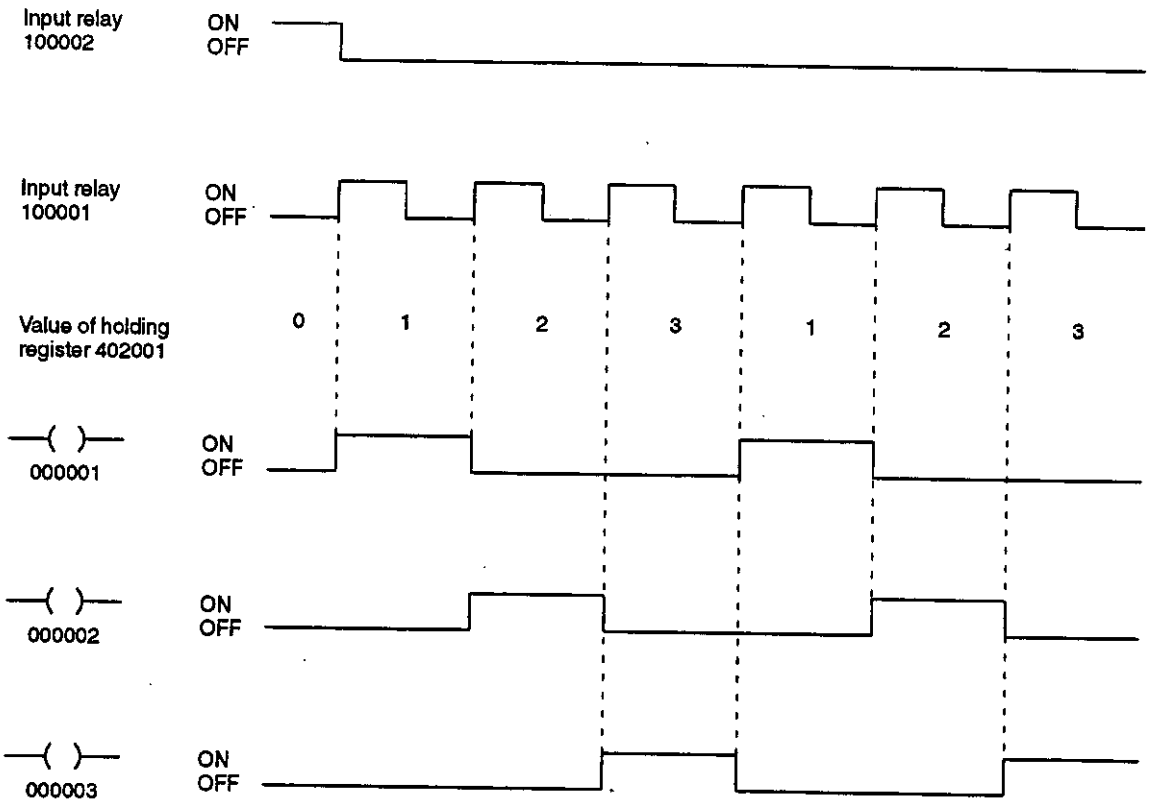
◀EXAMPLE▶ **Example 1: Controlling a Stepping Sequencer via an Up Counter**

1) Ladder Programming



2) Operation

a) Coils 000001 to 000003 operates as shown below.



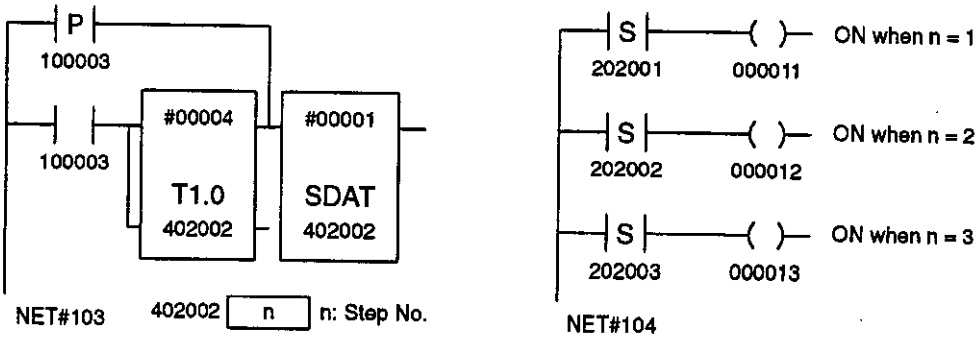
b) UCTR (UP COUNTER) and SDAT (16-BIT DATA SET) are used for the following purposes.

(1) UCTR is used to increment the step number one at a time.

(2) SDAT is used to force-set n to 1 in the scan when n becomes 4.

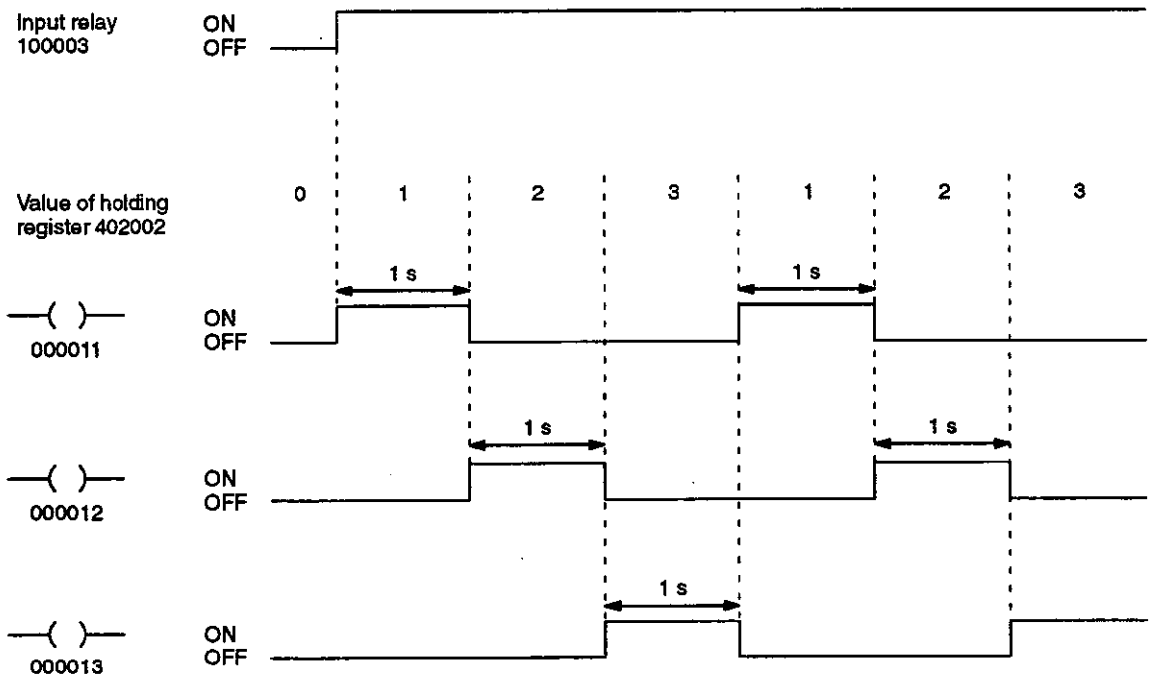
◀EXAMPLE▶ **Example 2: Controlling a Stepping Sequencer via a Timer**

1) Ladder Programming



2) Operation

a) Coils 000011 to 000013 operates as shown below.



b) T1.0 (1-SECOND TIMER) and SDAT (16-BIT DATA SET) are used for the following purposes.

(1) T1.0 is used to increment the step number one at a time.

(2) SDAT is used to force-set n to 1 in the scan when n becomes 4.

Program Control Instructions

11

This chapter describes the instructions used to control program execution.

11.1 Skip Node Instructions

This section describes the functions, structures, and operation of skip node instructions and provides simple examples of their application.

11.1.1 Skip Node Instructions

1. Function

1) The skip node instructions are used to freeze solving the program of a specified number of networks by skipping over them. None of the ladder programming in the skipped networks will be solved.

2) The skip node instructions can be used to reduce the scan time, to give solving specific networks higher priority, or to create subroutines in the high-speed and normal segments.

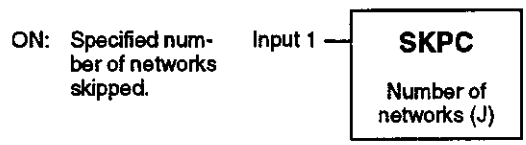
3) There are two skip node instructions. The function and operation of these instructions are the same.

   a) **SKIP CONSTANT (SKPC):** The number of networks to be skipped is specified as a constant between 0 and 9,999.

   b) **SKIP REGISTER (SKPR):** The number of networks to be skipped is specified as the contents of a register between 0 and 65,535.

2. Structure

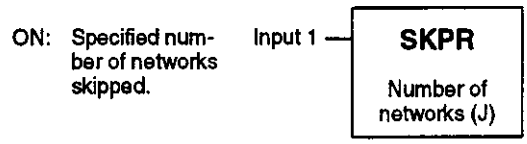
1) SKIP CONSTANT



1) SKPC is the symbol for SKIP CONSTANT.

2) SKPC requires one element (bottom) on the network. Specify a constant between #00000 and #09999 to specify the number of networks to be skipped.

3) SKIP REGISTER



1) SKPR is the symbol for SKIP REGISTER.

2) SKPR requires one element (bottom) on the network. Refer to *Table 11.1* to specify the reference number of a register. The contents of the specified register is the number of networks to be skipped.

Table 11.1 Structural Elements of SKPR

Element	Meaning	Possible Settings	
J	The contents of the specified register provide the number of networks (J) to be skipped. J can be between 0 and 65,535.	Input register:	300001 to 300512 (Z00001 to Z00512)
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 R20001 to R21024

3. Operation

The skip node instructions is described using SKPC as an example. Refer to the illustration on the following page. SKPR operates in the same way as SKPC.
The following processing will be executed while the input to SKPC is ON.

1) If SKPC is in network N of segment L and the constant specified for SKPC is J, the following consecutive networks will be skipped in segment L: N, N+1, N+2, ..., N+J–1.

2) All networks in segment L from network N on is skipped for either of the following conditions.

   a) If J is set to 0.

   b) If J is larger than the number of networks in segment L from network N on (i.e., larger than Z+N–1).

3) The effective range of SKPC is limited to segment L, i.e., networks in the segment following segment L (i.e., segment L+1) cannot be skipped.

4) The ladder programming in the networks that have been skipped is processed as follows:

   a) All of the ladder programming in network N (the network containing SKPC) after SKPC is frozen.

   b) All ladder programming in skipped networks from N+1 on is frozen.

5) The ON/OFF status of coils and the contents of registers used in the ladder programming that is frozen will remain unchanged, i.e., they will maintain the same status/contents as

they had just before being skipped. The status/contents of these coils/registers can still be changed from execution of ladder programming in networks that are not skipped.

6) The following illustration outlines the operation of SKPC.

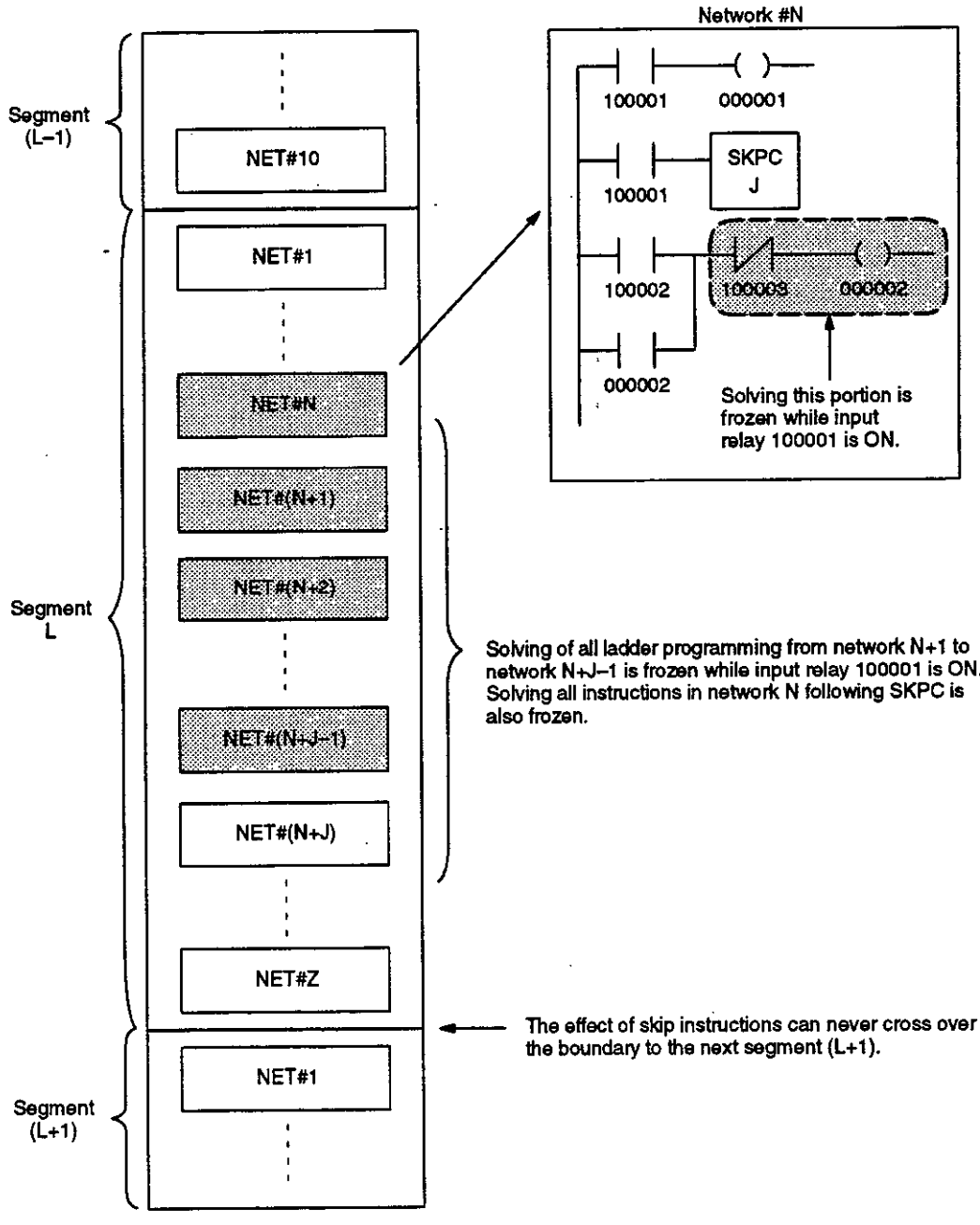


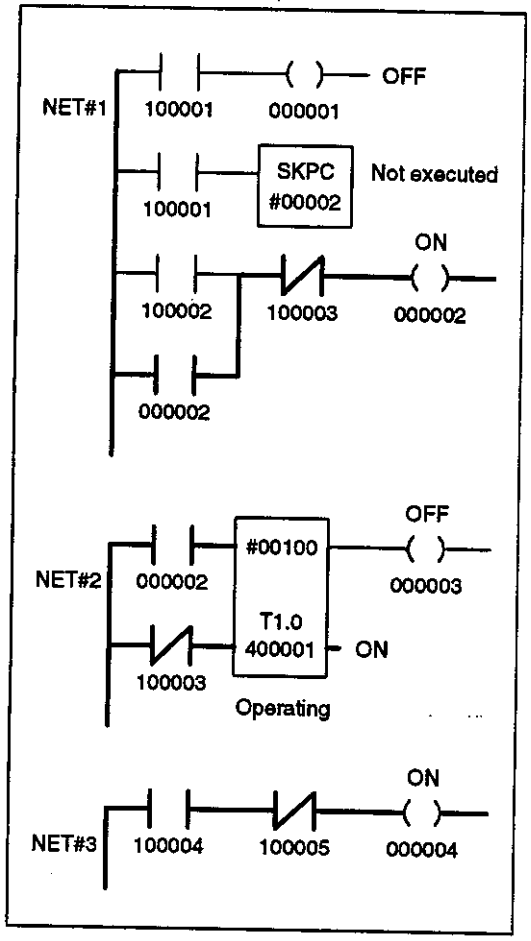
Figure 11.1 Operation of SKPC

4. Application Examples

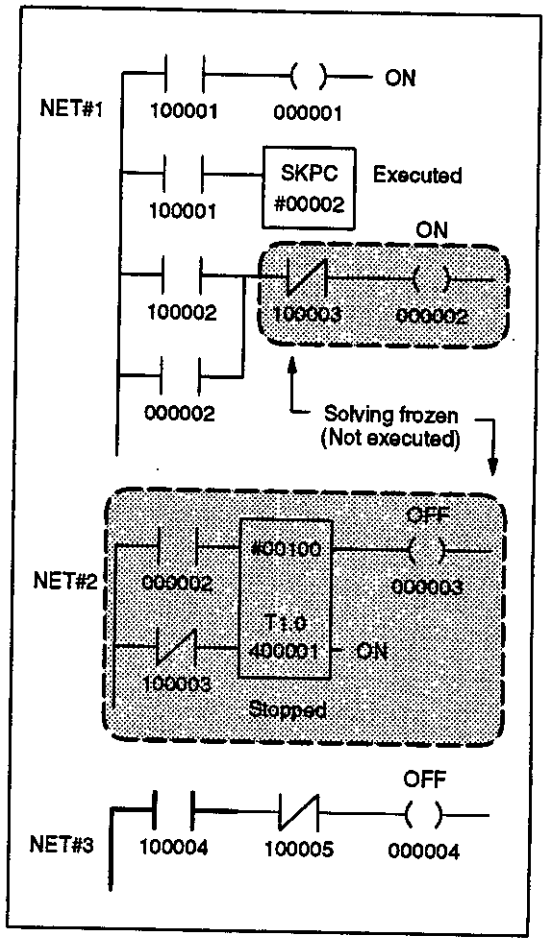
◀EXAMPLE▶ **Example 1**

1) Ladder Programming

a) Input Relay 100001 is OFF



b) Input Relay 100001 is ON



2) Operation

a) All of the ladder programming is solved normally as long as input relay 100001 is OFF.

b) Solving of the ladder programming will change as follows when input relay 100001 turns ON:

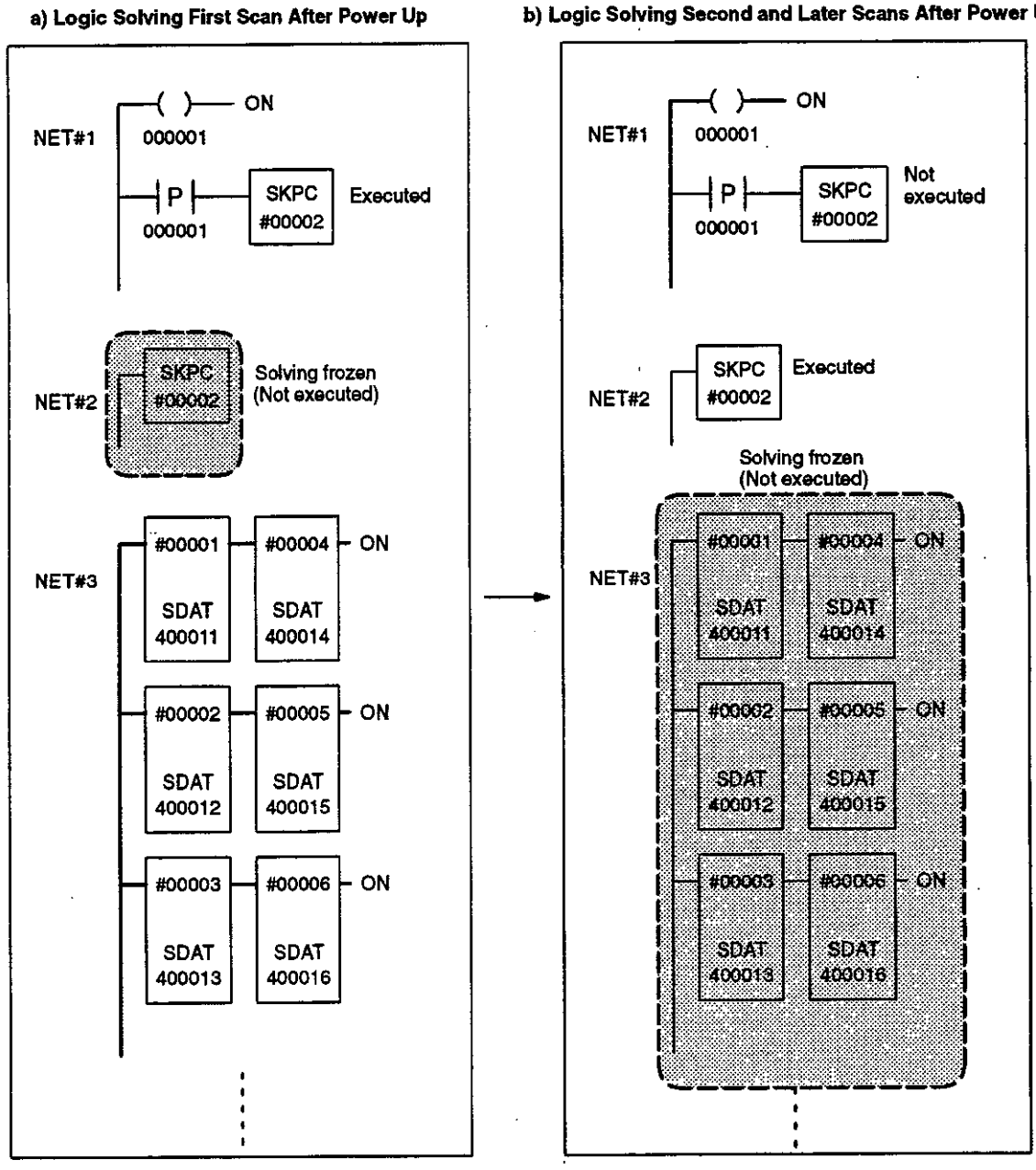
(1) Solving of the ladder programming shown in shaded areas is frozen, i.e., coil 000002 remains ON and coil 000003 remains OFF, and the contents of holding register 400001 is not incremented after the scan input relay 100001 changes from OFF to ON.

(2) All of the ladder programming not in shaded areas is solved normally, i.e., coil 000004 will turn OFF when input relay 100005 turns ON.

◀EXAMPLE▶ **Example 2**

In this example, the programming to set data is executed only in the first scan after power is turned ON.

1) Ladder Programming



2) Operation

a) Network 3 is solved in the first scan after power is turned ON to set constants 1 to 6 in holding registers 400011 to 400016.

b) Execution of network 3 is frozen in all other scans after the first scan so that data is not set again.

11.2 Subroutine Instructions

This section describes the functions, structures, and operation of subroutine instructions and provides simple examples of their application.

11.2.1 Subroutines

1. Subroutines

1) Up to 1,023 subroutines can be stored in the subroutine segment.

2) Subroutines are identified by subroutine numbers from 1 to 1,023, e.g., subroutine 1, subroutine 2, etc.

3) Each subroutine is made up of ladder programming constituting one or more networks. Each subroutine can consist of as many networks as required as long as user program memory capacity is not exceeded.

4) Each subroutine starts with the SUBROUTINE LABEL (LAB) instruction and ends with the SUBROUTINE RETURN (RET) instruction.

5) The following illustration shows two subroutines.

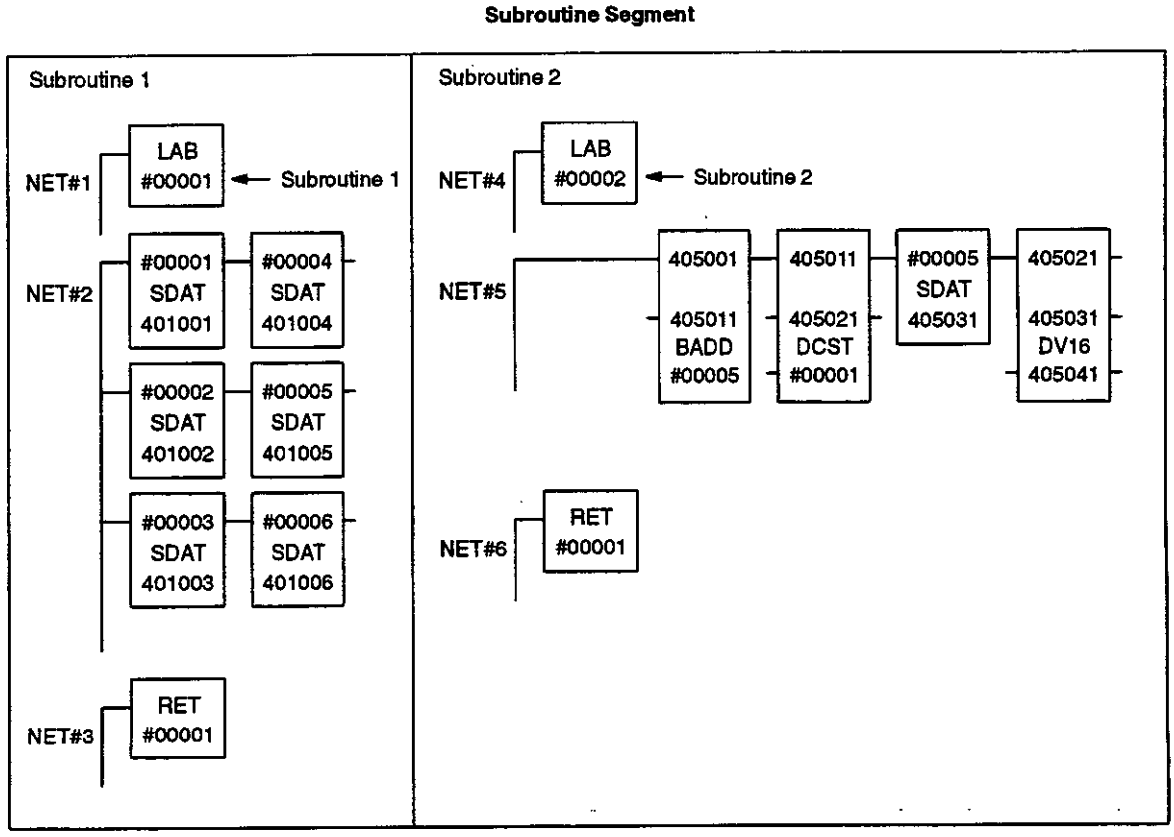


Figure 11.2 Subroutines

2. Subroutine Instructions

1) Three instructions are used with subroutines.

   a) **SUBROUTINE JUMP (JSR)**
   Used to call a subroutine.

   b) **SUBROUTINE LABEL (LAB)**
   Used to start a subroutine and assign it a subroutine number.

   c) **SUBROUTINE RETURN (RET)**
   Used to end a subroutine and return to the SUBROUTINE JUMP (JSR) instruction from which the subroutine was called.

2) Subroutines can be nested, i.e., another subroutine can be called from within a subroutine. Up to 100 subroutines can be nested. Excessive nesting should be avoided to simplify the program and avoid difficulties in debugging and troubleshooting.

3) Looping is possible by nesting subroutines, i.e., the subroutine can be repeatedly executed in one scan while a specific condition is met. Up to 100 loops can be executed.

Excessive looping of subroutines with long execution times can increase the scan time, causing inconsistency in scan times or even causing the CPU to stop due to a watchdog timer error. Be careful when programming loops.

3. Subroutine Usefulness

Subroutines can be used for the following purposes.

1) Reducing User Program Memory Usage
User program memory can be saved by programming subroutines for ladder programming used repeatedly at various places in the program in one scan.

2) Reducing Scan Time
The scan time can be reduced by programming subroutines for ladder programming that is not used very often. The ladder programming in the subroutine is solved only when it is required, reducing the scan time whenever the subroutine is not executed.

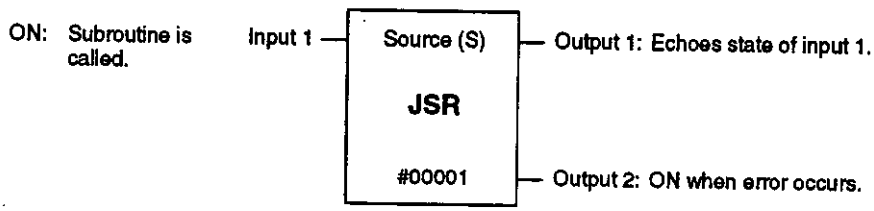
11.2.2 SUBROUTINE JUMP (JSR)

1. Function

1) A subroutine is called, i.e., a jump is made to the specified subroutine and the ladder programming is solved.

2) SUBROUTINE JUMP is used together with SUBROUTINE LABEL (used to assign a subroutine number) and SUBROUTINE RETURN (used to end the subroutine).

2. Structure



1) JSR is the symbol for SUBROUTINE JUMP.

2) JSR requires two elements, one top element and one bottom element, located vertically on the network. Refer to *Table 11.2* for details on specifying constants or registers for these elements.

Table 11.2 Structural Elements of JSR

Element	Meaning	Possible Settings	
Top (S)	The value of the constant or the contents of the register with the specified reference number specifies the number of the subroutine to call (1 to 1,023).	Constant:	#00001 to #01023
		Holding register:	400001 to 409999 (W00001 to W09999)
		Constant register:	700001 to 704096 (K00001 to K04096)
		Link register:	R10001 to R11024 R20001 to R21024
Bottom	Fixed	Constant:	#00001

3) JSR Storage Locations

JSR can be stored in the high-speed segment, normal segment, or the subroutine segment.

3. Operation

1) The following processing will be performed if JSR is solved when input 1 is ON. Here, the value of the constant or contents of the register specified for the top element of JSR is S. This processing is not performed if nesting has passed 100 levels or 100 loops.

   a) If S is between 1 and 1,023, the following processing will be performed.

      (1) A jump is made to subroutine S if it exists and the ladder programming in it is solved. Output 1 turns ON and output 2 turns OFF.

      (2) If subroutine S does not exist, outputs 1 and 2 will turn ON and execution of JSR will end.

   b) If S is not between 1 and 1,023, the subroutine does not exist, outputs 1 and 2 will turn ON, and execution of JSR will end.

2) If nesting has passed 100 levels or 100 loops, the following processing will be performed even if subroutine S actually exists.

   a) A jump is made from JSR to the 100th subroutine or loop and the ladder programming is solved.

   b) A jump is not made from JSR to the 101st subroutine or loop, outputs 1 and 2 from the JSR in the 100th subroutine turn ON, and execution of the 100th JSR ends.

3) The following illustration outlines the operation of JSR.

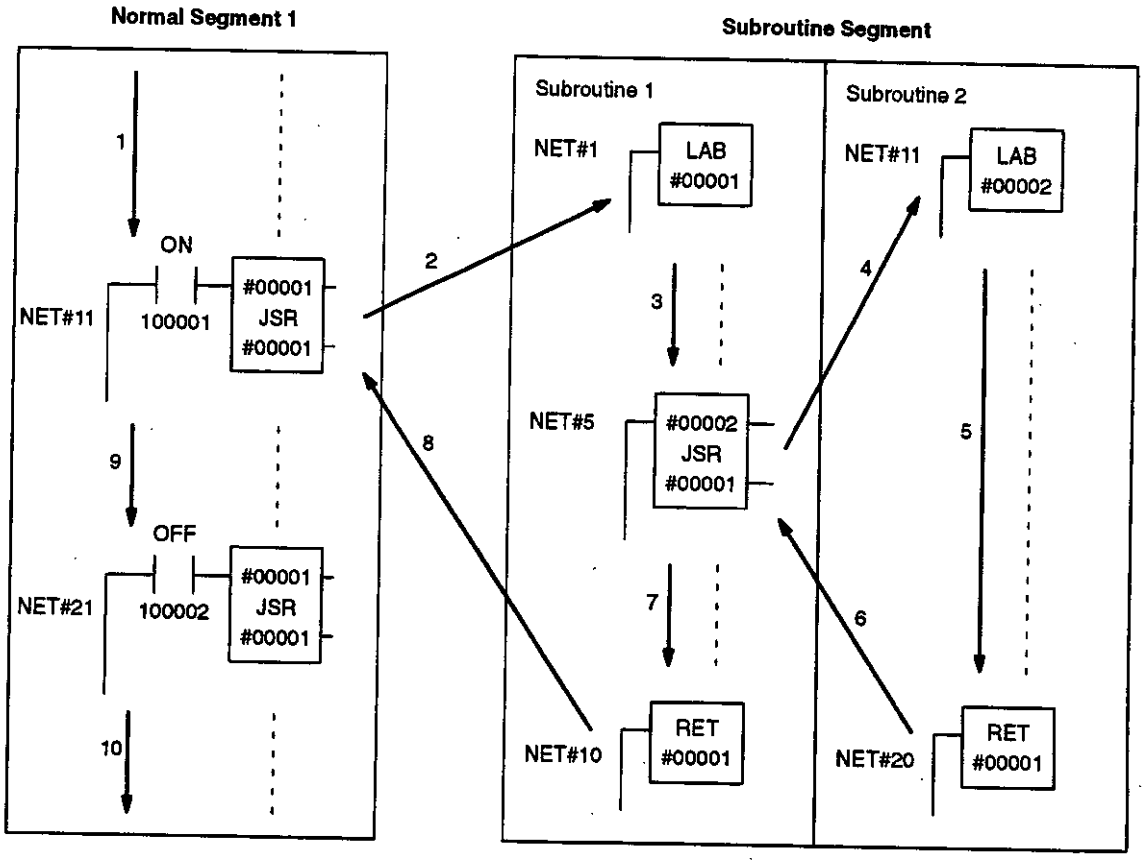


Figure 11.3 Operation of JSR

a) The ladder programming shown in the above illustration is stored in normal segment 1 and the subroutine segment.

b) When input relay 100001 turns ON, the ladder logic will be solved as shown by the numbers (1 to 10) in the illustration.

c) The same sequence of ladder programming will be solved when input relay 100002 turns ON as well.

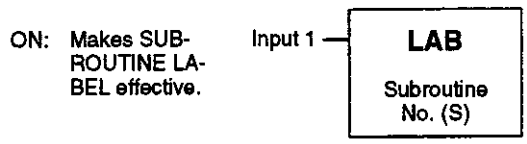
11.2.3 SUBROUTINE LABEL (LAB)

1. Function

1) A subroutine is started and assigned a subroutine number.

2) SUBROUTINE LABEL is used together with SUBROUTINE RETURN (used to end the subroutine).

2. Structure



1) LAB is the symbol for SUBROUTINE LABEL.

2) LAB requires one element (bottom) on the network. Specify a constant between #00001 and #01023 for the element. The specified constant is the subroutine number.

Note (1) **Programming Operation**
LAB cannot be stored on a network, deleted, or changed while the CPU Module is running. The CPU Module must be stopped to perform any of these programming operations for LAB.

(2) **Storage Locations**
LAB must be stored on the first row and first column of a network in the subroutine segment. It cannot be stored anywhere else.

(3) **Subroutine Numbers**
Each subroutine number can be used only once in a LAB instruction, i.e., each subroutine must have a unique subroutine number.

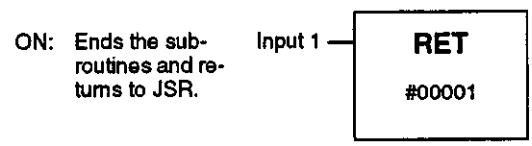
11.2.4 SUBROUTINE RETURN (RET)

1. Function

1) A subroutine is ended, i.e., solving the ladder programming in the subroutine is ended and a return is made to the JSR that called the subroutine.

2) SUBROUTINE RETURN is used together with SUBROUTINE LABEL (used to assign a subroutine number).

2. Structure



1) RET is the symbol for SUBROUTINE RETURN.

2) RET requires one element (bottom) on the network. Specify the constant #00001 for the element.

3) **Storage Locations**
RET can be stored on any column of the first row in a network in the subroutine segment. It cannot be stored after a coil (including link coils, MC coils, and MC control coils) or in the high-speed or a normal segment.

3. Operation

1) If RET is solved when input 1 is ON, solving of the subroutine will be ended and a return will be made to the JSR that called the subroutine.

2) Solving of the subroutine will be continued if RET is solved when input 1 is OFF. In this case, solving of the subroutine is ended and a return is made to the JSR that called the subroutine when either of the following conditions is met.

   a) LAB is executed.

   b) The last instruction in the last network of the subroutine segment is solved.

3) The processing given in item 2), above, will also be carried out if a RET corresponding to the LAB does not exist for a subroutine.

11.2.5 Application Example

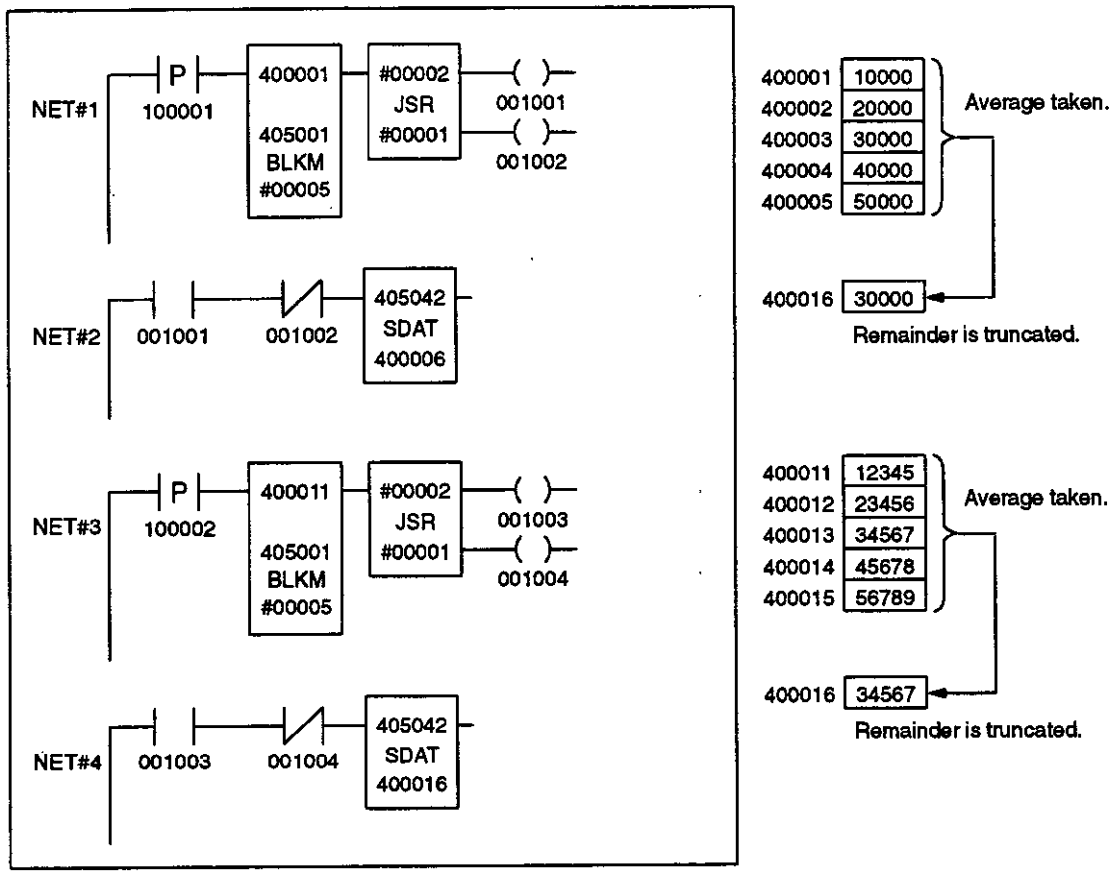
◀EXAMPLE▶ An example of using the subroutine instructions is shown below.

Example
The average of the contents (0 to 65,535) of 5 registers is taken.

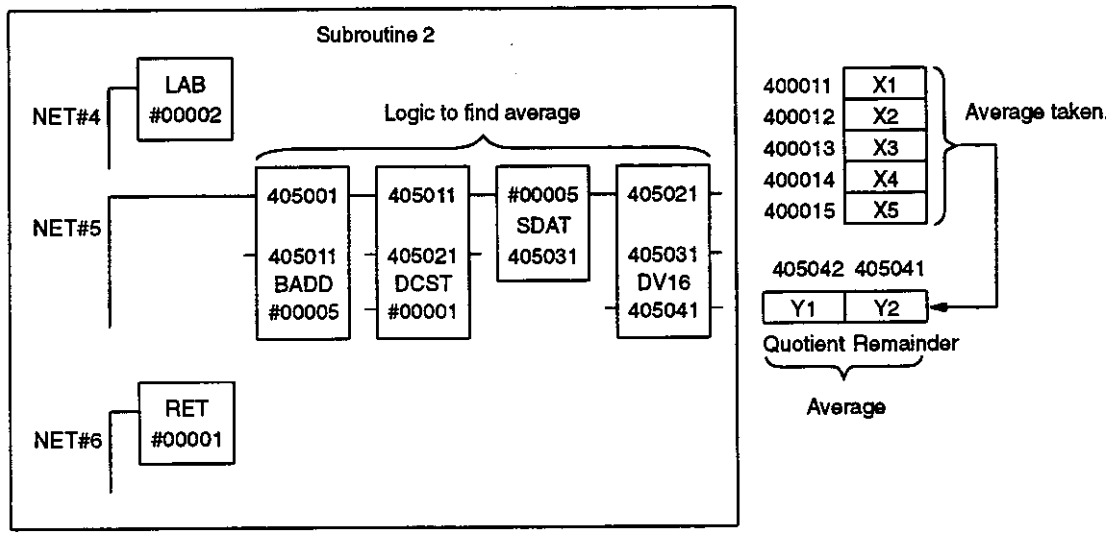
1) When input relay 100001 turns ON, the average of the contents of holding registers 400001 to 400005 will be taken and stored in holding register 400006.

2) When input relay 100002 turns ON, the average of the contents of holding registers 400011 to 400015 will be taken and stored in holding register 400016.

Normal Segment



Subroutine Segment



11.3 Master Control Instructions

This section describes the functions, structures, and operation of master control instructions and provides simple examples of their application.

11.3.1 Master Control Instructions

1. Function

1) Master control instructions can be used to turn OFF the power rail in specified networks. If the power rail is turned OFF in a network, the ladder programming will be solved without power flow from the power rail.

2) One example of using master control instructions is to turn OFF all the coils used in the relay circuits of the specified network.

2. Instructions

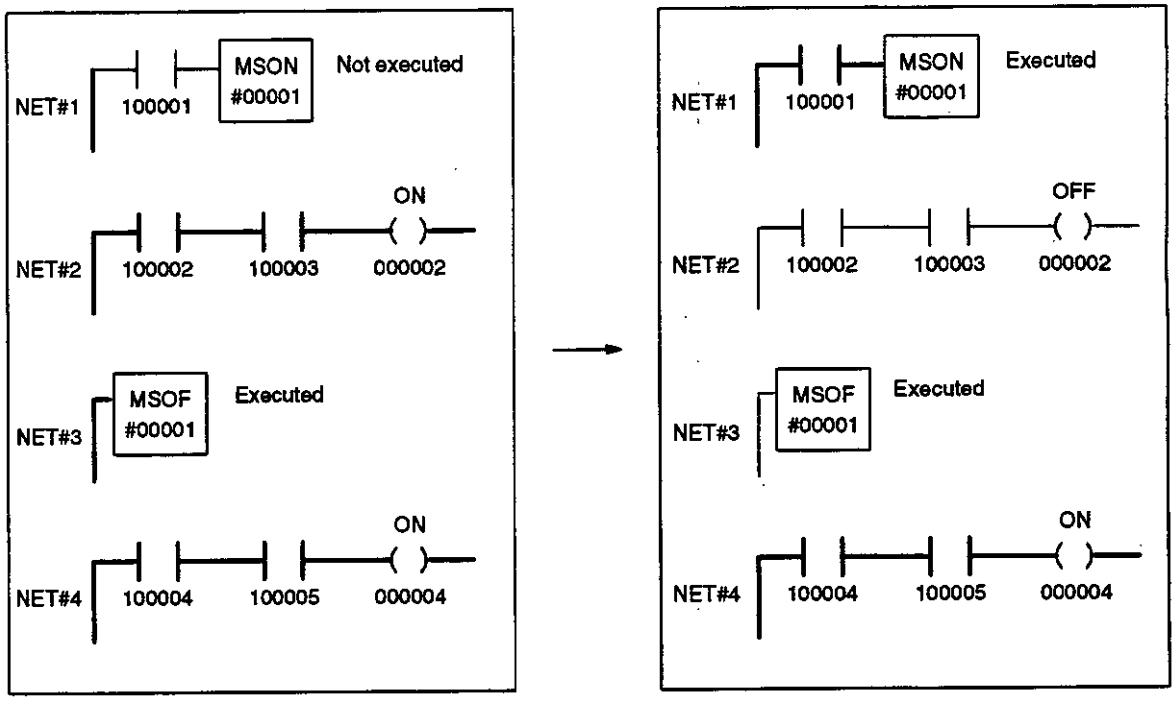
There are two master control instructions. These instructions are always used as a pair, as shown in the following example.

1) **MASTER CONTROL ON (MSON)**
MSON turns OFF the network's power rail.

2) **MASTER CONTROL OFF (MSOF)**
MSOF turns ON the network's power rail.

Example
The power rails in networks 2 and 3 will be turned OFF when input relay 100001 turns ON.

a) Input Relay 100001 is OFF. **b) Input Relay 100001 is ON.**



11.3.2 MASTER CONTROL ON (MSON)

1. Function

The power rail to a network is turned OFF. MASTER CONTROL ON is used in a pair with MASTER CONTROL OFF, which turns the power rail back ON.

2. Structure

ON: Power rails in next and following networks turned OFF.

Input 1 — **MSON** #00001

1) MSON is the symbol for MASTER CONTROL ON.

2) MSON requires one element (bottom) on the network. Specify the constant #00001 for the element.

Note MSON cannot be stored on a network, deleted, or moved while the CPU Module is running. The CPU Module must be stopped to perform any of these programming operations for MSON.

3. Operation

1) High-speed and Normal Segments

a) MSON is effective only in the segment in which it is stored. It cannot be used to affect other segments.

b) Assuming that MSON is in network N and MSOF is in network M of the same segments, the following processing is performed as long as input 1 to MSON is ON. Refer to the illustration on the following page.

(1) Normal Operation (N < M)
The power rails for all networks from network N+1 to network M is turned OFF.

(2) Error Operation (N ≥ M)
The power rails for all networks from network N+1 to the last network in the segment, network Z, is turned OFF.

c) If a segment containing MSON does not contain a MSOF and MSON is executed with input 1 ON, the power rails for all networks from the network after the one containing MSON to the last network in the segment will be turned OFF.

d) The ladder programming in the networks in which the power rail is turned OFF is solved without power flow from the power rail.

e) An outline of the operation of MSON is provided in the following illustration.

(1) Normal Operation (N < M)
All network power rails from NET# (N+1) to NET#M will be turned OFF when MSON is executed.

(2) Error Operation (N ≥ M)
All network power rails from NET# (N+1) to NET#Z will be turned OFF when

MSON is executed. The same operation will be performed if the MSOF that is paired with NSON is not stored in the same segment.

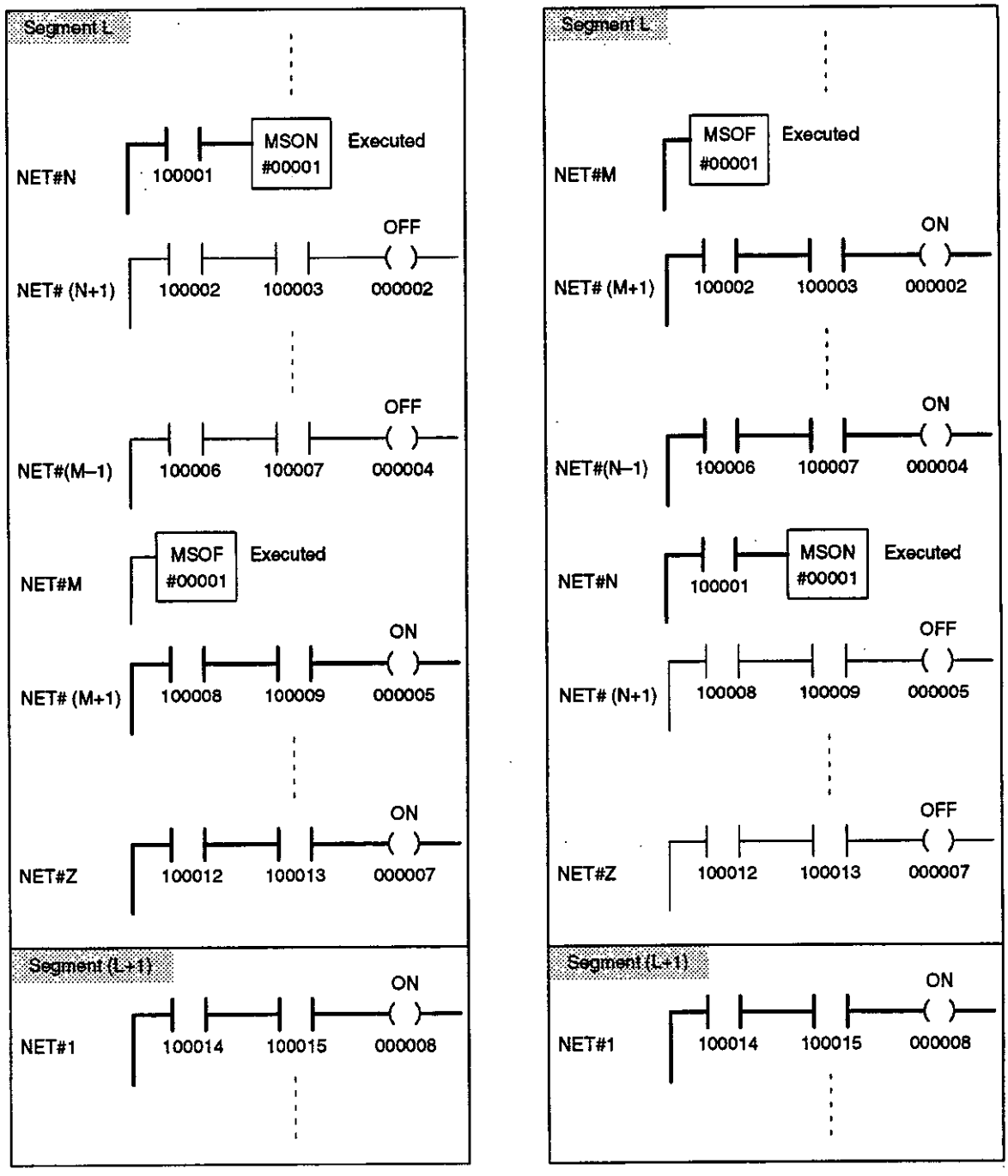


Figure 11.4 MSON Operation in High-speed and Normal Segments

2) Subroutine Segment

a) MSON is effective only in the subroutine in which it is stored. It cannot be used to affect other subroutines, the high-speed segment, or normal segments.

b) Assuming that MSON is in network N and MSOF is in network M of the same subroutine, the following processing is performed as long as input 1 to MSON is ON. Refer to the illustration on the following page.

(1) Normal Operation (N < M)
The power rails for all networks from network N+1 to network M is turned OFF.

(2) Error Operation (N ≥ M)
The power rails for all networks from network N+1 to the last network in the subroutine, network Z, is turned OFF.

c) If a subroutine containing MSON does not contain a MSOF and MSON is executed with input 1 ON, the power rails for all networks from the network after the one containing MSON to the last network in the subroutine will be turned OFF.

d) The ladder programming in the networks in which the power rail is turned OFF is solved without power flow from the power rail.

e) An outline of the operation of MSON is provided in the following illustration.

(1) Normal Operation (N < M)
All network power rails from NET# (N+1) to NET#M will be turned OFF when MSON is executed.

(2) Error Operation (N ≥ M)
All network power rails from NET# (N+1) to NET#Z will be turned OFF when MSON is executed. The same operation will be performed if the MSOF that is paired with NSON is not stored in the same segment.

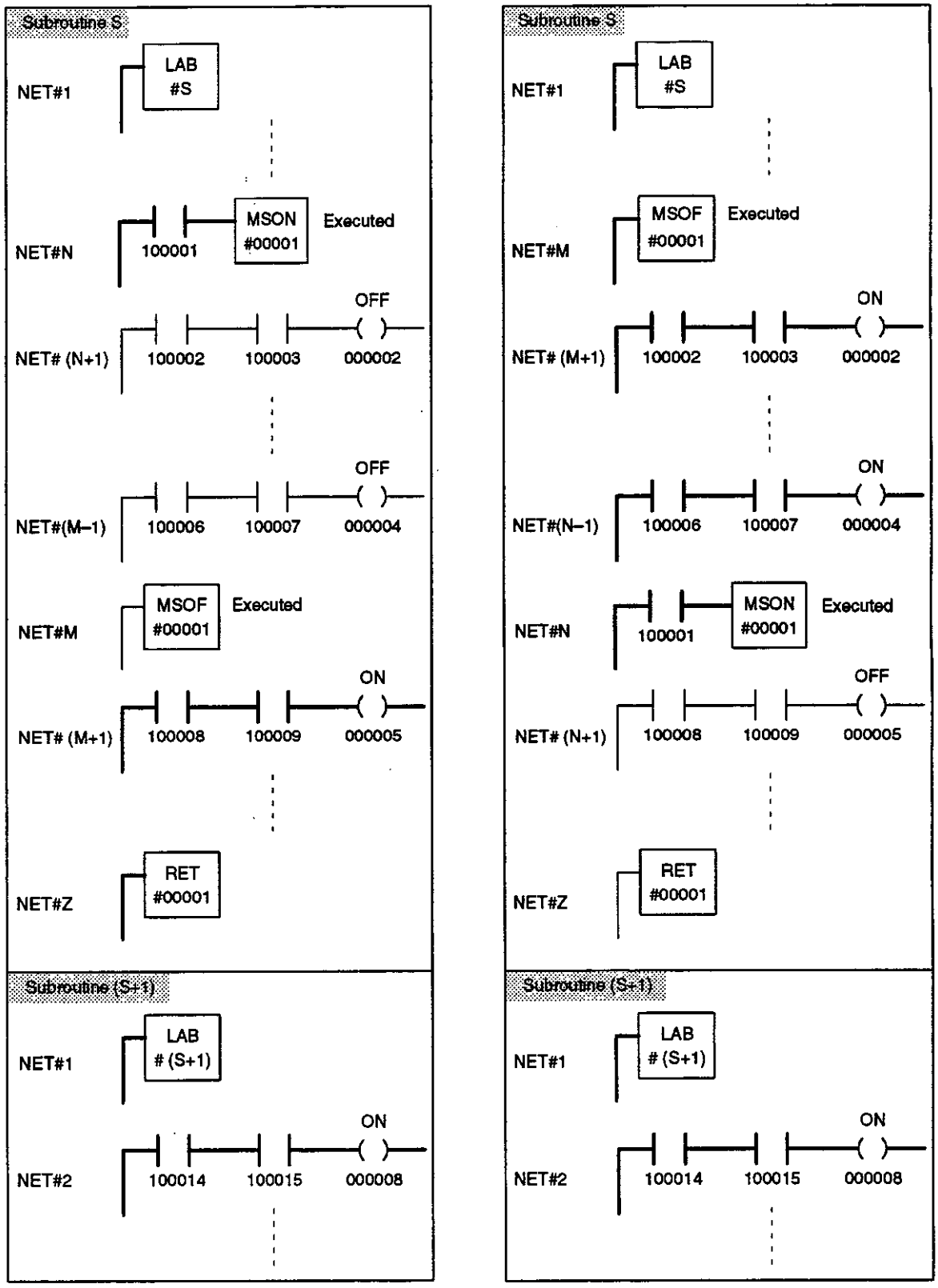


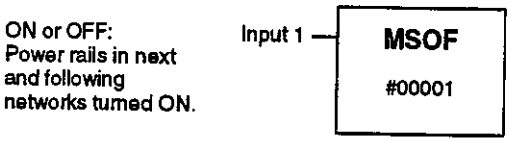
Figure 11.5 MSON Operation in Subroutines

11.3.3 MASTER CONTROL OFF (MSOF)

1. Function

The power rail to a network is turned back ON. MASTER CONTROL OFF is used in a pair with MASTER CONTROL ON, which turns the power rail OFF.

2. Structure



1) MSOF is the symbol for MASTER CONTROL OFF.

2) MSOF requires one element (bottom) on the network. Specify the constant #00001 for the element.

Note MSOF cannot be stored on a network, deleted, or moved while the CPU Module is running. The CPU Module must be stopped to perform any of these programming operations for MSOF.

3. Operation

1) MSOF turn ON the power rails in all networks after the one containing MSOF. The power rails are turned ON regardless of whether input 1 is ON or OFF.

2) The power rail in the network containing MSOF is treated as follows:

   a) If the MSON paired with the MSOF is operating, the power rail in the network containing MSOF will be OFF.

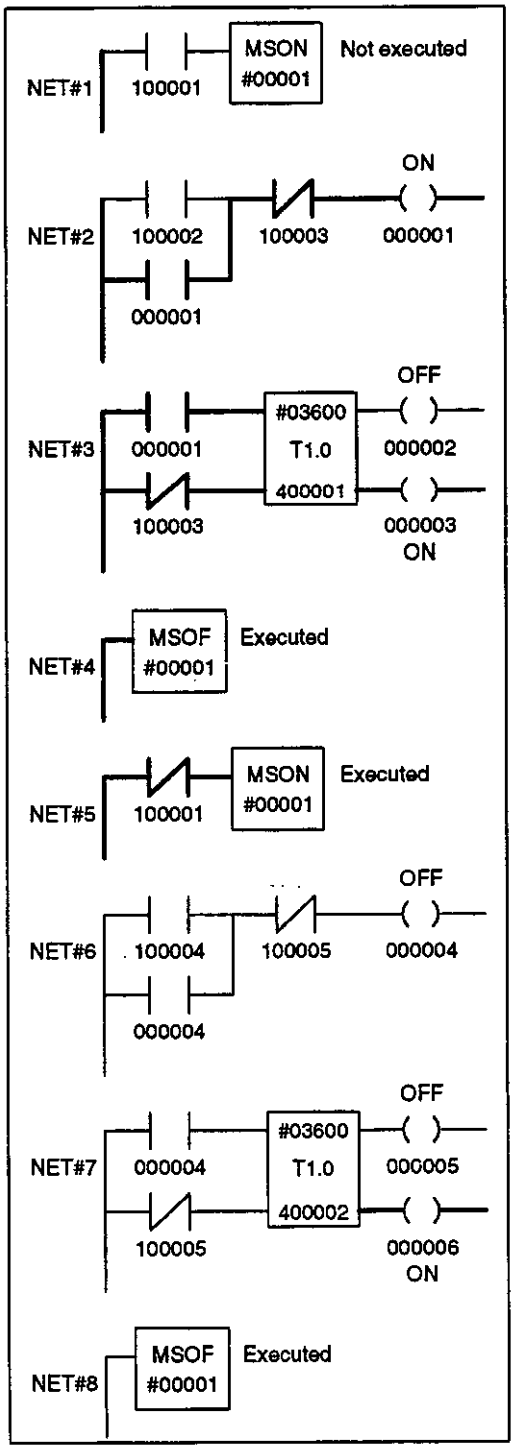
   b) If the MSON paired with the MSOF is not operating, the power rail in the network containing MSOF will be ON.

11.3.4 Application Example

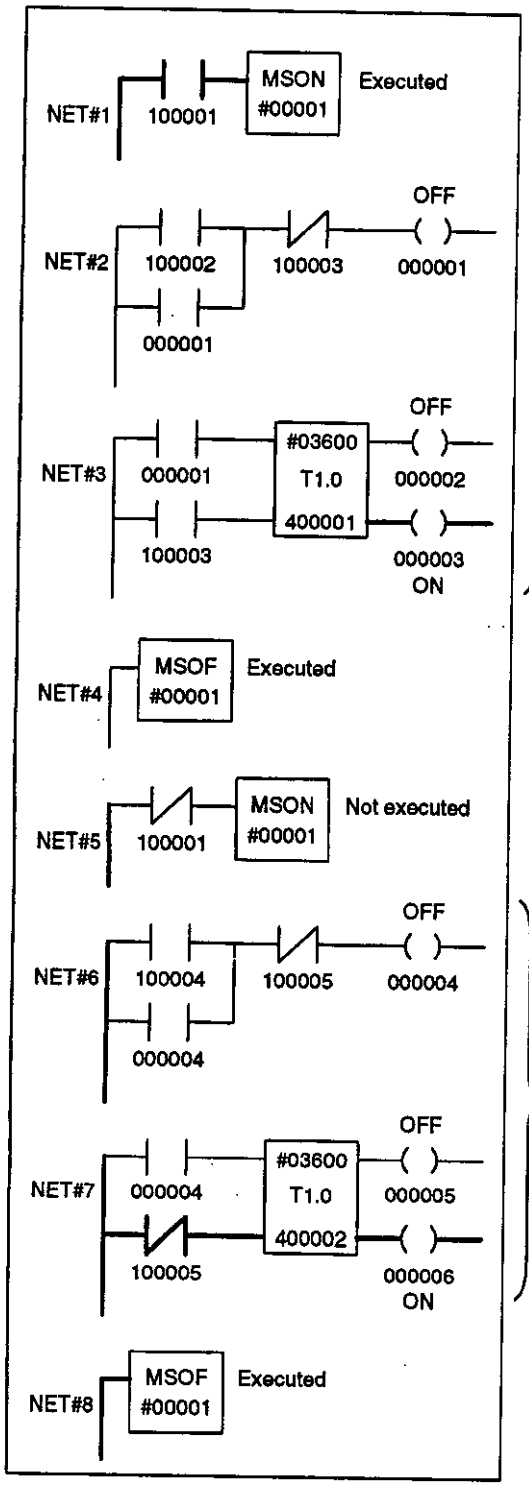
◀EXAMPLE▶ The following example illustrates the usage of the master control instructions.

Assume that the following ladder logic is stored in networks 1 to 8 of normal segment 1.

1) The ladder programming will be solved as shown below when input relay 100001 is OFF.



2) The ladder programming will be solved as shown below when input relay 100001 is ON.



MSON operates, so the power rails in networks 2 to 4 are turned OFF.

Logic solved without power flowing from the power rail.

Coil 000001 turns OFF.

Timer is reset and power flows from output 2, turning ON coil 000003.

The MSOF paired with MSON in network 1.

MSON does not operate and therefore has no effect.

Logic is solved with power flowing from the power rail.

The MSOF paired with MSON in network 5.

11.3.5 Building Programs

1. Programming Operations

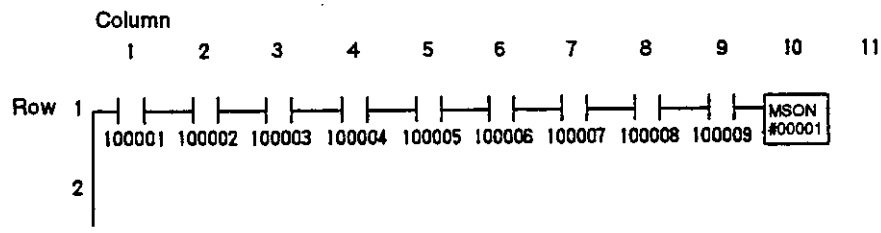
Master control instructions cannot be stored on a network, deleted, or moved while the CPU Module is running. The CPU Module must be stopped to perform any of these programming operations for master control instructions.

2. Storage Locations on Networks

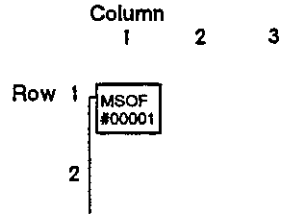
All master control instructions require only one element on the network, so they can be stored anywhere on a 7-row by 10-column matrix (rows 1 through 7 and columns 1 through 10).

Note Master control instructions cannot, however, be placed to the right of coils (including output coils, internal coils, link coils, MC coils, and MC control coils).

Example 1



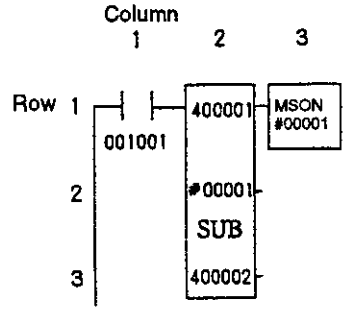
Example 2



3. Inputs

Inputs to master control instructions input 1 can be connected to relay elements (except coils) and/or outputs from timers, counters, math instructions, data transfer instructions, other instructions, etc. MASTER CONTROL OFF (MSOF), however, operates the same regardless of whether input 1 is ON or OFF, so it is meaningless to connect it to condition circuits. MSOF is thus normally programmed as shown in *Example 2*, above.

Example



4. Outputs

Master control instructions do not have outputs.

5. Effective Range of MSON

1) The effective range of MSON stored in the high-speed segment is limited to the high-speed segment only. It cannot affect normal segments or subroutines.

2) The effective range of MSON stored in the normal segments is limited to the normal segment in which MSON is stored. It cannot affect the high-speed segment, other normal segments, or subroutines.

3) The effective range of MSON stored in a subroutine is limited to the subroutine in which MSON is stored. It cannot affect the high-speed segment, normal segments, or other subroutines.

6. Using MSON and MSOF Together

1) MSON and MSOF are always used as a pair.

2) The pair must be used in the same segment or subroutine.

3) MSOF should be stored in a network after the network containing MSON.

7. Power Flow

1) Power flow will be provided from output 2 of the following instructions when solved without power flow supply from the power rail.

   a) 1-SECOND TIMER, 0.1-SECOND TIMER, 0.01-SECOND TIMER, and 0.001-SECOND TIMER,

   b) UP COUNTER and DOWN COUNTER

2) Power flow from outputs 2 and 3 will be supplied for the following instructions depending on the relationship between the pointer value, n, and the table size, Z, even when solved without power flow supply from the power rail.

   a) REGISTER-TO-TABLE MOVE, TABLE-TO-REGISTER MOVE, TABLE-TO-TABLE MOVE, FIRST IN, and FIRST OUT

   b) LOGICAL BIT MODIFY and LOGICAL SENSE

Appendix A

Index of Ladder Logic Elements and Instructions

This index provides an easy way to reference GL120 and GL130 instructions through either the instruction name or the instruction symbol.

Numbers

A

B

C

D

F

I

T

U

X

MEMOCON GL120, GL130
SOFTWARE USER'S MANUAL
VOL.2

TOKYO OFFICE
New Pier Takeshiba South Tower, 1-16-1, Kaigan, Minatoku, Tokyo 105-6891 Japan
Phone 81-3-5402-4511 Fax 81-3-5402-4580

YASKAWA ELECTRIC AMERICA, INC.
2121 Norman Drive South, Waukegan, IL 60085, U.S.A.
Phone 1-847-887-7000 Fax 1-847-887-7370

MOTOMAN INC. HEADQUARTERS
805 Liberty Lane West Carrollton, OH 45449, U.S.A.
Phone 1-937-847-6200 Fax 1-937-847-6277

YASKAWA ELÉTRICO DO BRASIL COMÉRCIO LTDA.
Avenida Fagundes Filho, 620 Bairro Saude-Sao Paulo-SP, Brazil CEP: 04304-000
Phone 55-11-5071-2552 Fax 55-11-5581-8795

YASKAWA ELECTRIC EUROPE GmbH
Am Kronberger Hang 2, 65824 Schwalbach, Germany
Phone 49-6196-569-300 Fax 49-6196-888-301

Motoman Robotics Europe AB
Box 504 S38525 Torsås, Sweden
Phone 46-486-48800 Fax 46-486-41410

Motoman Robotec GmbH
Kammerfeldstraße 1, 85391 Allershausen, Germany
Phone 49-8166-900 Fax 49-8166-9039

YASKAWA ELECTRIC UK LTD.
1 Hunt Hill Orchardton Woods Cumbernauld, G68 9LF, United Kingdom
Phone 44-1236-735000 Fax 44-1236-458182

YASKAWA ELECTRIC KOREA CORPORATION
Kfpa Bldg #1201, 35-4 Youido-dong, Yeongdungpo-Ku, Seoul 150-010, Korea
Phone 82-2-784-7844 Fax 82-2-784-8495

YASKAWA ELECTRIC (SINGAPORE) PTE. LTD.
151 Lorong Chuan, #04-01, New Tech Park Singapore 556741, Singapore
Phone 65-282-3003 Fax 65-289-3003

YASKAWA ELECTRIC (SHANGHAI) CO., LTD.
4F No.18 Aona Road, Waigaoqiao Free Trade Zone, Pudong New Area, Shanghai 200131, China
Phone 86-21-5866-3470 Fax 86-21-5866-3869

YATEC ENGINEERING CORPORATION
Shen Hsiang Tang Sung Chiang Building 10F 146 Sung Chiang Road, Taipei, Taiwan
Phone 886-2-2563-0010 Fax 886-2-2567-4677

YASKAWA ELECTRIC (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
Jianguomenwai Avenue, Beijing 100020, China
Phone 86-10-6532-1850 Fax 86-10-6532-1851

TAIPEI OFFICE
Shen Hsiang Tang Sung Chiang Building 10F 146 Sung Chiang Road, Taipei, Taiwan
Phone 886-2-2563-0010 Fax 886-2-2567-4677

SHANGHAI YASKAWA-TONGJI M & E CO., LTD.
27 Hui He Road Shanghai China 200437
Phone 86-21-6531-4242 Fax 86-21-6553-6060

BEIJING YASKAWA BEIKE AUTOMATION ENGINEERING CO., LTD.
30 Xue Yuan Road, Haidian, Beijing P.R. China Post Code: 100083
Phone 86-10-6233-2782 Fax 86-10-6232-1536

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

YASKAWA ELECTRIC CORPORATION

YASKAWA

MANUAL NO. SIEZ-C825-20.12B