

# HITACHI

SOFTWARE MANUAL  
PROGRAMMING  
**S10V LADDER CHART**  
For Windows®

---

**S10V**  
Programmable Controller

SVE-3-121(H)

SOFTWARE MANUAL  
PROGRAMMING  
**S10V LADDER CHART**  
For Windows®

---



First Edition, August 2005, SVE-3-121(E) (out of print)  
Second Edition, July 2006, SVE-3-121(F) (out of print)  
Third Edition, October 2007, SVE-3-121(G) (out of print)  
Fourth Edition, February 2013, SVE-3-121(H)

All Rights Reserved, Copyright © 2005, 2013, Hitachi, Ltd.

The contents of this publication may be revised without prior notice.


No part of this publication may be reproduced in any form or by any means without permission in writing from the publisher.


Printed in Japan.


BI<IC> (FL-MW2007)

## SAFETY PRECAUTIONS



- Read this manual thoroughly and follow all the safety precautions and instructions given in this manual before operations such as system configuration and program creation.
- Keep this manual handy so that you can refer to it any time you want.
- If you have any question concerning any part of this manual, contact your nearest Hitachi branch office or service engineer.
- Hitachi will not be responsible for any accident or failure resulting from your operation in any manner not described in this manual.
- Hitachi will not be responsible for any accident or failure resulting from modification of software provided by Hitachi.
- Hitachi will not be responsible for reliability of software not provided by Hitachi.
- Make it a rule to back up every file. Any trouble on the file unit, power failure during file access or incorrect operation may destroy some of the files you have stored. To prevent data destruction and loss, make file backup a routine task.
- Furnish protective circuits externally and make a system design in a way that ensures safety in system operations and provides adequate safeguards to prevent personal injury and death and serious property damage even if the product should become faulty or malfunction or if an employed program is defective.
- If an emergency stop circuit, interlock circuit, or similar circuit is to be formulated, it must be positioned external to the programmable controller. If you do not observe this precaution, equipment damage or accident may occur when this programmable controller becomes defective.
- Before changing the program, generating a forced output, or performing the RUN, STOP, or like procedure during an operation, thoroughly verify the safety because the use of an incorrect procedure may cause equipment damage or other accident.
- This manual contains information on potential hazards that is intended as a guide for safe use of this product. The potential hazards listed in the manual are divided into four hazard levels of danger, warning, caution, and notice, according to the level of their severity. The following are definitions of the safety labels containing the corresponding signal words DANGER, WARNING, CAUTION, and NOTICE.

 **DANGER** : This safety label identifies precautions that, if not heeded, will result in death or serious injury.

 **WARNING** : Identifies precautions that, if not heeded, could result in death or serious injury.

 **CAUTION** : Identifies precautions that, if not heeded, could result in minor or moderate injury.

**NOTICE** : This safety label without a safety alert symbol identifies precautions that, if not heeded, could result in property damage or loss not related to personal injury.

Failure to observe any of the  **CAUTION** and  **NOTICE** statements used in this manual could also lead to a serious consequence, depending on the situation in which this product is used. Therefore, be sure to observe all of those statements without fail.

The following are definitions of the phrases “serious injury,” “minor or moderate injury,” and “property damage or loss not related to personal injury” used in the above definitions of the safety labels.

***Serious injury:*** Is an injury that requires hospitalization for medical treatment, has aftereffects, and/or requires long-term follow-up care. Examples of serious injuries are as follows: vision loss, burn (caused by dry heat or extreme cold), electric-shock injury, broken bone, poisoning, etc.

***Minor or moderate injury:*** Is an injury that does not require either hospitalization for medical treatment or long-term follow-up care. Examples of minor or moderate injuries are as follows: burn, electric-shock injury, etc.

***Property damage or loss not related to personal injury:*** Is a damage to or loss of personal property. Examples of property damages or losses not related to personal injury are as follows: damage to this product or other equipment or their breakdown, loss of useful data, etc.

The safety precautions stated in this manual are based on the general rules of safety applicable to this product. These safety precautions are a necessary complement to the various safety measures included in this product. Although they have been planned carefully, the safety precautions posted on this product and in the manual do not cover every possible hazard. Common sense and caution must be used when operating this product. For safe operation and maintenance of this product, establish your own safety rules and regulations according to your unique needs. A variety of industry standards are available to establish such safety rules and regulations.

1.  Hazard Warning Statements

The following are the hazard warning statements contained in this manual.

1.1 NOTICE Statements

(chapter 2, page 2-17)

<b>NOTICE</b>
The LPU module stops due to “Invalid instruction detected.” This occurs if the long-word register having an odd number or a ladder program containing PSHO and POPO, which were created by the Ladder Chart System of Ver.-Rev. 01-16 or later, is sent to a Ladder Chart System having Ver.-Rev. 01-15 or earlier, or to an LPU having module revision L (Ver.-Rev. 02-05) or earlier using batch loading.

(supplements, page Z-5)

<b>NOTICE</b>
The LPU module stops due to “Invalid instruction detected.” This occurs if the long-word register having an odd number or a ladder program containing PSHO and POPO, which were created by the Ladder Chart System of Ver.-Rev. 01-16 or later, is sent to a Ladder Chart System having Ver.-Rev. 01-15 or earlier, or to an LPU having module revision L (Ver.-Rev. 02-05) or earlier using batch loading.

**This Page Intentionally Left Blank**

This manual provides information on the following program product:

<Program product>

S-7895-02, S10V LADDER CHART SYSTEM, 01-35



## Revision record

Revision No.	Revision record (revision details and reason for revision)	Month, Year	Remarks
E	First Edition	August 2005	S10V Ladder Chart System, 01-09 or later
F	The OPTET module is added for use in Ethernet communication.	July 2006	S10V Ladder Chart System, 01-12 or later
G	Extension of the converter function is added.	October 2007	S10V Ladder Chart System, 01-16 or later
H	<ul style="list-style-type: none"><li>• All the safety precautions and instructions in this manual have been reviewed and necessary changes are added to them.</li><li>• Windows® 7 (32-bit) operating system is newly supported.</li></ul>	February 2013	S10V Ladder Chart System, 01-34 or later

In addition to the above changes, all the unclear descriptions and typographical errors found are also corrected without prior notice.

## PREFACE

This manual describes a variety of instructions that are used when creating ladder programs. The instructions used in ladder programs may be classified into two major groups: ladder instructions and arithmetic function instructions. Ladder instructions are used to control relay circuits, whereas arithmetic function instructions are used to perform arithmetic operations, such as addition, subtraction, multiplication, and division.

The S10V product used with ladder programs is available in two types: standard model and environmentally resistant model. The environmentally resistant model has thicker platings and coatings than those for the standard model.

The model number of the environmentally resistant model is marked by adding the suffix “-Z” to the model number of the standard model.

(Example) Standard model: LQP520

Environmentally resistant model: LQP520-Z

This manual is applicable to both the standard model and environmentally resistant model. Although the descriptions contained in this manual are based on the standard model, follow the instructions set forth in this manual for proper use of the product even if you use the environmentally resistant model.

Task initiation using process registers (generically named “P”) is supported only by LPU and CMU modules whose module revisions are the same as those listed below. With any other module revision, the use of process registers does not initiate the tasks at all.

Module name	Module model	Module revision
LPU (basic module)	LQP510	D or later
CMU	LQP520	B or later

<Related manual>

SOFTWARE MANUAL OPERATION S10V LADDER CHART For Windows®  
(Manual number SVE-3-131)

<Trademarks>

- Microsoft® Windows® operating system, Microsoft® Windows® 2000 operating system, Microsoft® Windows® XP operating system, Microsoft® Windows® 7 (32-bit) operating system are registered trademarks of Microsoft Corporation in the United States and/or other countries.
- Ethernet® is a registered trademark of Xerox Corp.

# CONTENTS

1	LADDER INSTRUCTIONS .....	1-1
1.1	Ladder Programs .....	1-2
1.2	Operation Sequence of Ladder Programs .....	1-5
1.3	Ladder Program Instructions .....	1-6
1.3.1	Ladder program instructions .....	1-6
1.3.2	a-contacts .....	1-9
1.3.3	b-contacts .....	1-10
1.3.4	Rising-edge and falling-edge contacts .....	1-11
1.3.5	Operation result push, read, and pop .....	1-11
1.3.6	Operation result push + a-contact, read + a-contact, and pop + a-contact .....	1-12
1.3.7	Operation result push + b-contact, read + b-contact, and pop + b-contact .....	1-13
1.3.8	Block union -- parallel connection (ORB) .....	1-13
1.3.9	NOT .....	1-14
1.3.10	Coils .....	1-15
1.3.11	Set and reset coils .....	1-15
1.3.12	Comparison instructions .....	1-16
1.3.13	Specifying indices in ladder instructions .....	1-19
1.3.14	Circuits and steps .....	1-20
1.4	Register Statuses at a Reset, Power Recovery, and State Transition between STOP and RUN .....	1-21
1.5	Registers .....	1-22
1.5.1	Registers usable in ladder instructions .....	1-22
1.5.2	Register numbers .....	1-24
1.6	Ladder Watchdog Timer .....	1-62
1.6.1	An outline of the ladder watchdog timer's operation .....	1-62
1.6.2	Range of settable monitoring time values .....	1-63
1.6.3	Error information presented upon ladder WDT errors .....	1-63
2	ARITHMETIC FUNCTIONS .....	2-1
2.1	Functional Overview .....	2-2
2.2	Functional Specifications .....	2-4
2.3	Registers Used in Arithmetic Functions .....	2-8
2.3.1	Registers usable in arithmetic functions .....	2-8
2.3.2	Handling of bit registers .....	2-11
2.3.3	Relationships between bit registers and word registers .....	2-12
2.4	Inputs to Arithmetic Functions .....	2-13

2.5	Arithmetic Functions .....	2-20
2.6	Details on the Instructions .....	2-25
2.7	Ethernet Communication Instructions .....	2-180
2.7.1	Functional overview .....	2-180
2.7.2	Usage .....	2-183
2.7.3	Details on the instructions .....	2-196
2.7.4	Sample programs .....	2-220
SUPPLEMENTS.....		Z-1
SUPPLEMENT A  CHECKING OUT THE AVERAGE SCAN TIME.....		Z-2
A.1	Check-out using the S10V ladder chart system .....	Z-2
A.2	Check-out using a ladder program.....	Z-3
SUPPLEMENT B  SPECIAL PRECAUTIONS .....		Z-4
B.1	Precautions in converting ladder programs .....	Z-4

## FIGURES

Figure A-1	Scan Time-Indicating Circuit .....	Z-3
------------	------------------------------------	-----

## TABLES

Table 1-1	Basic Instructions .....	1-6
Table 1-2	Comparison Instructions .....	1-8
Table 1-3	Arithmetic Function Instructions .....	1-8
Table 1-4	Usable Registers .....	1-22
Table 1-5	Register Numbers .....	1-24
Table 1-6	System Registers .....	1-50
Table 2-1	Registers Usable in Arithmetic Functions.....	2-8

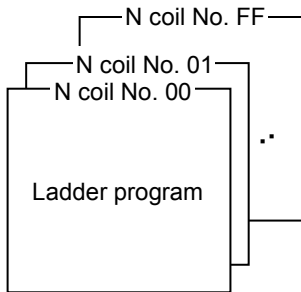
# 1 LADDER INSTRUCTIONS

# 1 LADDER INSTRUCTIONS

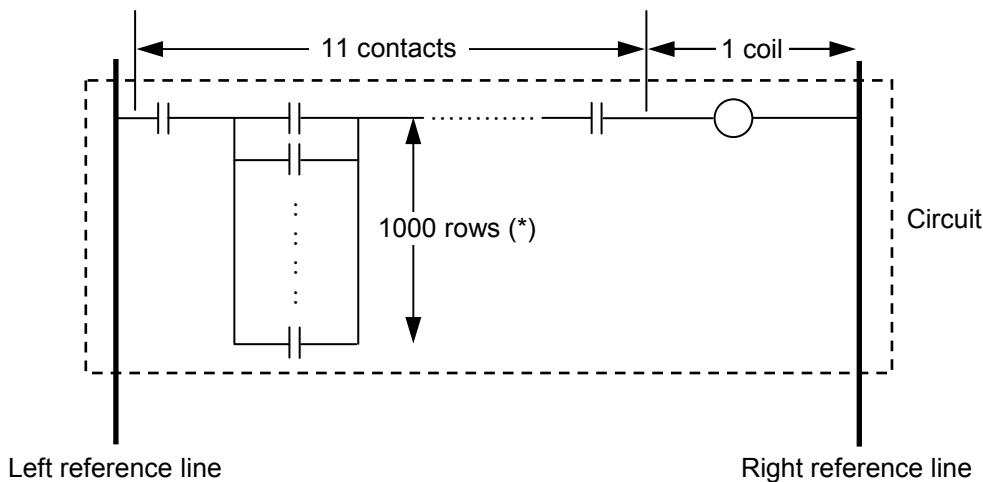
## 1.1 Ladder Programs

The ladder program is a program that is created as a combination of ladder instructions (instructions relating to a- or b-contacts) and arithmetic function instructions (instructions for such operations as addition or subtraction).

Any ladder program is constructed from one or more units of programming each called a nesting coil or, simply, N-coil. Up to 256 N coils, numbered 00 through FF, may be created in a single ladder program. The N coil number 00 is called the master N coil and is executed as the main routine every time a sequence cycle occurs for the execution of the ladder program. Each of the N coil numbers 01 through FF is called a sub-N coil and is initiated as a subroutine from the master N coil or another sub-N coil.

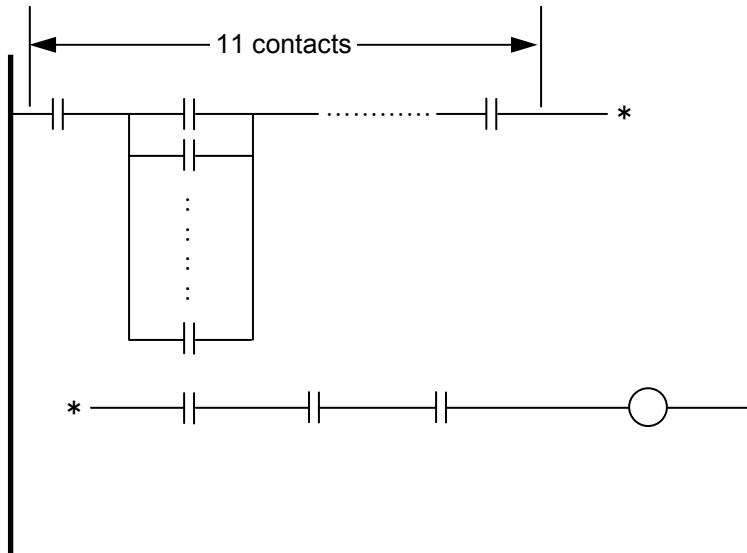


The ladder program can operate only when it contains a proper combination of ladder instructions and arithmetic function instructions. Its operation starts from the left reference line in the ladder chart and ends at the connection with the right reference line therein. The only ladder instruction that can be connected directly to the right reference line is an output instruction (i.e., a coil or arithmetic function instruction). The smallest unit of programming that can run as a ladder program is called a circuit. The maximum allowable size of circuit is 1000 rows (\*) times 12 columns (11 contacts plus 1 output).



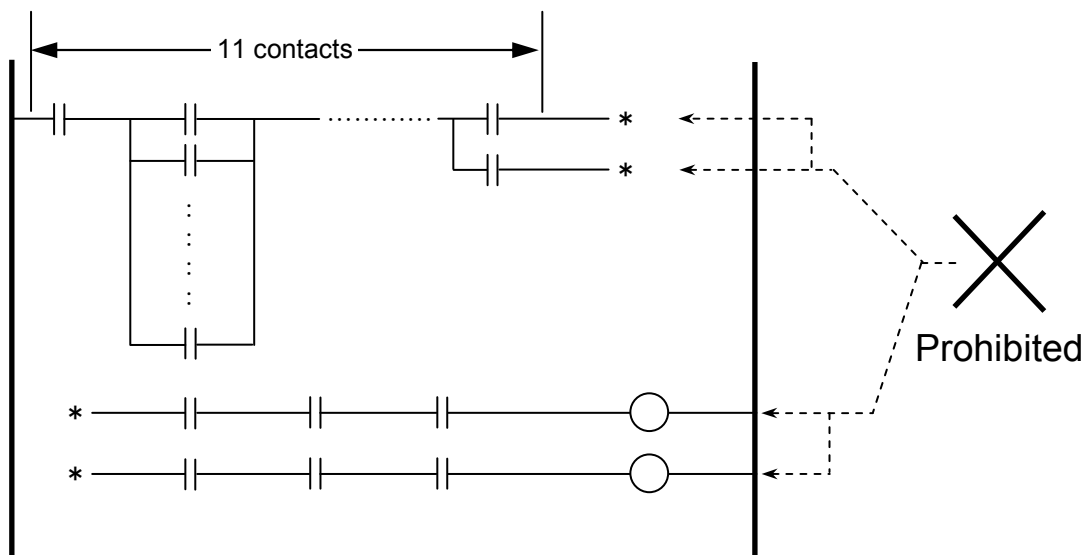
(\*) As the maximum number of rows in a single circuit, 1000 rows are supported for Ver.-Rev. 01-16 and later of the Ladder Chart System. Up to 16 rows are supported before Ver.-Rev. 01-16 of the Ladder Chart System.

If 11 or more contacts need to be AND-connected, the circuit may be wrapped around as shown below, provided that the circuit meets the restrictions described below.



<Restriction 1>

No parallel logic path in a circuit may be wrapped around and then AND-connected. As shown below, branch paths may not be formed before the asterisked (\*) points.



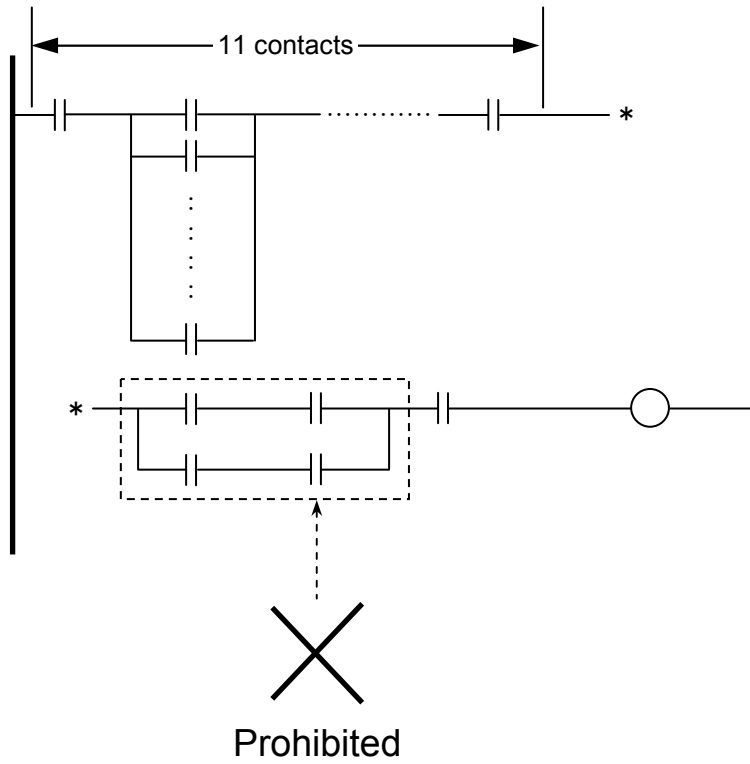


# 1 LADDER INSTRUCTIONS

---

<Restriction 2>

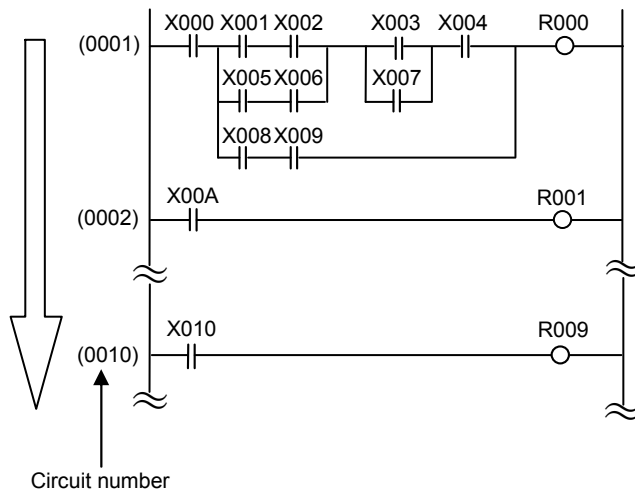
As shown below, branch paths may not be formed after the asterisked (\*) point.



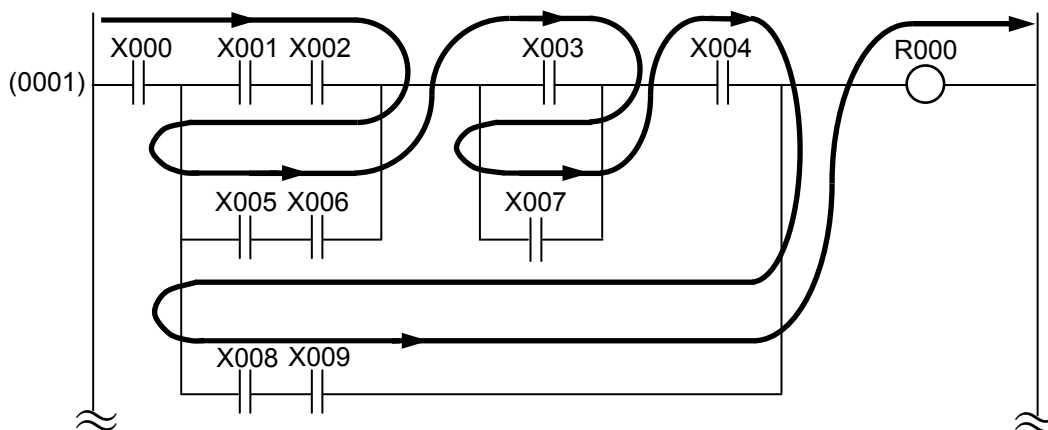
## 1.2 Operation Sequence of Ladder Programs

Ladder programs run in the (ascending) order of specified circuit numbers.

Ladder program example:



The operation sequence in circuits is exemplified below. The operation sequence in any circuit proceeds from left to right. If a circuit contains a set of parallel logic paths, the operation sequence will not proceed to the next logic path until all the parallel paths have been processed. The figure below shows the operation sequence that occurs in a sample circuit. The thick arrows in the circuit indicate the operation sequence.



# 1 LADDER INSTRUCTIONS

## 1.3 Ladder Program Instructions

### 1.3.1 Ladder program instructions

Table 1-1 is a list of all basic ladder instructions that can be used in ladder programs.

#### (1) Basic instructions

One single basic instruction forms one single step in the ladder program, except when an index is specified as the register name. In the latter case, one single basic instruction forms two steps in the ladder program.


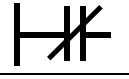

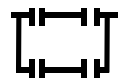
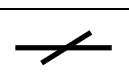
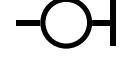
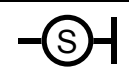
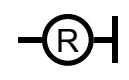
Table 1-1 Basic Instructions

(1/2)

Instruction name	Symbol	Operation code	Function
a-contact start		LD	Denotes the start of an a-contact. (The a-contact becomes ON when the value of a specified register is 1.)
a-contact series-connection		AND	Denotes the series connection of an a-contact with the preceding instruction.
b-contact start		LDN	Denotes the start of a b-contact. (The b-contact becomes ON when the value of a specified register is 0.)
b-contact series-connection		ANDN	Denotes the series connection of a b-contact with the preceding instruction.
Rising-edge contact		EGP	Remains ON only during the sequence cycle in which the rising edge of an input is detected.
Falling-edge contact		EGF	Remains ON only during the sequence cycle in which the falling edge of an input is detected.
Operation result push		SPS	Stores the result of the previous operation performed.
Operation result read		SRD	Reads the operation result stored by an operation result push.
Operation result pop		SPP	Reads the operation result stored by an operation result push and then resets (clears) the stored operation result.
Operation result push + a-contact		SPSAND	Stores the result of the previous operation performed and executes the a-contact.
Operation result read + a-contact		SRDAND	Reads the operation result stored by an operation result push and executes the a-contact.
Operation result pop + a-contact		SPPAND	Reads the operation result stored by an operation result push, executes the a-contact with the obtained operation result, and then resets the stored operation result.

Table 1-1 Basic Instructions

(2/2)


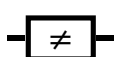
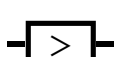
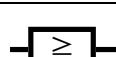
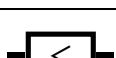

Instruction name	Symbol	Operation code	Function
Operation result push + b-contact		SPSANDN	Stores the result of the previous operation performed and executes the b-contact.
Operation result read + b-contact		SRDANDN	Reads the operation result stored by an operation result push and executes the b-contact.
Operation result pop + b-contact		SPPANDN	Reads the operation result stored by an operation result push, executes the b-contact with the obtained operation result, and then resets the stored operation result.
Block union (parallel connection)		ORB	Connects two logical blocks in parallel.
NOT		NOT	Inverts an input and outputs the result.
Coil		OUT	Produces an output in a specified register. The function of this coil varies with specified registers, as follows: T: ON-delay timer; U: One-shot timer; C: Up-down counter; N: Nesting coil; P: Process initiation coil.
Set coil		OUTS	When the set coil is energized, it maintains the ON condition of the keep relay until the reset coil is energized. Only a keep relay, whose generic register name is K, may be specified for the set or reset coil.
Reset coil		OUTR	

## 1 LADDER INSTRUCTIONS

### (2) Comparison instructions

One single comparison instruction forms three steps in the ladder program, except when an index is specified as the register name. In the latter case, one single comparison instruction forms four or five steps in the ladder program.

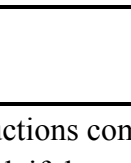
Table 1-2 Comparison Instructions

Instruction name	Symbol	Operation code	Function
Equal (EQU)		LEQU	<p>Each of these comparison instructions compares the contents (data) of two words and, if the condition is met, becomes ON. If the condition is not met, it is OFF.</p> <ul style="list-style-type: none"> <li>• Constants may be specified as the comparison data.</li> <li>• The most significant bit of a specified constant or variable (register content) is treated as the sign bit during comparison.</li> </ul>
Not equal (NEQ)		LNEQ	
Greater than (GT)		LGT	
Greater than or equal (GE)		LGE	
Less than (LT)		LLT	
Less than or equal (LE)		LLE	

### (3) Arithmetic function instructions

One single arithmetic function instruction forms one to ten steps in the ladder programs. For details on the arithmetic function instructions, see Chapter 2, “ARITHMETIC FUNCTIONS.”

Table 1-3 Arithmetic Function Instructions

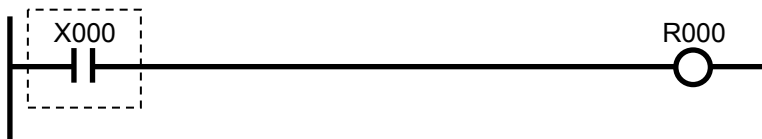
Instruction name	Symbol	Operation code	Function
Arithmetic function		–	Each of a variety of available arithmetic instructions is executed using registers and/or constants specified as word, long word, or floating.

## 1.3.2 a-contacts

Any a-contact becomes ON when the value of a specified register is 1 (ON).

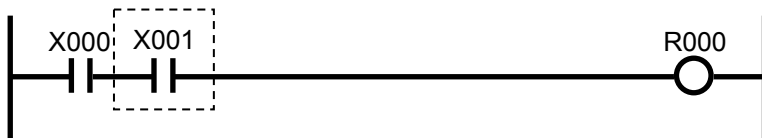
## (1) a-contact start (LD)

The a-contact start instruction becomes ON when the value of a specified register is 1 (ON). For example, in the circuit shown below, if the value of X000 is 1 (ON), R000 will be set to 1 (ON).



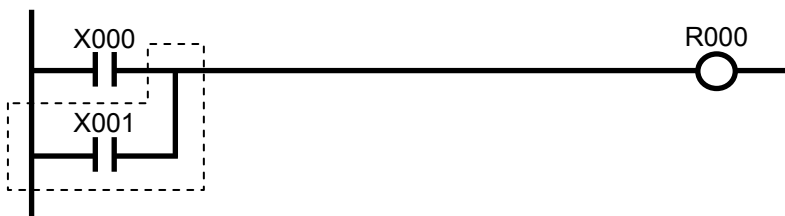
## (2) a-contact series-connection (AND)

The a-contact series-connection instruction performs an AND operation on the value of a specified register and the result of the previous operation performed and, if the AND operation results in 1 (ON), becomes ON. For example, in the circuit shown below, if the values of X000 and X001 are both 1 (ON), R000 will be set to 1 (ON).



## (3) a-contact parallel-connection (LD + ORB)

The a-contact parallel-connection instruction performs an OR operation on the value of a specified register and the result of the previous operation performed and, if the OR operation results in 1 (ON), becomes ON. For example, in the circuit shown below, if the value of X000 or X001 is 1 (ON), R000 will be set to 1 (ON).



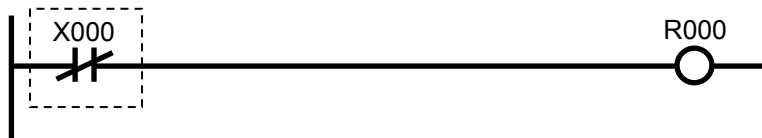
# 1 LADDER INSTRUCTIONS

## 1.3.3 b-contacts

Any b-contact becomes ON when the value of a specified register is 0 (OFF).

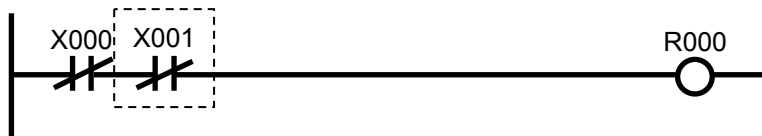
### (1) b-contact start (LDN)

The b-contact start instruction becomes ON when the value of a specified register is 0 (OFF). For example, in the circuit shown below, if the value of X000 is 0 (OFF), R000 will be set to 1 (ON).



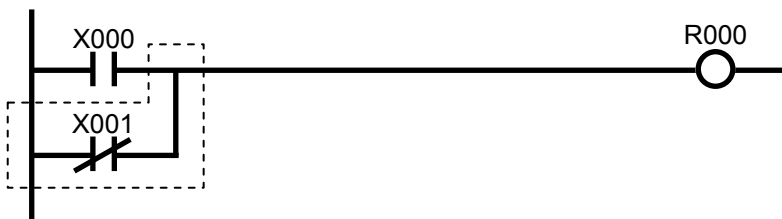
### (2) b-contact series-connection (ANDN)

The b-contact series-connection instruction performs an AND operation on the inverted value of a specified register and the result of the previous operation performed and, if the AND operation results in 1 (ON), becomes ON. For example, in the circuit shown below, if the values of X000 and X001 are both 0 (OFF), R000 will be set to 1 (ON).



### (3) b-contact parallel-connection (LDN + ORB)

The b-contact parallel-connection instruction performs an OR operation on the inverted value of a specified register and the result of the previous operation performed and, if the OR operation results in 1 (ON), becomes ON. For example, in the circuit shown below, if the value of X000 is 1 (ON) or that of X001 is 0 (OFF), then R000 will be set to 1 (ON).



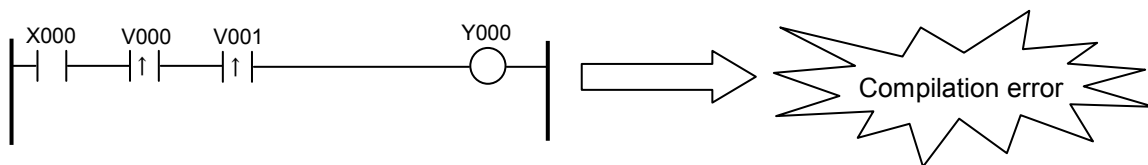
### 1.3.4 Rising-edge and falling-edge contacts

The rising-edge contact ( $\neg \uparrow \neg$ ) and falling-edge contact ( $\neg \downarrow \neg$ ) remain ON only during the sequence cycle in which the rising edge or falling edge (respectively) of the previous operation's result is detected.

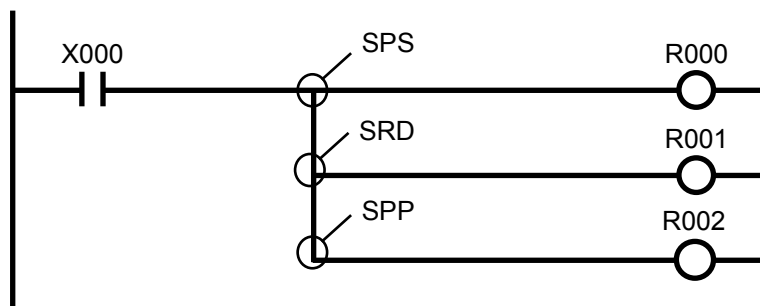
For details, see the description under “V -- edge contacts” in Section 1.5, “Registers.”

Note: Any circuit in which an edge contact precedes another edge contact (as exemplified below) will result in an error during compilation and hence may not be created.

<Example>



### 1.3.5 Operation result push, read, and pop



(1) Operation result push (SPS)

The SPS instruction stores the result of the previous operation performed.

(2) Operation result read (SRD)

The SRD instruction reads the operation result stored by an SPS, SPSAND, or SPSANDN instruction.

(3) Operation result pop (SPP)

The SPP instruction:

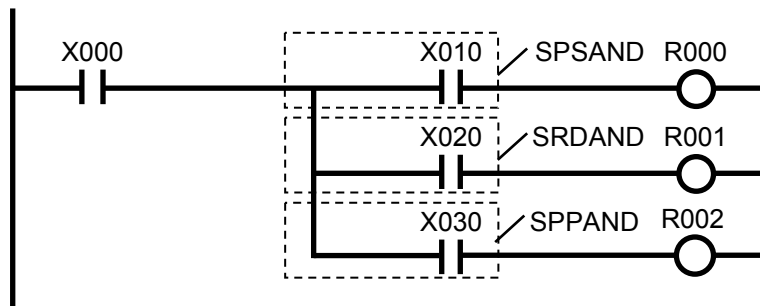
- Reads the operation result stored by an SPS, SPSAND, or SPSANDN instruction; and then
- Clears the stored operation result.

In the above example circuit, when the value of X000 is 1 (ON), all of R000, R001, and R002 will be set to 1 (ON).



## 1 LADDER INSTRUCTIONS

### 1.3.6 Operation result push + a-contact, read + a-contact, and pop + a-contact



(1) Operation result push + a-contact (SPSAND)

The SPSAND instruction stores the result of the previous operation performed and executes the subsequent a-contact with that result.

(2) Operation result read + a-contact (SRDAND)

The SRDAND instruction reads the operation result stored by an SPS, SPSAND, or SPSANDN instruction and executes the subsequent a-contact with the operation result read out.

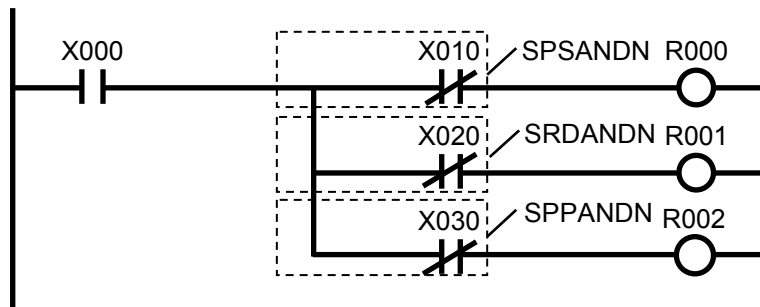
(3) Operation result pop + a-contact (SPPAND)

The SPPAND instruction:

- Reads the operation result stored by an SPS, SPSAND, or SPSANDN instruction and executes the subsequent a-contact with the operation result read out; and then
- Clears the stored operation result.

In the above example circuit, when the values of X000 and X010 are both 1 (ON), R000 will be set to 1 (ON). When the values of X000 and X020 are both 1 (ON), R001 will be set to 1 (ON). When the values of X000 and X030 are both 1 (ON), R002 will be set to 1 (ON).

## 1.3.7 Operation result push + b-contact, read + b-contact, and pop + b-contact



## (1) Operation result push + b-contact (SPSANDN)

The SPSANDN instruction stores the result of the previous operation performed and executes the subsequent b-contact with that result.

## (2) Operation result read + b-contact (SRDANDN)

The SRDANDN instruction reads the operation result stored by an SPS, SPSAND, or SPSANDN instruction and executes the subsequent b-contact with the operation result read out.

## (3) Operation result pop + b-contact (SPPANDN)

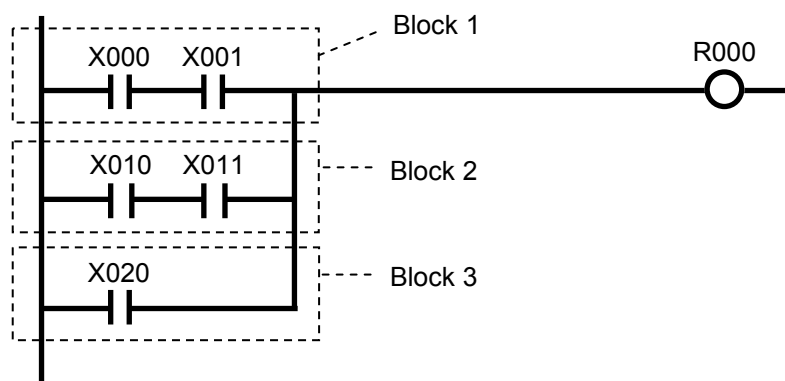
The SPPANDN instruction:

- Reads the operation result stored by an SPS, SPSAND, or SPSANDN instruction and executes the subsequent b-contact with the operation result read out; and then
- Clears the stored operation result.

In the above example circuit, when the value of X000 is 1 (ON) and that of X010 is 0 (OFF), R000 will be set to 1 (ON). When the value of X000 is 1 (ON) and that of X020 is 0 (OFF), R001 will be set to 1 (ON). When the value of X000 is 1 (ON) and that of X030 is 0 (OFF), R002 will be set to 1 (ON).

## 1.3.8 Block union -- parallel connection (ORB)

The ORB instruction performs an OR operation on parallel blocks in a multi-block circuit. For example, in the circuit shown below, when an OR operation on any given two of blocks 1 through 3 results in 1 (ON), R000 will be set to 1 (ON).



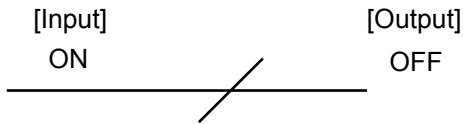
# 1 LADDER INSTRUCTIONS

---

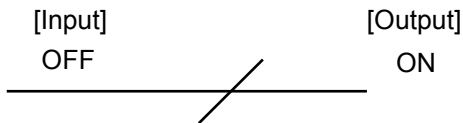
## 1.3.9 NOT

The NOT instruction inverts a given input and outputs the result.

<When the input is 1 (ON)>

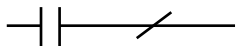


<When the input is 0 (OFF)>

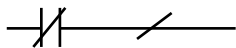


The instruction that can be specified as an input to the NOT instruction is one of the following: the a-contact, b-contact, edge contact, comparison, and parallel connection. A NOT instruction with no input symbol may also be used.

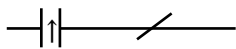
<Inverting the result of an a-contact>



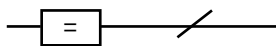
<Inverting the result of a b-contact>



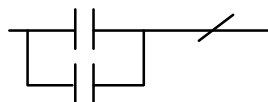
<Inverting the result of an edge contact>



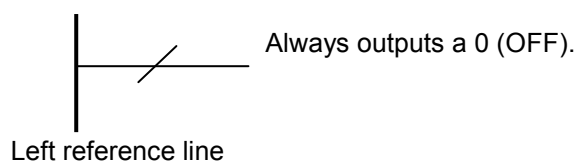
<Inverting the result of a comparison>



<Inverting the result of a parallel connection>



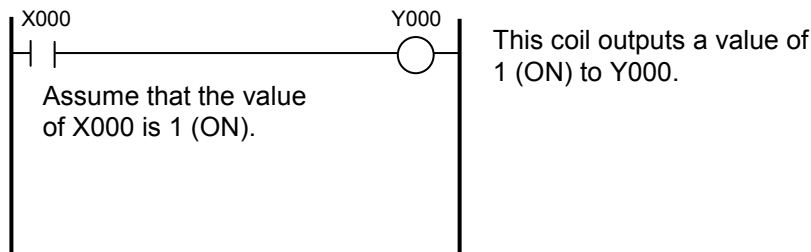
<NOT instruction with no input symbol>



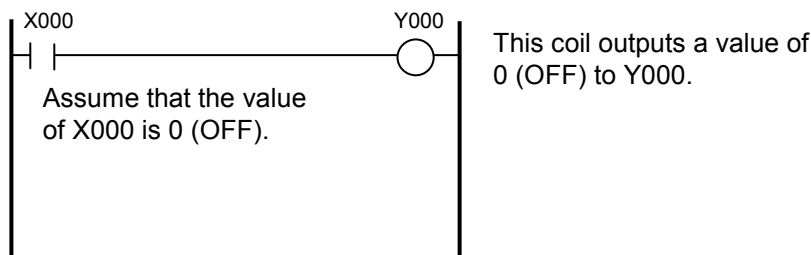
### 1.3.10 Coils

Coils are used to output the result (ON or OFF) of the previous operation performed to a specified register. If a timer (T-register), one-shot timer (U-register), or a counter (C-register) is specified as the coil, it will operate as described under “T -- ON-delay timers,” “U -- one-shot timers,” or “C -- up-down counters,” respectively, in Section 1.5, “Registers.”

<When the condition remains ON right before a coil>



<When the condition remains OFF right before a coil>



### 1.3.11 Set and reset coils

Set coils turn on a given keep relay when the result of the previous operation performed is 1 (ON). The keep relay remains ON thereafter even if the operation result becomes 0 (OFF). Reset coils, on the other hand, turn off the keep relay that has been turned on by a set coil. For more information, see the description under “K -- keep relays” in Section 1.5, “Registers.”

# 1 LADDER INSTRUCTIONS

---

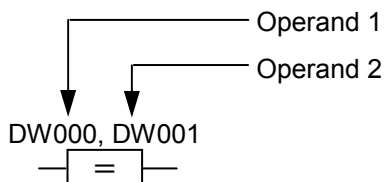
## 1.3.12 Comparison instructions

There are the following six types of comparison instructions available:

- Equal (EQU)
- Not equal (NEQ)
- Greater than (GT)
- Greater than or equal (GE)
- Less than (LT)
- Less than or equal (LE)

### (1) Equal (EQU)

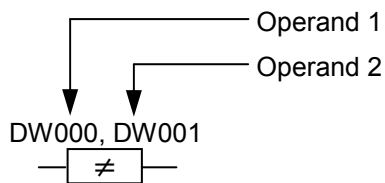
The EQU instruction outputs a value of 1 (ON) if the value of operand 1 equals that of operand 2. Otherwise, it outputs a value of 0 (OFF).



This instruction accepts only a constants word or word register as its operand. (\*)

### (2) Not equal (NEQ)

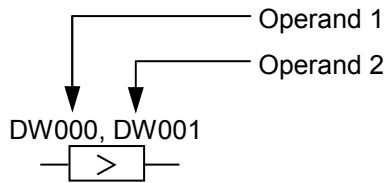
The NEQ instruction outputs a value of 1 (ON) if the value of operand 1 does not equal that of operand 2. Otherwise, it outputs a value of 0 (OFF).



This instruction accepts only a constants word or word register as its operand. (\*)

## (3) Greater than (GT)

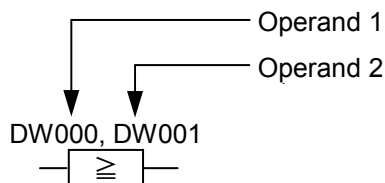
The GT instruction outputs a value of 1 (ON) if the value of operand 1 is greater than that of operand 2. Otherwise, it outputs a value of 0 (OFF).



This instruction accepts only a constants word or word register as its operand. (\*)

## (4) Greater than or equal (GE)

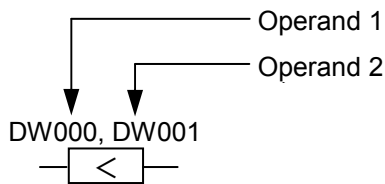
The GE instruction outputs a value of 1 (ON) if the value of operand 1 is greater than or equal to that of operand 2. Otherwise, it outputs a value of 0 (OFF).



This instruction accepts only a constants word or word register as its operand. (\*)

## (5) Less than (LT)

The LT instruction outputs a value of 1 (ON) if the value of operand 1 is less than that of operand 2. Otherwise, it outputs a value of 0 (OFF).



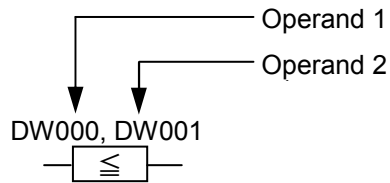
This instruction accepts only a constants word or word register as its operand. (\*)

## 1 LADDER INSTRUCTIONS

---

### (6) Less than or equal (LE)

The LE instruction outputs a value of 1 (ON) if the value of operand 1 is less than or equal to that of operand 2. Otherwise, it outputs a value of 0 (OFF).

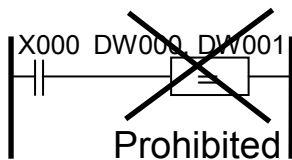


This instruction accepts only a constants word or word register as its operand. (\*)

(\*) Constants (integers) may be specified only as operand 2 and, if specified, must be within the range -32768 to 32767. No registers other than word types of register may be used in the comparison instructions. The values of constants and registers are compared, assuming that they are signed numbers.

Note: No comparison instruction may be connected directly to the right reference line (output).

<Example of a direct connection to the right reference line (output)>



### 1.3.13 Specifying indices in ladder instructions

Of the ladder instructions available, such instructions as a-contact, b-contact, rising-/falling-edge contact, coil, and comparison accept specified indices.

- Indexing using the “base register (index register)” format

Execution register address = base register number + index register content (expressed in units of words)

This indexing method uses as the execution address the location that is identified by the content of the index register relative to the register number of the base register.

The only register type that may be specified as the base register is bit in such instructions as a-contact, b-contact, edge contact, and coil, and is word in comparison instructions.

The index registers specified in the above mentioned types of instructions are all word-type registers.

Example: X020 (FW000)

In this example, if the content of FW000 is the value H0020, the resulting execution address is as follows: X020 + H0020 -> X040.

Note 1: If the content of FW000 is such a value as H0FF0 or H1200, which will result in a value greater than XFFF (i.e., the maximum value of X) when added to the number X020, the normal operation of the instruction using the index is not guaranteed.

Note 2: The execution register address is calculated for instructions other than comparison instructions in the following way:

Execution register address = base register number + index register content

For comparison instructions, it is calculated as follows:

Execution register address = base register number + index register content × H0010 (hexadecimal)

Example: XW000 (FW001)

In this example, if the content of FW001 is the value H0040, the execution register address is calculated as follows:

000 (base register number) + H0040 (index register content) × H0010 = XW400

#### [Restrictions]

If one of the register names listed below is used as the coil, an index may not be specified. Disregarding this rule will result in an input error.

Function name	Register name
ON-delay timer	T
One-shot timer	U
Up-down counter	C
Nesting	N
Process register	P

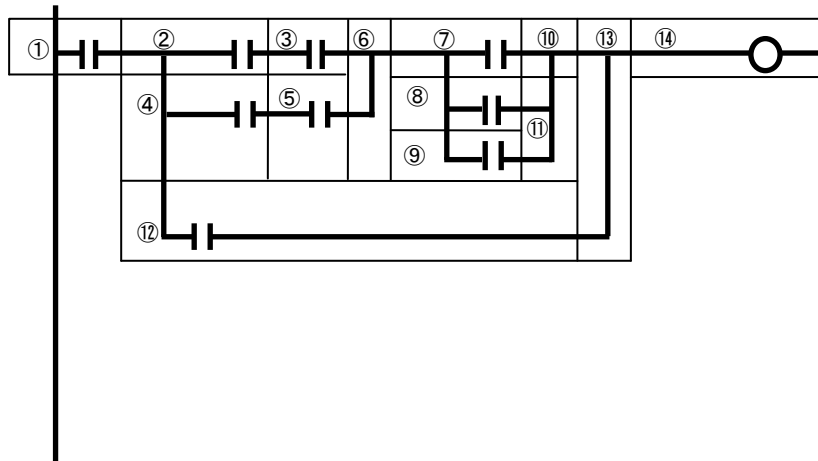


# 1 LADDER INSTRUCTIONS

---

## 1.3.14 Circuits and steps

It should be noted that, although the circuit shown below apparently consists of 14 steps in terms of the instructions, it actually consists of 15 steps because a start instruction is automatically added to the beginning of the circuit during compilation. In addition, the program below is stored and executed in the order of the encircled numbers given. The program portion enclosed in each box forms a step.



### 1.4 Register Statuses at a Reset, Power Recovery, and State Transition between STOP and RUN

	Register name	Status at a reset or power recovery	Status at a transition between STOP and RUN
Bit registers	T-/U-contact and coil	Cleared	Remaining unchanged
	C-contact and coil	Remaining unchanged	Remaining unchanged
	K	Remaining unchanged	Remaining unchanged
	S	Initialized (with initial value)	Remaining unchanged
	X, Y, R, M, A, N, P, E, V, Z, J, Q, LB, LR, LV	Cleared	Remaining unchanged
Word and long-word registers	T-/U-set value	Remaining unchanged	Remaining unchanged
	T-/U-count value	Cleared	Remaining unchanged
	C-set value and count value	Remaining unchanged	Remaining unchanged
	FW, DW, BD, LX, LM, LG	Remaining unchanged	Remaining unchanged
	LW, LL, LF, IW, OW, HH	Cleared	Remaining unchanged

# 1 LADDER INSTRUCTIONS








## 1.5 Registers

### 1.5.1 Registers usable in ladder instructions

Table 1-4 is a list of all registers usable in ladder instructions.

Table 1-4 Usable Registers







(1/2)

	Function name	Register name	Ladder symbols						Status after reset or power recovery	
										
I/O	External input	X	●	–	–	–	●	●	Cleared	
	External output	Y	●	–	●	–	●	●	Cleared	
Internal auxiliary functions	Internal register	R	●	–	●	–	●	●	Cleared	
	Extension internal register	M, A	●	–	●	–	●	●	Cleared	
	Keep relay	K	●	–	–	●	●	●	Remaining unchanged	
	ON-delay timer	Contact, coil	T	●	–	●	–	●	●	Cleared
		Set value	TS	–	–	–	–	●	●	Remaining unchanged
		Count value	TC	–	–	–	–	●	●	Cleared
	One-shot timer	Contact, coil	U	●	–	●	–	●	●	Cleared
		Set value	US	–	–	–	–	●	●	Remaining unchanged
		Count value	UC	–	–	–	–	●	●	Cleared
	Up-down counter	Contact, coil	CU	–	–	●	–	–	–	Remaining unchanged
			CD	–	–	●	–	–	–	Remaining unchanged
			CR	–	–	●	–	–	–	Remaining unchanged
			C0	●	–	–	–	●	●	Remaining unchanged
		Count value	CS	–	–	–	–	●	●	Remaining unchanged
		Count value	CC	–	–	–	–	●	●	Remaining unchanged
	Global link register	G	●	–	●	–	●	●	Cleared	
	Nesting coil	NM	–	–	●	–	–	–	Cleared	
		NZ	–	–	●	–	–	–	Cleared	
		N0	●	–	–	–	●	●	Cleared	
	Process register	P	●	–	●	–	●	●	Cleared	
	Event register	E	●	–	●	–	●	●	Cleared	
	Edge contact	V	–	●	–	–	●	●	Cleared	
Zee register	Z	●	–	●	–	●	●	Cleared		
System register	S	●	–	–	–	●	●	Initialized		
Shared-data register between HI-FLOW and ladder	J	●	–	–	–	●	●	Cleared		
	Q	●	–	●	–	●	●	Cleared		

●: Usable register.  
–: Non-usable register.

Table 1-4 Usable Registers

(2/2)

	Function name	Register name	Ladder symbols						Status after reset or power recovery
									
Internal auxiliary functions	Register between HI-FLOW processes	HH	–	–	–	–	–	–	Cleared
	Extension internal register	LB	●	–	●	–	●	●	Cleared
	Converter-specific internal register	LR	●	–	●	–	●	●	Cleared
	Converter-specific edge contact register	LV	–	●	–	–	●	●	Cleared
	Input register (reserved for future use)	IW	–	–	–	–	●	●	Cleared
	Output register (reserved for future use)	OW	–	–	–	–	●	●	Cleared
	Internal register	BD	–	–	–	–	–	●	Remaining unchanged
		BW (*)	–	–	–	–	–	●	Depending on BD
	Function data register	DW	–	–	–	–	●	●	Remaining unchanged
	Function work register	FW	–	–	–	–	●	●	Remaining unchanged
	Extension function work register	LW	–	–	–	–	●	●	Cleared
	Long-word work register	LL	–	–	–	–	–	●	Cleared
	Single-precision floating-point work register	LF	–	–	–	–	–	●	Cleared
	Backup word work register	LX	–	–	–	–	●	●	Remaining unchanged
	Backup long-word work register	LM	–	–	–	–	–	●	Remaining unchanged
Backup single-precision floating-point work register	LG	–	–	–	–	–	●	Remaining unchanged	

●: Usable register.

–: Non-usable register.

(\*) Accessed by indirect addressing.

# 1 LADDER INSTRUCTIONS

## 1.5.2 Register numbers

Table 1-5 is a list of all register numbers that can be used in ladder programs. As shown, the range of usable register numbers depends on the types of registers accessed by their generic register names.

Table 1-5 Register Numbers

(1/2)

No.	Register name	Register types accessed			
		Bit	Word	Long-word	Single-precision floating-point
1	X	X000 to XFFF	XW000 to XWFF0	XL000 to XLFE0	–
2	Y	Y000 to YFFF	YW000 to YWFF0	YL000 to YLFE0	–
3	R	R000 to RFFF	RW000 to RWFF0	RL000 to RLFE0	–
4	M	M000 to MFFF	MW000 to MWFF0	ML000 to MLFE0	–
5	A	A000 to AFFF	AW000 to AWFF0	AL000 to ALFE0	–
6	K	K000 to KFFF	KW000 to KWFF0	KL000 to KLFE0	–
7	T	T000 to T7FF	TW000 to TW7F0	TL000 to TL7E0	–
8	TS	–	TS000 to TS1FF	–	–
9	TC	–	TC000 to TC1FF	–	–
10	U	U000 to UOFF	UW000 to UW0F0	UL000 to UL0E0	–
11	US	–	US000 to US0FF	–	–
12	UC	–	UC000 to UC0FF	–	–
13	CU	CU00 to CUFF	–	–	–
14	CD	CD00 to CDFF	–	–	–
15	CR	CR00 to CRFF	–	–	–
16	C0	C000 to C0FF	CW000 to CW0F0	CL000 to CL0E0	–
17	CS	–	CS000 to CS0FF	–	–
18	CC	–	CC000 to CC0FF	–	–
19	G	G000 to GFFF	GW000 to GWFF0	GL000 to GLFE0	–
20	NM	NM01 to NMFF	–	–	–
21	NZ	NZ01 to NZFF	–	–	–
22	N0	N001 to N0FF	NW000 to NW0F0	NL000 to NL0E0	–
23	P	P001 to P080	PW000 to PW080	PL000 to PL060	–
24	E	E000 to EFFF	EW000 to EWFF0	EL000 to ELFE0	–
25	V	V000 to VFFF	VW000 to VWFF0	VL000 to VLFE0	–
26	Z	Z000 to Z3FF	ZW000 to ZW3F0	ZL000 to ZL3E0	–
27	S	S000 to SBFF	SW000 to SWBF0	SL000 to SLBE0	–

–: Not accessible.

Table 1-5 Register Numbers

(2/2)

No.	Register name	Register types accessed			
		Bit	Word	Long-word	Single-precision floating-point
28	J	J000 to JFFF	JW000 to JWFFF0	JL000 to JLFE0	–
29	Q	Q000 to QFFF	QW000 to QWFFF0	QL000 to QLFE0	–
30	LB	LB0000 to LBFFFF	LBW0000 to LBWFFF0	LBL0000 to LBLFFE0	–
31	LR	LR0000 to LR0FFF	LRW0000 to LRW0FFF0	LRL0000 to LRL0FE0	–
32	LV	LV0000 to LV0FFF	LVW0000 to LVW0FFF0	LVL0000 to LVL0FE0	–
33	IW	–	IW000 to IWFFF	IL000 to ILFFE	–
34	OW	–	OW000 to OWFFF	OL000 to OLFFE	–
35	BD	–	–	BD000 to BD1FE	–
36	BW (*)	–	BW000 to BW1FE	BL000 to BL1FE	–
37	DW	–	DW000 to DWFFF	DL000 to DLFFE	–
38	FW	–	FW000 to FWBFF	FL000 to FLBFE	–
39	LW	–	LWW0000 to LWWFFFF	LWL0000 to LWLFFFE	–
40	LL	–	–	LLL0000 to LLL1FFF	–
41	LF	–	–	–	LF0000 to LF1FFF
42	LX	–	LXW0000 to LXW3FFF	LXL0000 to LXL3FFE	–
43	LM	–	–	LML0000 to LML1FFF	–
44	LG	–	–	–	LG0000 to LG1FFF

–: Not accessible.

(\*) Accessed by indirect addressing.

# X, Y EXTERNAL INPUT AND OUTPUT

Range of numbers	000 to FFF
Input/output range of remote I/O	000 to 7FF

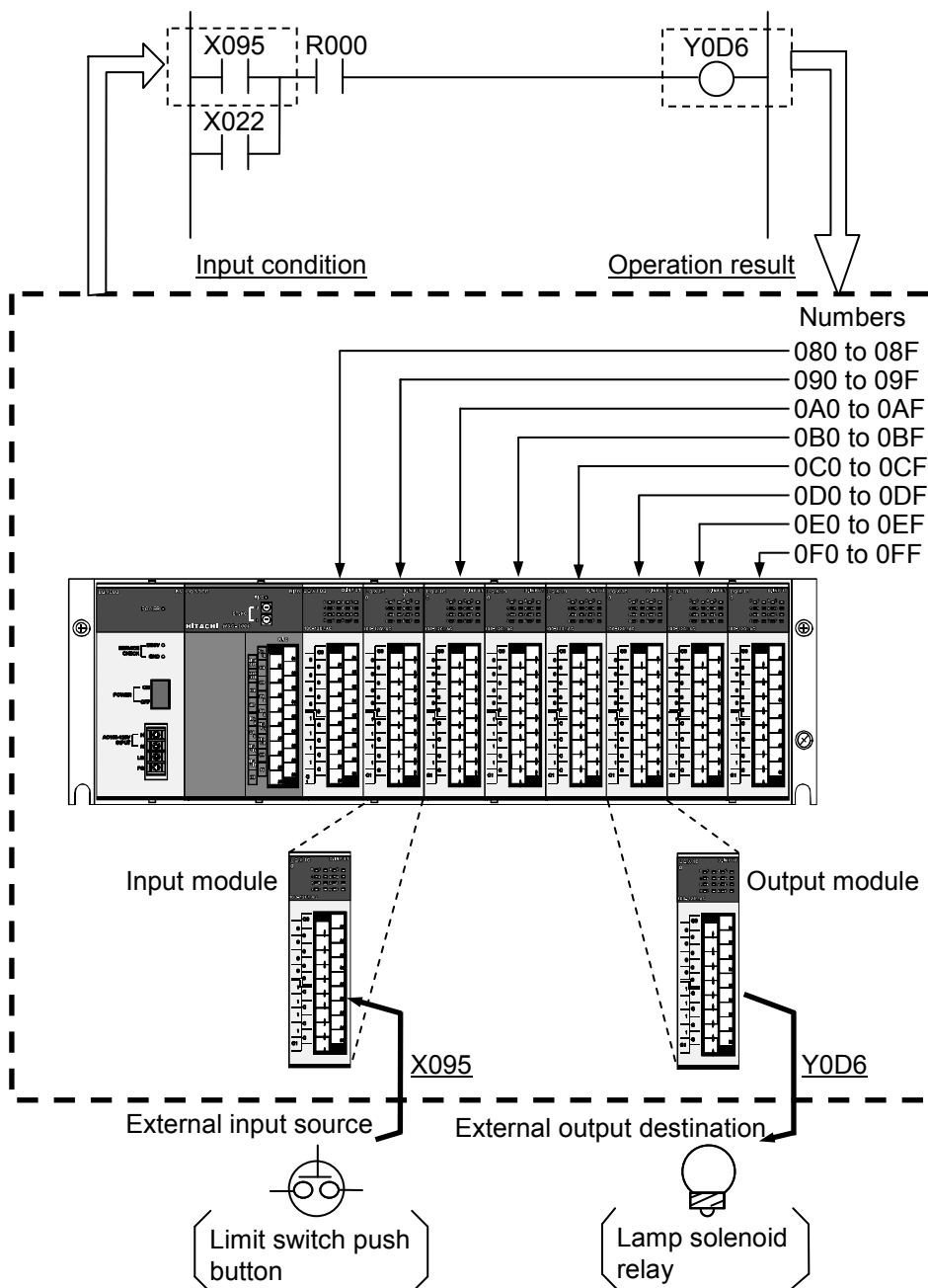
Each external input/output register is used to input or output signals via the external input or output module connected to the PCs.

X: Receive input signals from external sources via the input module.

Y: Send operation results from the ladder program to external destinations via the output module.

- Usage example

The circuit shown below outputs a signal to the Y0D6 of the output module when the X095 of the input module contains a value of 1 (ON).



**This Page Intentionally Left Blank**

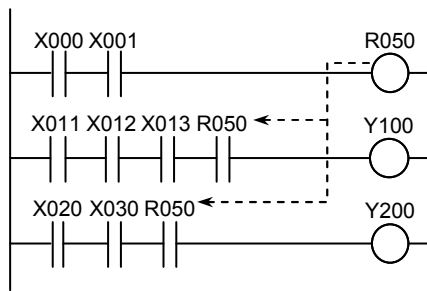


# R, M, A, LB INTERNAL REGISTERS

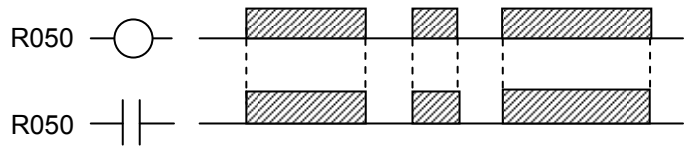
Register name	R, M, A	LB
Range of numbers	000 to FFF	0000 to FFFF

These internal registers are used to pass operation results between ladder instructions. When the coil of a specified internal register becomes ON, its contact will also become ON simultaneously. In contrast, when the former becomes OFF, the latter will also become OFF simultaneously.

● Usage example



● Timing chart



All of the R, M, A, and LB registers are functionally the same.

**This Page Intentionally Left Blank**

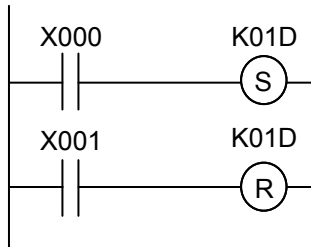
# K KEEP RELAYS

Range of numbers	000 to FFF
Settling-pulse width	At least 1 sequence cycle
When a set and a reset signal are input simultaneously:	Whichever coil comes later in the program has priority.

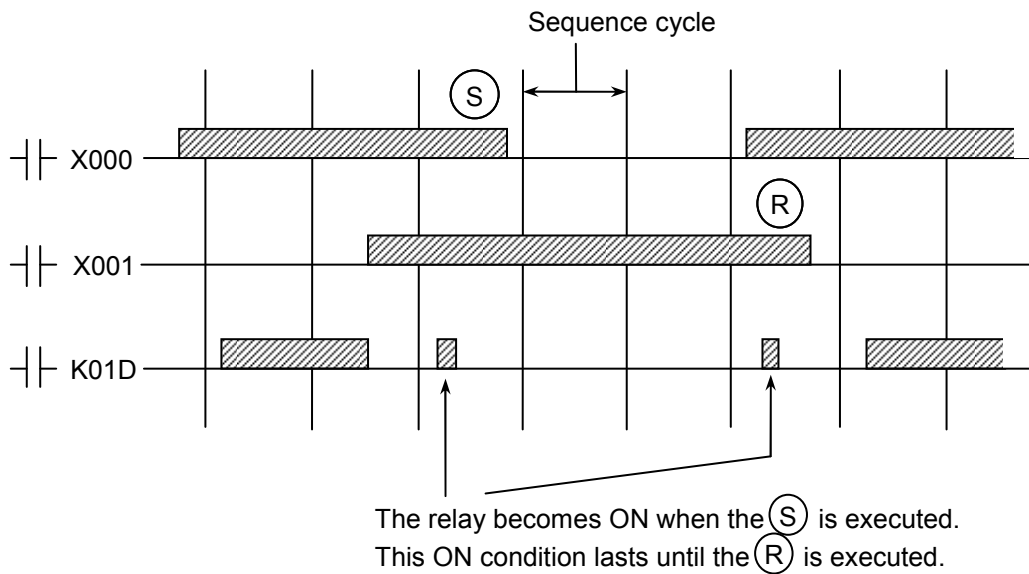
A keep relay has its contact closed (ON) when its set coil is energized (ON). This closed (ON) condition is maintained until its reset coil is energized (ON). This is true even when the power to the keep relay is OFF. If the set and the reset coil are energized at the same time, whichever coil comes later in the program has priority.

## (1) Reset-first circuit

- Usage example

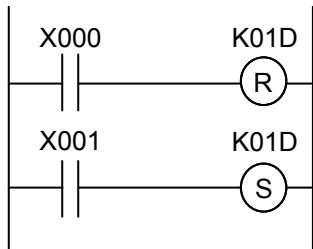


- Timing chart

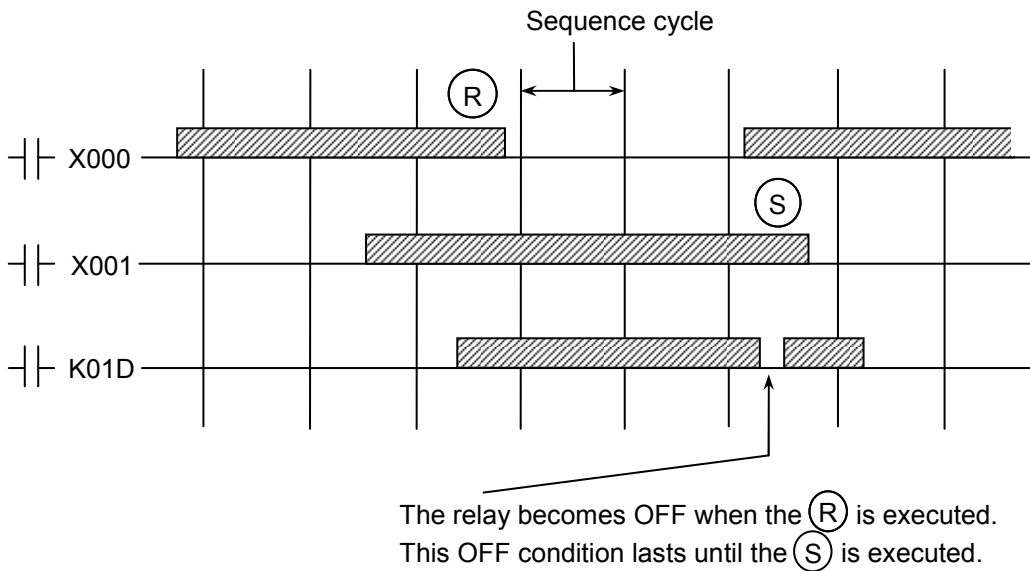


(2) Set-first circuit

- Usage example



- Timing chart



## T ON-DELAY TIMERS

	100-ms timer	10-ms timer (used by setting)
Range of numbers (*1)	000 to 1FF or 000 to 7FF	000 to 00F
Set value	0 to 9999 (0.0 to 999.9 seconds)	0 to 9999 (0.0 to 99.99 seconds)
Error	At least 100 ms + 1 sequence cycle	At least 10 ms + 1 sequence cycle
Settling-pulse width (*2)	At least 100 ms	At least 10 ms

(\*1) The range of numbers that can be used in 100-ms timer specifications varies depending on the module revisions of LPU modules used. The table below shows the ranges of usable numbers and the corresponding LPU module revisions.

LPU module model	Module revision	Range of usable numbers	Remarks
LQP510	[A] to [H]	000 to 1FF	–
LQP510-Z	[A] to [H]		
LQP710-Z	[A] to [E]		
LQP510	[I] or later	000 to 7FF	Of these, the numbers 200 through 7FF are usable only in S10V Ladder Chart System (model S-7895-02) of version 01-09 or later.
LQP510-Z	[I] or later		
LQP710-Z	[F] or later		

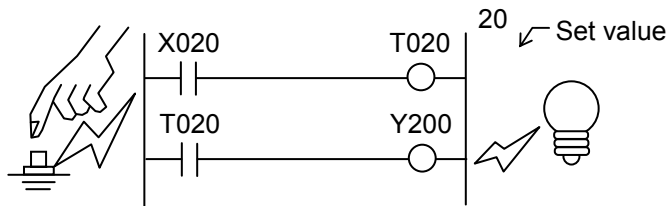
(\*2) The settling-pulse width stands for the minimum time period in which the contact to energize the coil of an ON-delay timer must remain closed (ON).

The contact of an ON-delay timer is not closed (ON) until the delay after the energization of its coil, which is specified by the set value, has elapsed. This set value may be specified in units of 0.1 second in the range 0.0 to 999.9.

The first 16 registers T000 through T00F can be used as 10-ms timers by settings.

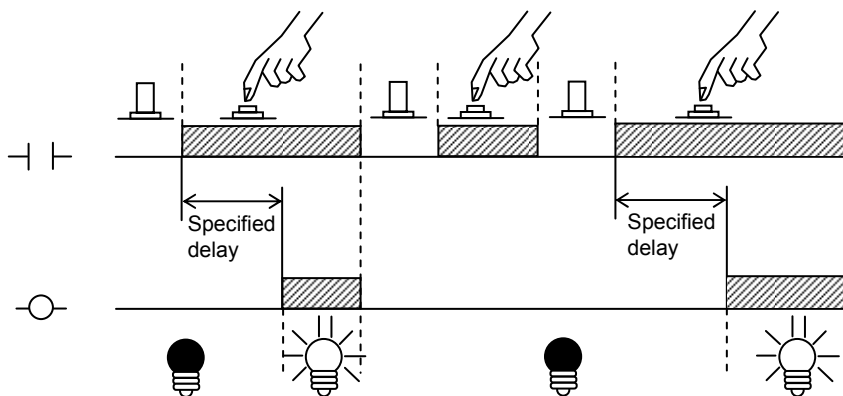
These settings can be made by choosing [Utility] – [PCs edition] – [Change capacity] in the S10V programming software product called the S10V Ladder Chart System (model S-7895-02). For information on how to operate the ladder chart system, refer to the “SOFTWARE MANUAL OPERATION S10V LADDER CHART For Windows® (Manual number SVE-3-131).”

● Usage example



In the circuit shown left, the lamp (Y200) is not lit until the delay specified by the set value (two seconds) has elapsed after the push of the push button (X020). Once the lamp is lit by holding down the push button after the push, it goes out when the push button is released.

● Timing chart



Notes:

- If the coil is de-energized before the specified delay expires, the contact will not be closed (ON). In this case, when the coil is energized again, the timer will start counting up from 0.
- The count is incremented from 0 to 65535. When the count reaches 65535, it is reset for counting from 0 again.
- Where the ON-delay timer is used as a 100-ms timer, the detection of its coil's ON/OFF condition is performed at 100-ms intervals asynchronously with the ladder circuit's execution cycle, also called the sequence cycle (with 10-ms timers, it is done at 10-ms intervals). If the coil's ON condition lasts for less than 100 milliseconds, it will not be detected, resulting in no operation of the ON-delay timer. To ensure the operation of the timer, create a ladder circuit in a way that maintains the ON condition of the coil for at least 100 milliseconds.

# U ONE-SHOT TIMERS

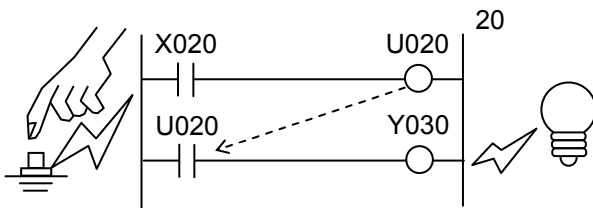
Range of numbers	000 to 0FF
Set value	0 to 9999 (0.0 to 999.9 seconds)
Error	At least 100 ms + 1 sequence cycle
Settling-pulse width (*)	At least 100 ms

(\*) The settling-pulse width stands for the minimum time period in which the contact to energize the coil of a one-shot timer must remain closed (ON).

When the coil of a one-shot timer is energized, its contact is closed (ON).

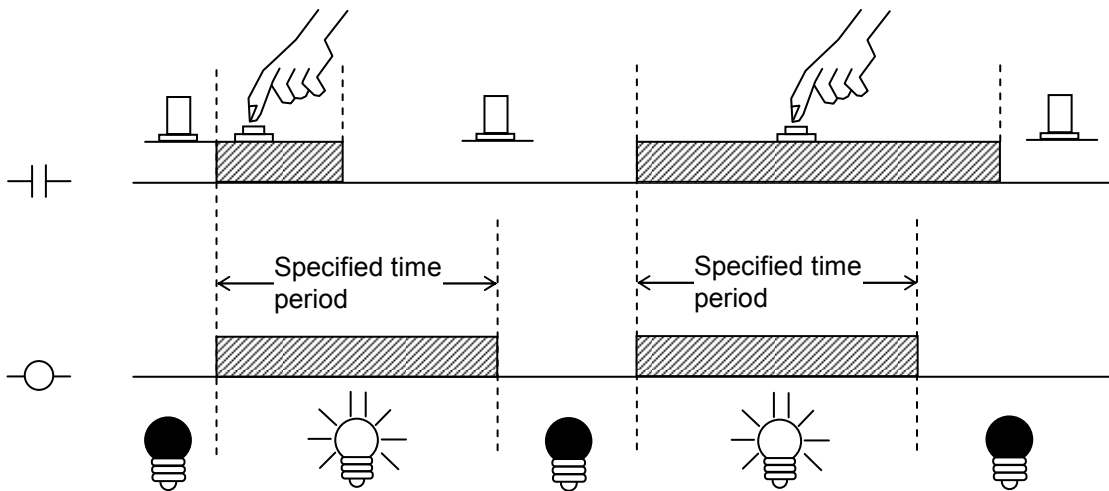
This ON condition then lasts for the time period specified by the set value, which can be specified in units of 0.1 second in the range 0.0 to 999.9 (0 to 9999 if it is specified from the PC used as a S10V base system).

### ● Usage example



In the circuit shown left, the lamp (Y030) is lit by pushing the push button. Once the lamp is lit, it stays ON for the time period specified by the set value (2 seconds).

### ● Timing chart






Notes:

- Detection of a one-shot coil's ON/OFF condition is performed at 100-ms intervals asynchronously with the ladder circuit's execution cycle, also called the sequence cycle. If the coil's ON condition lasts for less than 100 milliseconds, it will not be detected, resulting in no operation of the one-shot timer. To ensure the operation of the timer, create a ladder circuit in a way that maintains the ON condition of the coil for at least 100 milliseconds.
- Even if the coil of a one-shot timer is de-energized (OFF) before the time period specified by the set value expires, its contact (U-register) remains closed (ON) until it expires. That is, the one-shot timer continues counting up until the specified time period expires, regardless of the current ON/OFF condition of its coil.



## C UP-DOWN COUNTERS

Range of numbers		CU	00 to FF
		CD	
		CR	
Set value		C0	
Set value	Count in the range 0 to 9999		
Settling-pulse width (*)	At least 1 sequence cycle		
When a set and a reset signal are input simultaneously:	Reset having priority		
In the event of a power outage:	Non-volatile		

(\*) The settling-pulse width stands for the minimum time period in which the contact to energize the coil of an up or down counter or a reset coil must remain closed (ON).

An up-down counter is a combination of an up counter (CU) and a down counter (CD). Its count is incremented every time the up counter's coil is energized, and decremented every time the down counter's coil is energized.

The counter contact (C0) is closed (ON) when the count exceeds the set value. The reset coil (CR) is used to clear the count and open the counter contact (OFF).

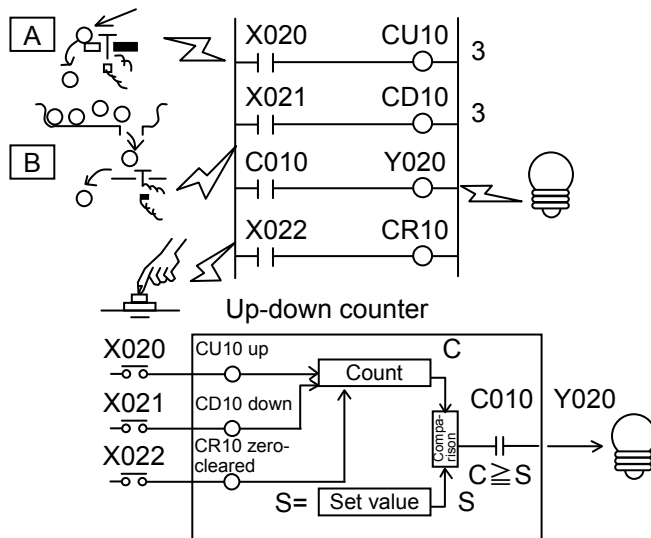
CU: Up counter

CD: Down counter

CR: Reset coil

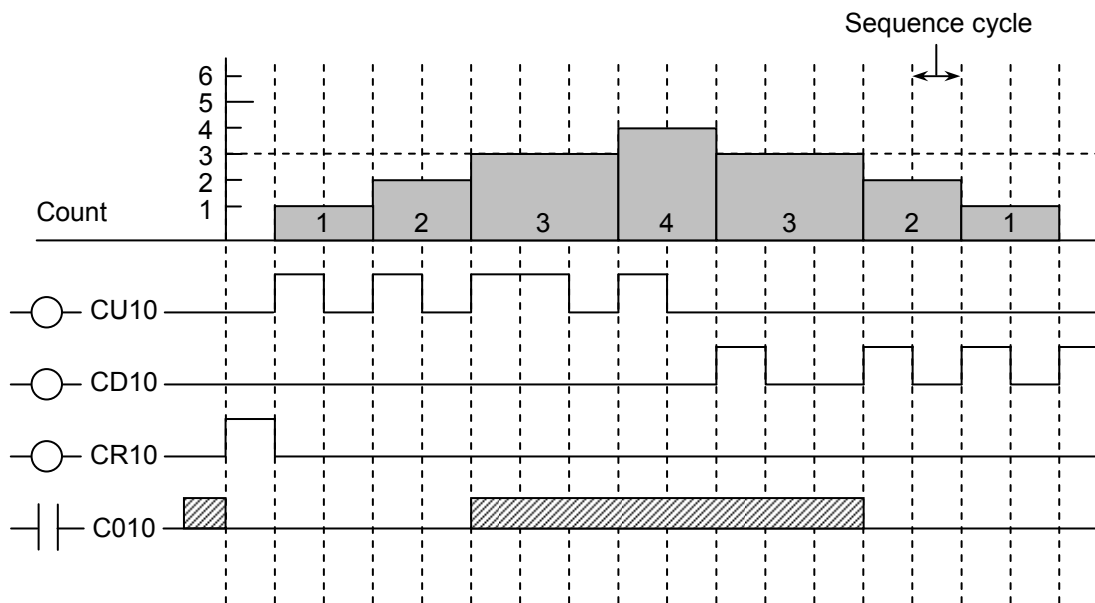
C0: Counter contact

### ● Usage example



- In the circuit shown left, the switch **A** (X020) is used to count the number of balls that drop into the basket, and the switch **B** (X021) is used to count the balls that drop from the basket; that is, the circuit is used to count the balls that are currently in the basket.
- When the number of balls in the basket reaches 3 or greater, the lamp (Y020) is lit. When the push button (X022) is pushed, the count is zero-cleared and the lamp goes out.

● Timing chart



Notes:

- The up counter continues counting even when its count exceeds the set value. When the count overflows (i.e., it exceeds the value 0xFFFF), the counter starts counting from 0 again, the closed counter contact being opened (OFF).
- The down counter stops counting when its count reaches 0.
- If the power to the PLC is turned on during the sequence cycle in which the counter coil is being energized from OFF to ON level, the counter coil will be energized normally, but the count may not be incremented. To avoid this, observe the following rules:
  - ① Shut off the power to the PLC only when the coil is in a stable condition. Never change the coil's condition from OFF to ON during the power shut-off operation.
  - ② Use an uninterruptible power supply (UPS) for protection against power outages.

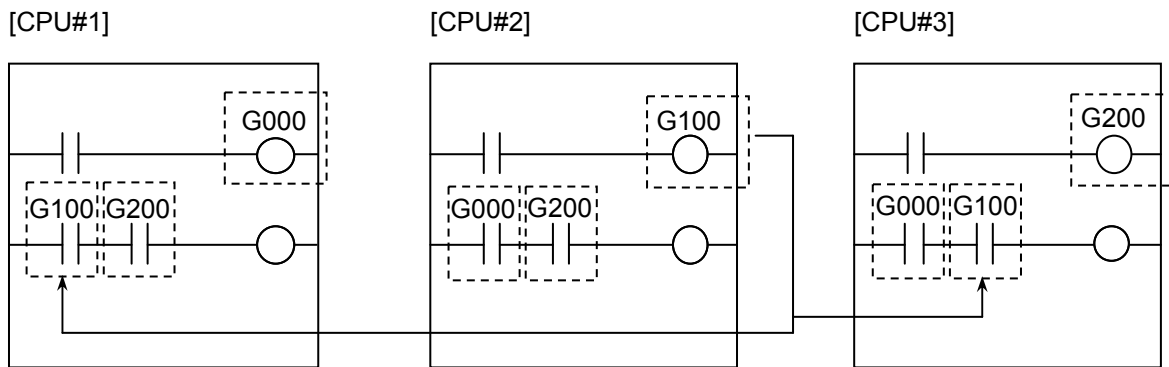
## G GLOBAL LINK REGISTERS

Range of numbers	000 to FFF
------------------	------------

A global link register (G-register) is used in cases where inter-CPU link modules (option) are installed. These registers are provided as a means of exchanging interlock information between the interlocked CPUs.

When the coil of a global link register is energized (ON) (or de-energized [OFF]) in one such CPU, the contact(s) with the same register number are closed (ON) (or opened [OFF]) in the other such CPU(s).

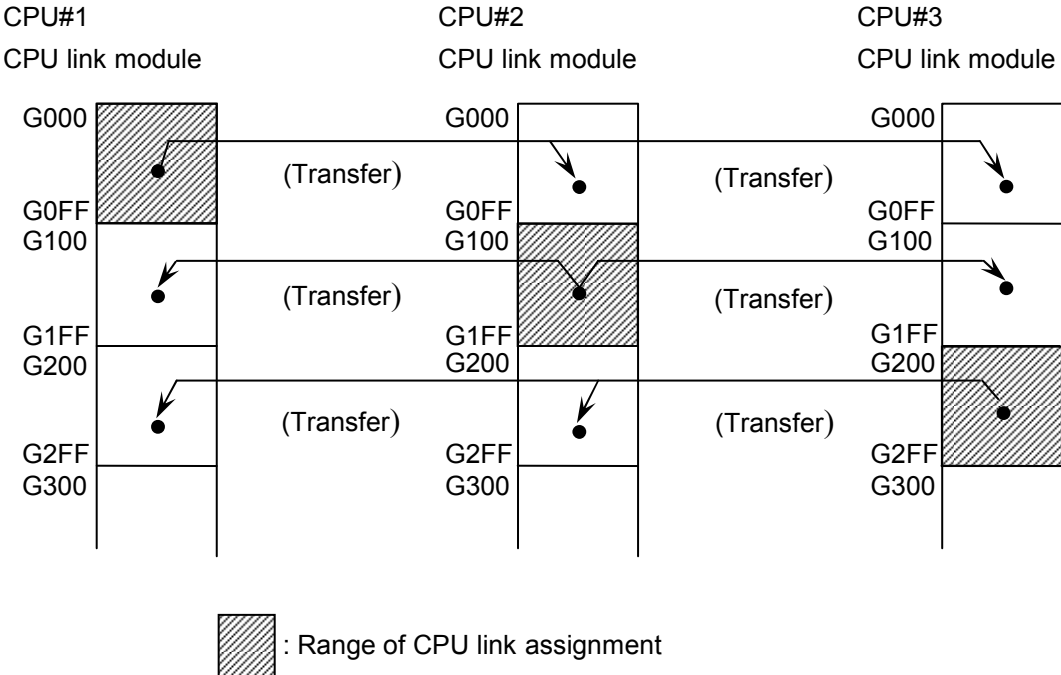
- Usage example (where inter-CPU link modules are used)



### [Operation]

- When the coil of G000 is energized (ON) (or de-energized [OFF]) in CPU#1, the a-contacts with the same number G000 are closed (ON) (or opened [OFF]) in both CPU#2 and CPU#3.
- When the coil of G100 is energized (ON) (or de-energized [OFF]) in CPU#2, the a-contacts with the same number G100 are closed (ON) (or opened [OFF]) in both CPU#1 and CPU#3.
- When the coil of G200 is energized (ON) (or de-energized [OFF]) in CPU#3, the a-contacts with the same number G200 are closed (ON) (or opened [OFF]) in both CPU#1 and CPU#2.

● Operation of the CPU link modules



- The contents of the G-register area ranging from G000 to G0FF in CPU#1 are transferred to the same register areas in CPU#2 and CPU#3.
- The contents of the G-register area ranging from G100 to G1FF in CPU#2 are transferred to the same register areas in CPU#1 and CPU#3.
- The contents of the G-register area ranging from G200 to G2FF in CPU#3 are transferred to the same register areas in CPU#1 and CPU#2.

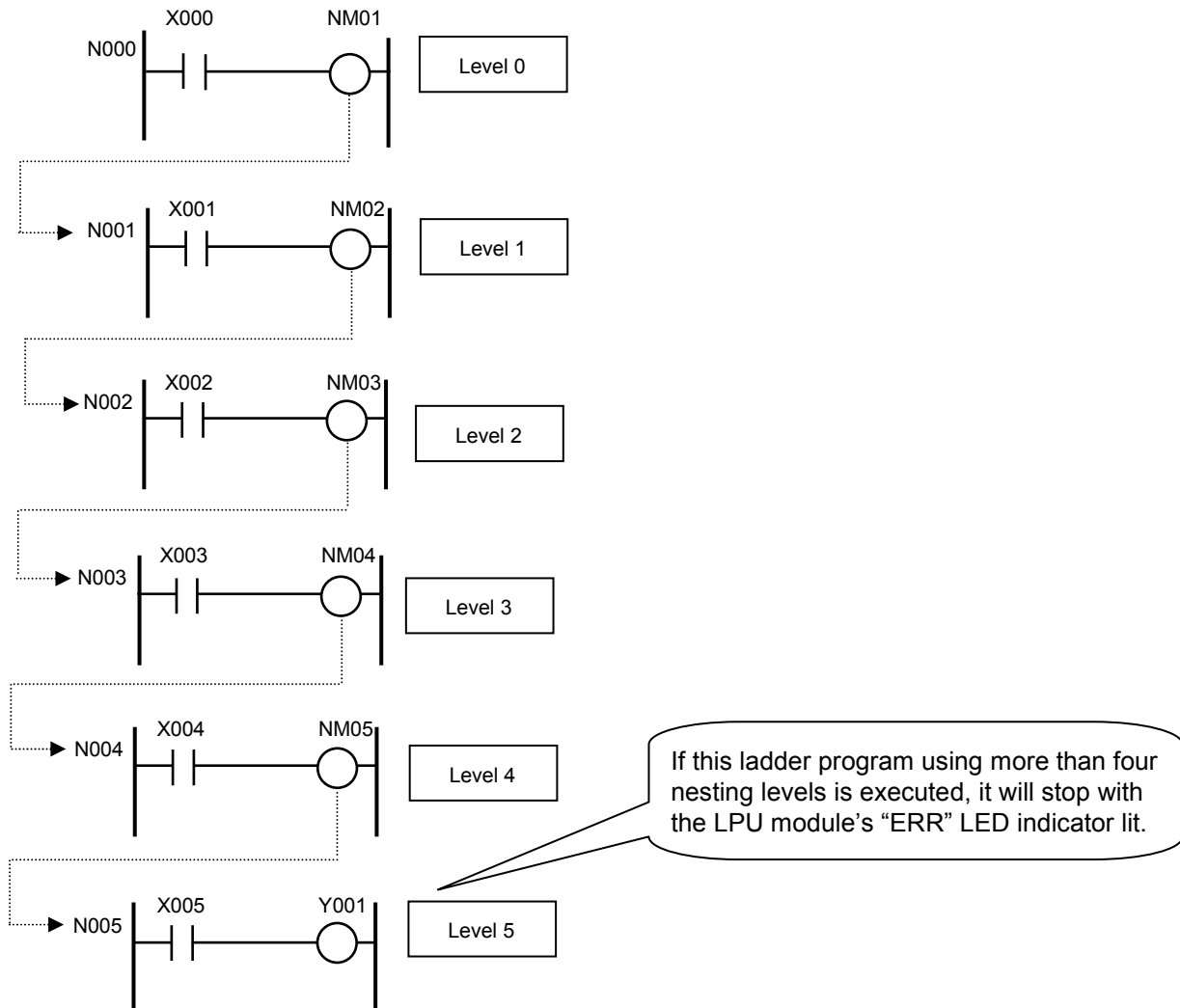
# N NESTING COILS

Range of numbers	000 to 0FF
Maximum number of nesting levels usable	4

Nesting coils serve as a means of dividing one single sequence program into as many smaller modules as the number of plants to be controlled by that program. Of the available nesting coils, N000 is called the master N-coil, and each of N001 through N0FF is called a sub-N coil.

The master N coil is initiated in each sequence cycle when the LPU module is in RUN state. The sub-N coil, on the other hand, is invoked by the master N coil or another sub-N coil. Nesting coils can be chosen from among the two groups of coils: the master control (NM) coils, each of which de-energizes the other subsequent coil(s) in the same program when making a transition from ON to OFF state, and the zone control (NZ) coils each of which maintains the previously initiated condition of the other subsequent coil(s) in the same program when making a transition from ON to OFF state. In addition, nesting coils (N-coils) can be nested in up to four levels. Any nesting in more than four levels will result in an error at the time of a program run.

<An example of master control using more than four levels of nesting>



Note: The master N coil (N-coil number 000) may not be used as a contact or coil. (If so used, the master N coil will cause an input error.)

(1) Master control (NM)

As described in “<Operation modes of master control>” below, the master control runs in two different modes. These operation modes can be switched by selecting [Utility] – [PCs edition] – [Change capacity] in the S10V Ladder Chart System (model S-7895-02). For information on how to operate the ladder chart system, refer to the “SOFTWARE MANUAL OPERATION S10V LADDER CHART For Windows® (Manual number SVE-3-131).”

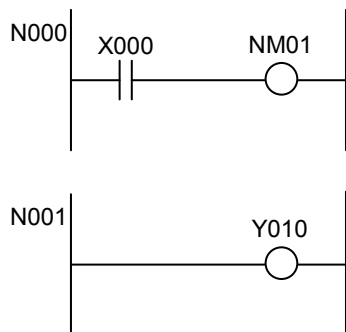
<Operation modes of master control>

Normal mode: In this mode, when the master control coil makes a transition from ON to OFF state, the rising-(/falling-)edge contact(s) and normal coil(s) used as N-coils after the master control coil in the same program become OFF.

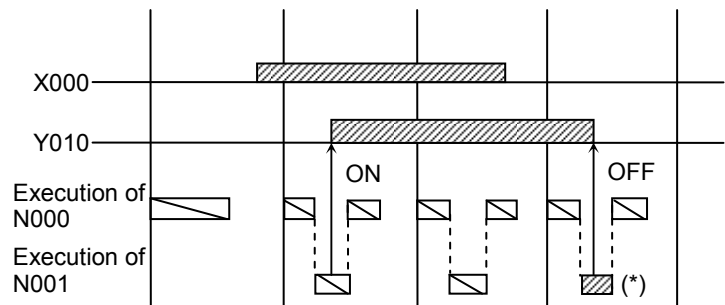
0-output mode: In this mode, the same effect as in normal mode is produced at the time of the master control coil’s making a transition from ON to OFF state, plus any set and reset coils become OFF.

● Usage example

The following program exerts master control over N001 from N000:



● Timing chart

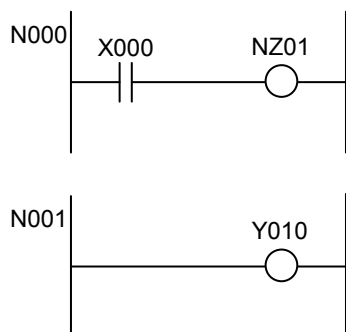


(\*) This period of execution de-energizes the coil used in N001 when NM01 makes a transition from ON to OFF. The types of coils de-energized in master control vary depending on the set operation mode of master control.

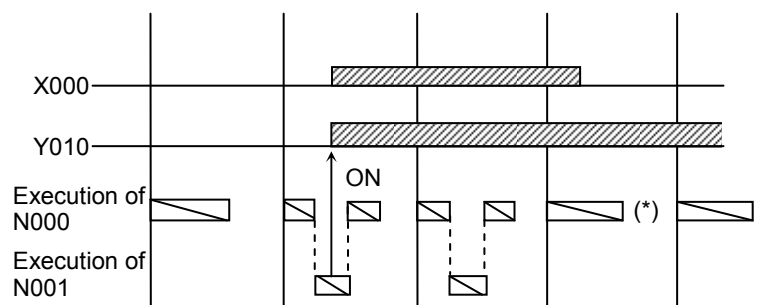
(2) Zone control (NZ)

● Usage example

The following program exerts zone control over N001 from N000:

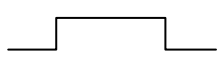


● Timing chart



(\*) Unlike the case of master control, this period of execution under zone control maintains the previously initiated condition of the coil used with N001 when NZ01 makes a transition from ON to OFF state.

# P PROCESS REGISTERS

Range of numbers	001 to 080
Initiation method	Level start 

A process register is used to initiate a program written in such a computer language as C or assembly language (hereinafter called a task) from the ladder program.

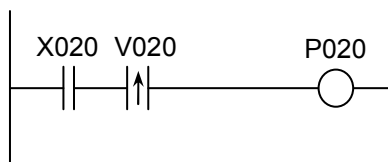
When energized, the coil specified with a process register (P-coil) triggers the execution of the identified task, which is identified by its specified number (task number). The task execution is done on the CMU module.

### <Assignment to process registers>

Classification	Number	Name	Description
User-created	P001	Initializing task	A task that is executed whenever the LPU module is manually reset or power-on reset. Assign a system initialization program to this number.
	P002 to P080	User task	Assign user-written programs to these numbers.

### (1) Initiating only once when a given contact is closed (ON)

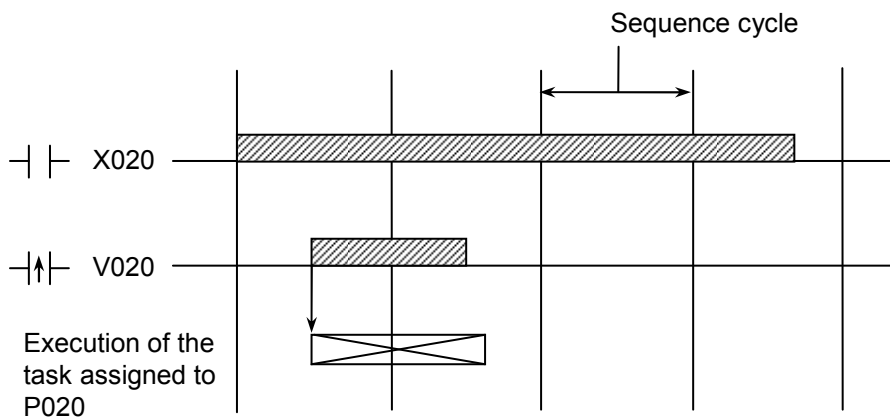
#### ● Usage example



When X020 makes a transition from OFF to ON state, the task assigned to P020 is initiated only once.

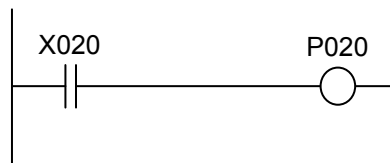
P020  
 ↑  
 Task number in hexadecimal. In this example, the task numbered 32 is initiated.

#### ● Timing chart



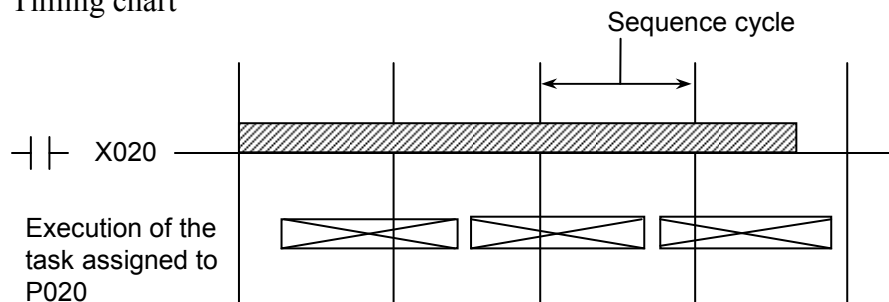
(2) Initiating repeatedly while a given contact is closed (ON)

● Usage example



As long as X020 remains closed (ON), the task assigned to P020 is initiated repeatedly.

● Timing chart



Notes:

- If an attempt is made to energize the P-coil for which a task is not registered, nothing will be executed.
- The tasks that can be initiated with a specified P-coil are those whose task numbers are in the range 1 to 128 (001 to 080 in terms of process register numbers). The tasks numbered 129 through 255 cannot be initiated with a specified P-coil.
- The registered task is executed on the CMU module. If a CMU module is not installed in the system, the P-coil will only be energized and de-energized; the registered task will not be initiated.
- If a specified P-coil is energized (ON) more than once during the execution time of the task registered with the same number as that P-coil's, these multiple task initiation requests will be met in the following way:  
 When the task is waiting for execution on the CPU (i.e., it is already initiated but not executed yet): The task will be initiated only a second time after the first initiated execution has been completed.  
 When the task is in a state other than waiting for execution on the CPU: The task will be initiated only a second and a third time, if any, after the first initiated execution has been completed.
- After it has issued a task initiation request to the CMU module, the LPU module starts to monitor responses from the CMU module. If no response is received therefrom within a fixed time period, the LPU module discontinues the execution of the ladder program. The fixed time period here is either the set value (in the range 50 to 10000 [ms]; defaulted to 2000) of the watchdog timer's monitoring time or the value 10000 (ms), whichever is smaller.
- This task initiation feature using a process register (generically named "P") is available only in the LPU and CMU modules' revisions listed in the table below. With any other combination of the two modules' revisions, even if process registers are used as described above, they will not initiate the tasks having the same numbers as the process registers'.

Module name	Module model	Module revision
LPU (basic module)	LQP510	D or later
CMU	LQP520	B or later



## E EVENT REGISTERS

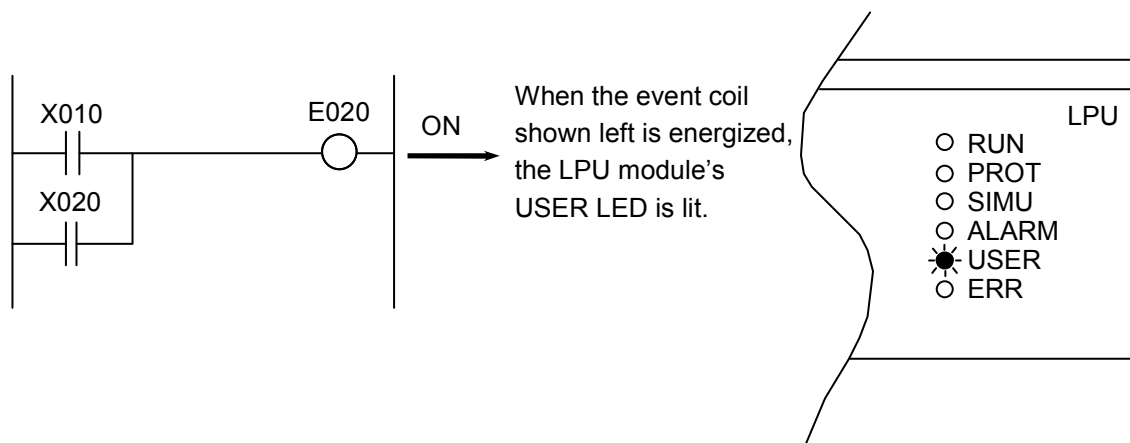
Range of numbers	000 to FFF
Range of numbers usable for lighting the USER LED	000 to 1FF
Range of numbers usable for 4-channel analog and pulse counter I/O operations	400 to FFF

An event register is used to output event information, such as information on user errors. The event information output can be monitored by using the S10V base system's event register monitoring function. In addition to the event information output, the USER LED indicator of the LPU module is also lit when one of the switches E000 through E01F is turned on.

Once it is lit, the USER LED will stay ON until all of E000 through E01F have been turned off. However, the range of register numbers that are presented on screen by the above-mentioned event register monitoring function of the basic tool (S10V base system), and the range of register numbers that can be used to light the LPU module's USER LED, are both from E000 to E1FF. As for the numbers 400 through FFF, they can be used for input and output with the analog and pulse counter modules connected for remote I/O operations. To use them, proper settings must be made in advance by selecting [Utility] – [PCs edition] – [Analog counter] in the S10V Ladder Chart System (model S-7895-02). For information on how to operate the ladder chart system, refer to the "SOFTWARE MANUAL OPERATION S10V LADDER CHART For Windows® (Manual number SVE-3-131)."

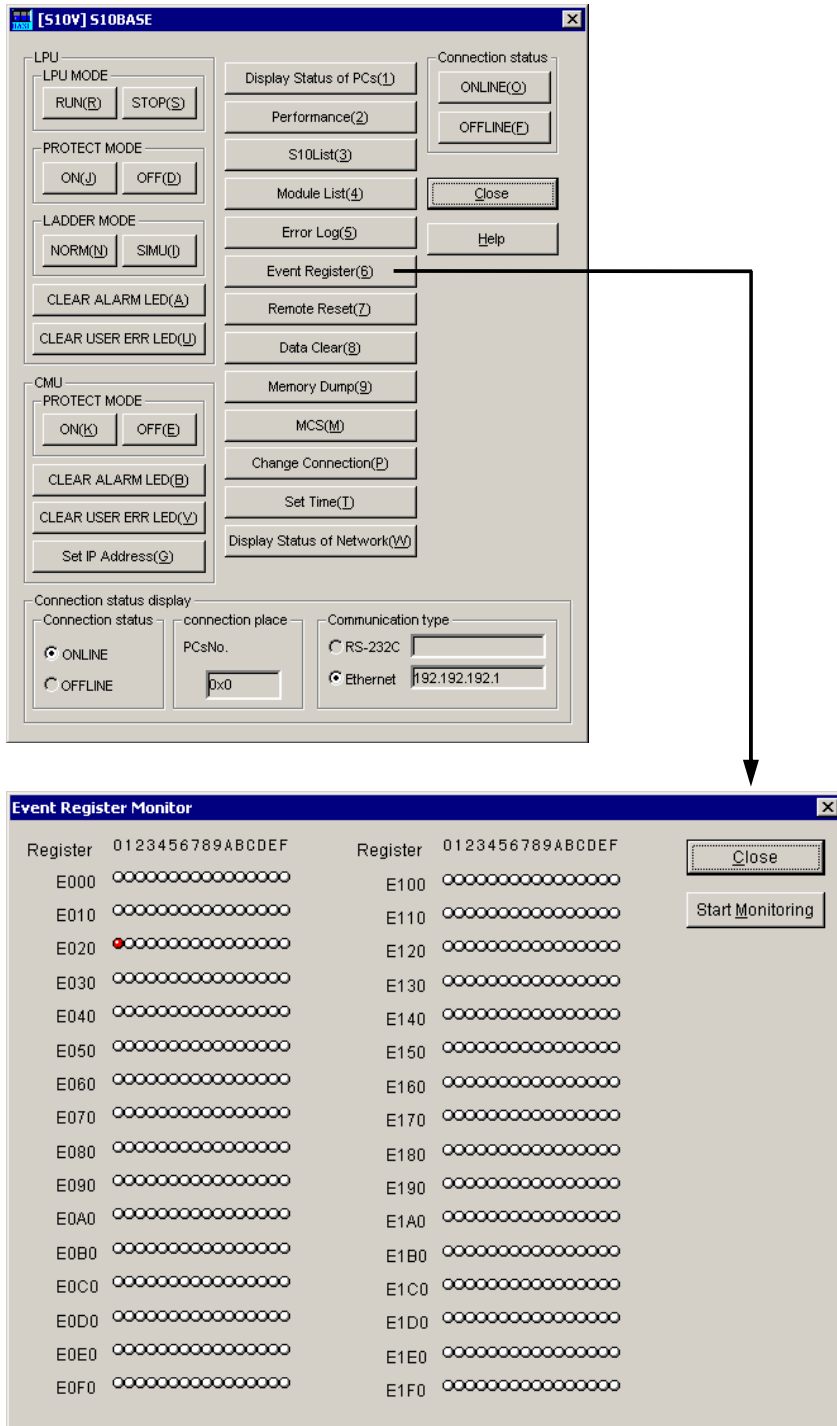
Note: Not all types of available analog modules use event registers. For information on which type of analog module uses event registers, refer to instruction manuals on analog modules.

### ● Usage example



The event information output can be viewed by using the S10V base system's event information display function:

Event register monitoring using the S10V base system



## V EDGE CONTACTS

Range of numbers	000 to FFF
------------------	------------

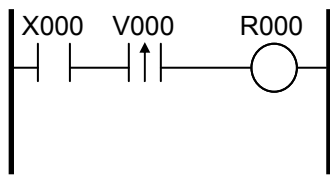
An edge contact is classified either as a rising-edge contact (⌈ ↑ ⊥) or a falling-edge contact (⌈ ↓ ⊥) only during the sequence cycle in which its rising edge or falling edge, respectively, is detected.

**None of the numbers shown left may be used for both a rising-edge contact and a falling-edge contact in the same program.**

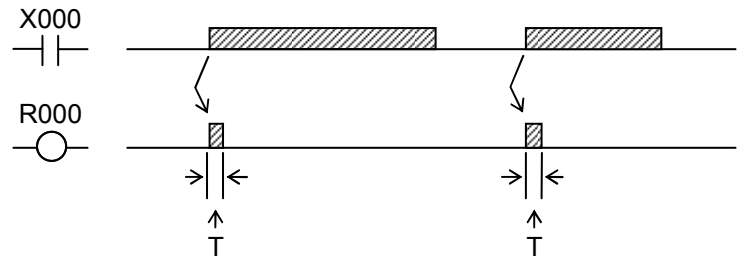
### (1) Rising-edge contact

A rising-edge contact remains closed (ON) only during the sequence cycle in which its rising edge (a transition from OFF to ON state) is detected.

<Usage example>



<Timing chart>

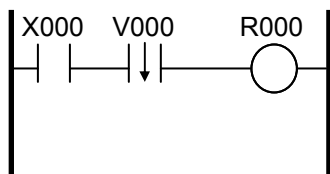


T: Each is one single sequence cycle.

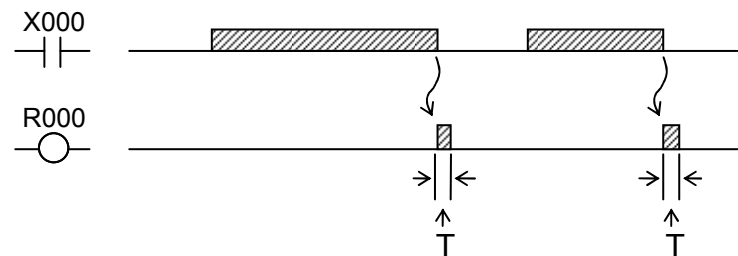
### (2) Falling-edge contact

A falling-edge contact remains closed (ON) only during the sequence cycle in which its falling edge (a transition from ON to OFF state) is detected.

<Usage example>



<Timing chart>



T: Each is one single sequence cycle.

Note: Do not use more than one rising-/falling-edge contact with the same number in more than one place in the same ladder program. Disregarding this rule will cause the ladder program to malfunction.

**This Page Intentionally Left Blank**

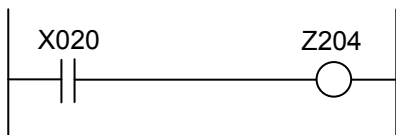
## Z ZEE REGISTERS

Range of numbers	000 to 3FF
Register usable for generating interrupts to host	204
Register usable for generating interrupts from RS-422 modules to host	200 to 203

The zee register Z204 is used to generate H-7338-based interrupts to the host computer connected to the LPU module's UP LINK. The zee registers Z200 through Z203 are used to generate interrupts from an RS-422 module (option) to the host computer. For details on Z200 through Z203, refer to the "USER'S MANUAL OPTION RS-232C/422 (LQE560/565) (Manual number SVE-1-121)." No zee register numbers other than Z200 through Z204 may be used.

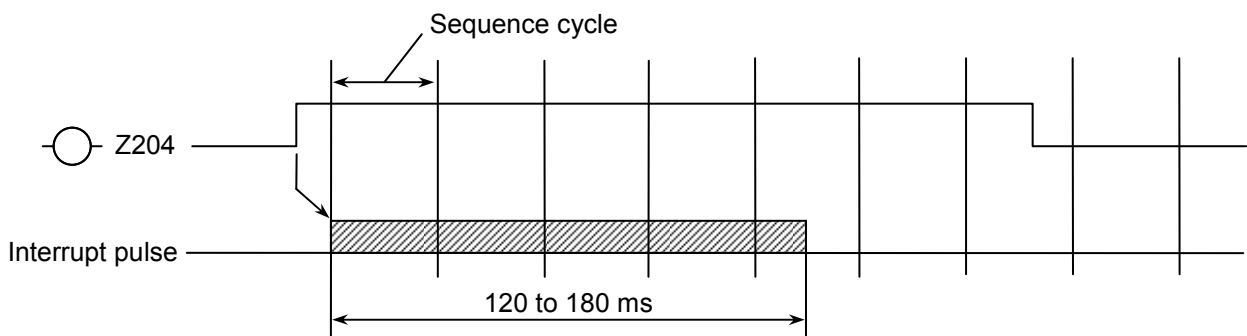
Settling pulse width: At least 1 sequence cycle

<Usage example for generating interrupts to host>



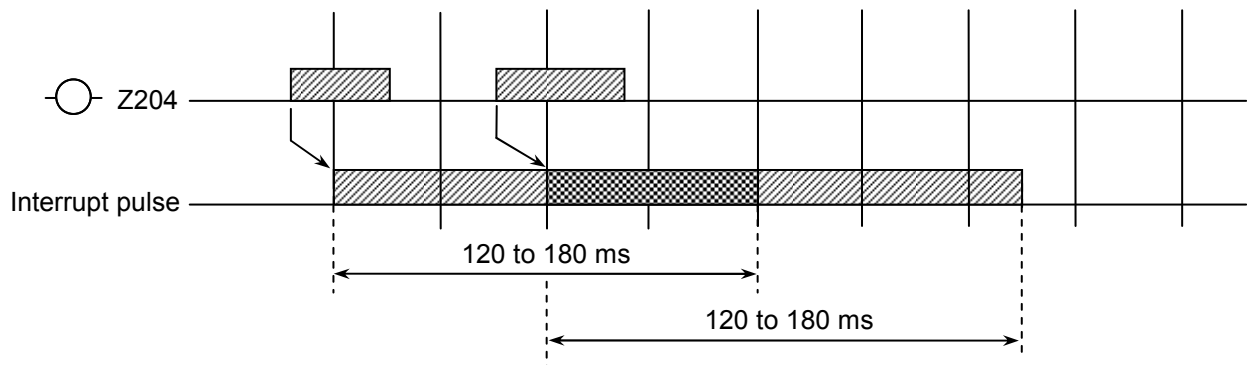
This circuit is used to generate an interrupt pulse of 120 to 180 ms duration to the host computer at the time when a transition from OFF to ON state made by the coil Z204 is detected. This processing occurs synchronously with sequence cycles. If another OFF-to-ON transition made by Z204 is detected during the interrupt pulse being generated, an interrupt pulse of 120 to 180 ms duration will be generated again.

<Timing chart>



<Cases where a generated interrupt pulse is elongated>

In the above circuit, if Z204 makes two or more ON-to-OFF transitions during an interval of 120 to 180 ms, multiple interrupts will be generated, resulting in an interrupt pulse longer than normal.



To avoid this, proper interlocking is required between the above circuit and the host computer receiving interrupts.

## S SYSTEM REGISTERS

Range of numbers	000 to BFF	System registers are read-only registers reflecting the system's operation performed or other things relating to the system.
------------------	------------	--

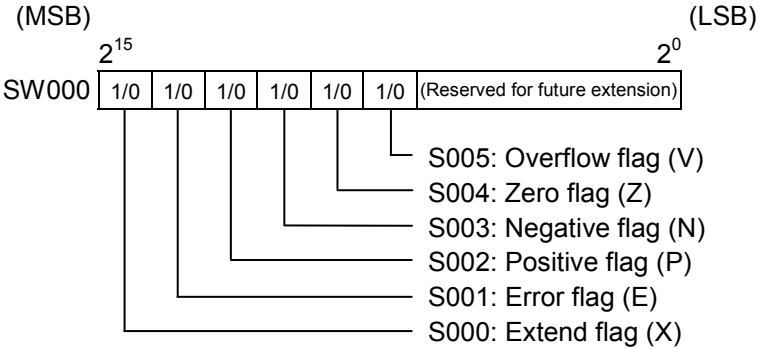
Table 1-6 is a list of all available system registers.

Table 1-6 System Registers

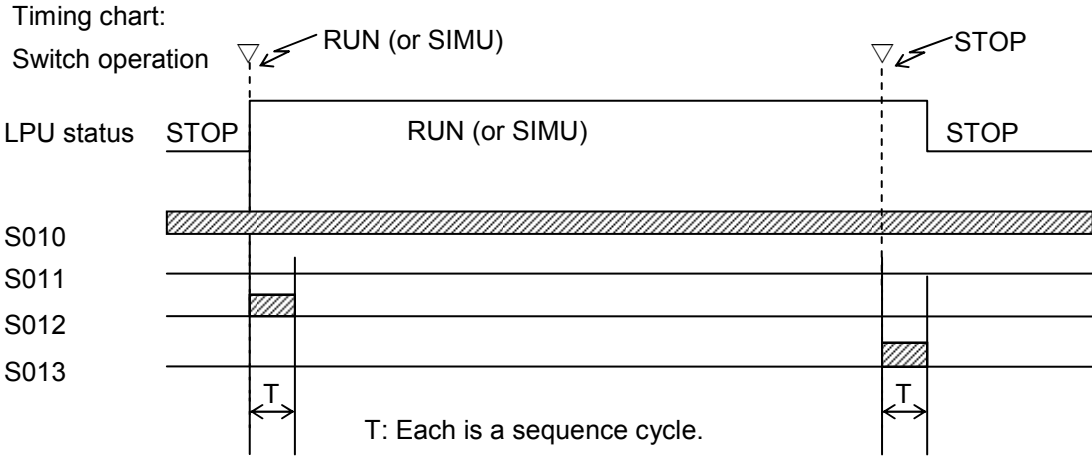
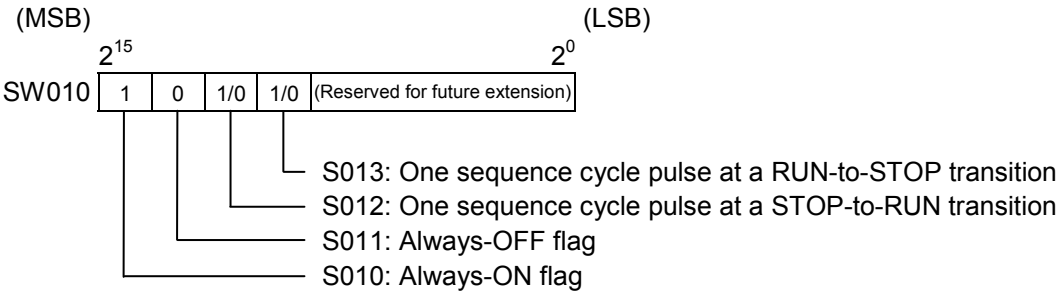
No.	Register numbers	Register naming
1	S000 thru S00F	Arithmetic-function flag registers
2	S010 thru S01F	Ladder program control registers
3	S020 thru S02F	HI-FLOW application-instruction execution-result flag registers
4	S030 thru S0FF	(Reserved for future use by the system)
5	S100 thru S15F	Ladder program control counter
6	S160 thru S1FF	(Reserved for future use by the system)
7	S200 thru S20F	Time control registers
8	S210 thru S27F	(Reserved for future use by the system)
9	S280 thru S2EF	Time setting registers
10	S2F0 thru S2FF	(Reserved for future use by the system)
11	S300 thru S47F	Remote I/O status registers
12	S480 thru S4FF	(Reserved for future use by the system)
13	S500 thru S68F	Optional-module status registers
14	S690 thru S6AF	Ethernet communication result flag registers
15	S6B0 thru S8FF	(Reserved for future use by the system)
16	S900 thru S93F	Sequence-cycle scan-time registers
17	S940 thru S97F	Ladder execution-time registers
18	S980 thru S9BF	Optional-module status registers (D.NET)
19	S9C0 thru S9FF	Ethernet communication result flag registers
20	SA00 to SA8F	Optional-module status registers (J.NET/IR.LINK)
21	SA90 thru SAFF	(Reserved for future use by the system)
22	SB00 thru SB1F	LPU-unit I/O information registers
23	SB20 thru SBEF	(Reserved for future use by the system)
24	SBF0 thru SBFF	LPU status registers

(1) Arithmetic-function flag registers

Arithmetic-function flag registers indicate the set/reset statuses of predefined flags that occur upon the execution of system arithmetic-function instructions. These registers cannot be referenced from the ladder circuit monitor and MCS functions (if an attempt is made to do so, the registers are always displayed as “OFF”).



(2) Ladder program control registers



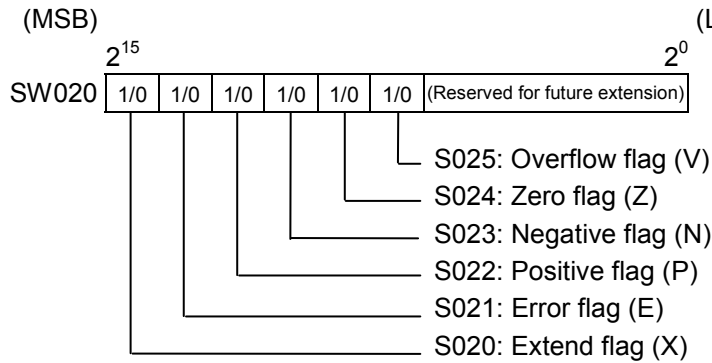
Note: None of the above bit registers S010, S011, S012, and S013 become ON in the event of a power outage.



## S SYSTEM REGISTERS

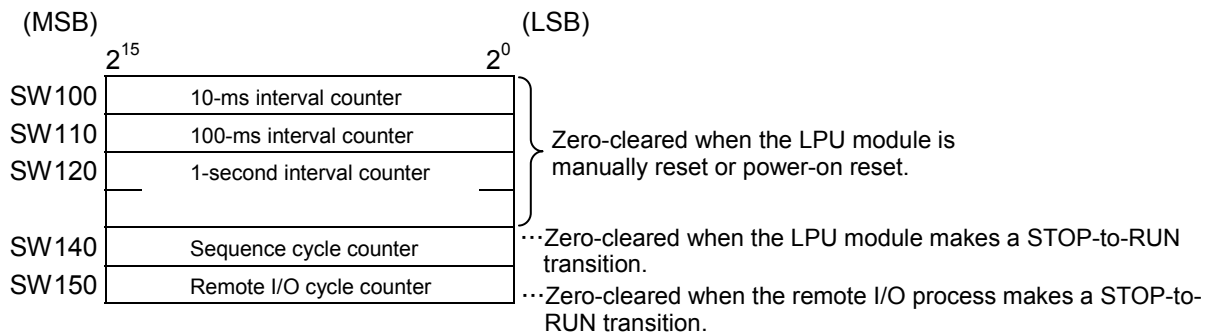
### (3) HI-FLOW application-instruction execution-result flag registers

HI-FLOW application-instruction execution-result flag registers indicate the statuses of predefined flags that occur upon the execution of HI-FLOW application instructions.



### (4) Ladder program control registers

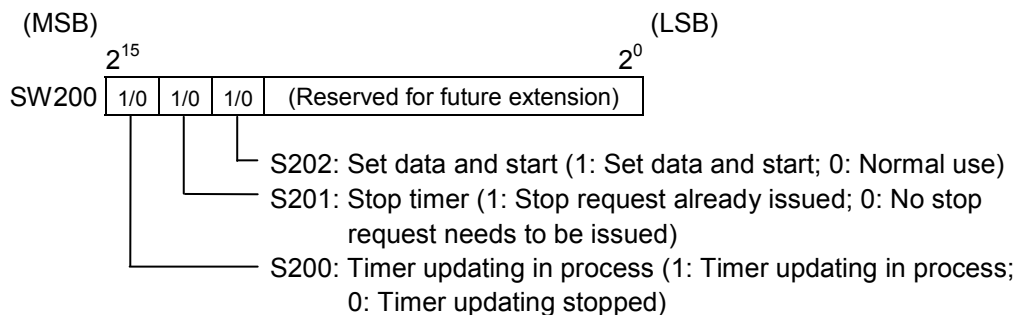
Ladder program control registers are counters that can be used in sequence control.



- All the above counters start counting from 0 again when they overflow.
- Any of the above counters will have an error of approximately  $\pm 10\%$  because their precision depends on interrupts handled by the operating system (OS).

### (5) Time control registers

Time control registers are provided as a means of controlling the setting of current time in the LPU module. They are used when setting the current time in the LPU module.



## (6) Time setting registers

Time setting registers are used to store values indicating the year, month, day of month, hours, minutes, seconds, and day of week. When you make time settings in the LPU module, store time information in these registers. Data stored in these registers must be in binary format.

	(MSB) $2^{15}$	$2^8$ $2^7$	(LSB) $2^0$
SW280	Unused		Seconds
SW290	Unused		Minutes
SW2A0	Unused		Hours
SW2B0	Unused		Day of month
SW2C0	Unused		Month
SW2D0	Year		
SW2E0	Unused		Day of week

Seconds: Must be in the range 0 to 59.

Minutes: Must be in the range 0 to 59.

Hours: Must be in the range 0 to 23.

Day of month: Must be in the range 1 to 31.

Month: Must be in the range 1 to 12.

Year: Must be in the range 1970 to 2069.

Day of week: Must be in the range 1 to 7.

(1: Sun; 2: Mon; 3: Tue; 4: Wed; 5: Thu; 6: Fri; 7: Sat)

## S SYSTEM REGISTERS

### (7) Remote I/O status registers

Remote I/O status registers present remote I/O station information, such as station registered or not, timeout error detected or not, and fuse blown or not.

<Register assignment>

S300	Registered stations
S380	Timed-out stations
S400	Fuse-blown stations
S47F	

- All stations that are currently connected to the communication line and that have thus far responded normally at least once have their associated registers set to 1. (\*)
- All registered stations in which a timeout error has been detected have their associated registers set to 1. (\*)
- All registered stations in which a fuse-blown condition (DO module fuse blown) has been detected have their associated registers set to 1. (\*)

(\*) One-to-one correspondence between stations and bits:

No.	X- or Y-number	Registered station	Timed-out station	Fuse-blown station
0	000 to 00F	S300	S380	S400
1	010 to 01F	S301	S381	S401
2	020 to 02F	S302	S382	S402
3	030 to 03F	S303	S383	S403
4	040 to 04F	S304	S384	S404
⋮	⋮	⋮	⋮	⋮
124	7C0 to 7CF	S37C	S3FC	S47C
125	7D0 to 7DF	S37D	S3FD	S47D
126	7E0 to 7EF	S37E	S3FE	S47E
127	7F0 to 7FF	S37F	S3FF	S47F

## (8) Optional-module status registers

Optional-module status registers are used to store error information for inter-CPU links, RS-232C/RS-422, etc. As shown below, these registers are organized into five groups according to the module types used. Data is set in each group of registers by a system program provided for its associated module type. For details on the bit configuration unique to each type of optional module, refer to the user's manual on the optional module.

<Register assignment for optional modules>

S500	(Reserved for future use by the system)
S580	Inter-CPU link module information
S5C0	RS-232C/RS-422 module information
S640	High-speed remote I/O module information
S680 S6FF	(Reserved for future use by the system)

The above registers are zero-cleared when the S10V unit is power-on reset or the LPU module is manually or remotely reset.

## S SYSTEM REGISTERS

### (9) Sequence-cycle scan-time registers

Sequence-cycle scan-time registers are used to store the result of measurements of sequence cycles.

	(MSB) $2^{15}$	(LSB) $2^0$	
SW900	Latest measurement value (ms)		} Zero-cleared when the status of the LPU module is switched from STOP to RUN.
SW910	Maximum measurement value (ms)		
SW920	Minimum measurement value (ms)		
SW930	Average value of latest 16 measurements (ms) (*)		

(\*) The above average value is not stored in place until the 16th measurement is completed.

### (10) Ladder execution-time registers

Ladder execution-time registers are used to store the result of measurements of ladder execution times. Where HI-FLOW is used, the ladder and the HI-FLOW execution time are added together and the result is stored in place.

	(MSB) $2^{15}$	(LSB) $2^0$	
SW940	Latest measurement value (ms)		} Zero-cleared when the status of the LPU module is switched from STOP to RUN.
SW950	Maximum measurement value (ms)		
SW960	Minimum measurement value (ms)		
SW970	Average value of latest 16 measurements (ms) (*)		

(\*) The above average value is not stored in place until the 16th measurement is completed.

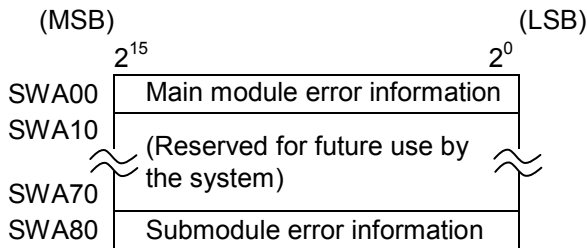
### (11) Optional-module status registers (D.NET)

These optional-module status registers are used to store error information on errors detected in each D.NET module (one of channels 0 through 3). For details, refer to the “USER’S MANUAL OPTION D.NET (LQE570/575) (Manual number SVE-1-106).”

	(MSB) $2^{15}$	(LSB) $2^0$
SW980	Channel-0 module error information	
SW990	Channel-1 module error information	
SW9A0	Channel-2 module error information	
SW9B0	Channel-3 module error information	

(12) Optional-module status registers (J.NET/IR.LINK)

These optional-module status registers are used to store error information on errors detected in each J.NET or IR.LINK module (main or submodule). For details, refer to the “USER’S MANUAL OPTION J.NET (LQE540) (Manual number SVE-1-104)” and “USER’S MANUAL OPTION J.NET-INT (LQE545) (Manual number SVE-1-107),” or the “USER’S MANUAL OPTION IR.LINK (LQE546) (Manual number SVE-1-117).”



(13) LPU-unit I/O information registers

LPU-unit I/O information registers are used to store I/O unit information for the LPU module. Each bit in the information is in one-to-one correspondence with one of the slots involved.

Bit configuration for LPU unit I/O information

	(MSB)	2 <sup>15</sup>	2 <sup>14</sup>	2 <sup>13</sup>	2 <sup>12</sup>	2 <sup>11</sup>	2 <sup>10</sup>	2 <sup>9</sup>	2 <sup>8</sup>	2 <sup>7</sup>	2 <sup>6</sup>	2 <sup>5</sup>	2 <sup>4</sup>	2 <sup>3</sup>	2 <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>	(LSB)
SWB00	Error information (=0: Normal; =1: Fuse blown)								Mounting information (=0: Not mounted; =1: Mounted)									
SWB10	I/O type (=0: DI/DO; =1: AI/AO/PCT (*))								0 (fixed; reserved for future use)									
Bit No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		

(\*) PCT: Pulse counter.

Slot No.	Bit register		
	Error information	Mounting information	I/O type
0	SB00	SB08	SB10
1	SB01	SB09	SB11
2	SB02	SB0A	SB12
3	SB03	SB0B	SB13
4	SB04	SB0C	SB14
5	SB05	SB0D	SB15
6	SB06	SB0E	SB16
7	SB07	SB0F	SB17

## S SYSTEM REGISTERS

### (14) LPU status registers

LPU status registers indicates the current state of the LPU.

<LPU status bit configuration>

	(MSB)															(LSB)
	2 <sup>15</sup>	2 <sup>14</sup>	2 <sup>13</sup>	2 <sup>12</sup>	2 <sup>11</sup>	2 <sup>10</sup>	2 <sup>9</sup>	2 <sup>8</sup>	2 <sup>7</sup>	2 <sup>6</sup>	2 <sup>5</sup>	2 <sup>4</sup>	2 <sup>3</sup>	2 <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>
SWBF0	1/0	1/0	*	1/0	1/0	*	1/0	*	1/0	1/0	1/0	1/0	*	1/0	1/0	0
Bit No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

\* Each of these bits is reserved for future extension.

Bit No.	Bit register No.	Meanings of bits	
		ON (=1)	OFF (=0)
0	SBF0	Currently in STOP state.	Currently in RUN state.
1	SBF1	Simulation currently in process.	Currently running normally.
2	SBF2	(Reserved for future extension)	
3	SBF3	Protection switch currently in ON state.	Protection switch currently in OFF state.
4	SBF4	Remote I/O operation currently in progress.	Remote I/O operation currently stopped.
5	SBF5	(Reserved for future extension)	
6	SBF6	Ladder-rewriting process currently in progress.	Ladder-rewriting process completed.
7	SBF7	(Reserved for future extension)	
8	SBF8	CELL error (*1) warning.	CELL normal.
9	SBF9	Timed-out station existent.	No timed-out station existent.
A	SBFA	Fuse-blown station existent.	No fuse-blown station existent.
B	SBFB	Optional-module error (*2) detected.	No optional-module error (*2) detected.
C	SBFC	(Reserved for future extension)	
D	SBFD	Zero-cleared in a general (power-on) reset (GR) or manual/remote reset.	
E	SBFE	LPU currently down.	LPU currently up and running normally.
F	SBFF	—	LPU's OS currently running.

(\*1) The CELL error is a “battery low” condition of the memory backup battery provided in the LPU.

(\*2) The optional-module error is a parity error detected during accessing the internal memory of the optional module from the LPU.

## (15) Ethernet communication result flag registers

Ethernet communication result flag registers are used to store special flags for indicating the result of execution of Ethernet communication instructions.

Execution results are flagged in the system registers S9C0 through S9EF and S690 through S6AF according to the management numbers, which are predefined in one-to-one correspondence with all available sockets. When the execution of an Ethernet communication instruction is terminated normally or abnormally, the result is flagged by setting the system register associated with the management number to 0 or 1, respectively.

Register type		Management number	Remarks
Word	Bit		
SW9C0	S9C0	1	Provided for CMU Ethernet communications.
	S9C1	2	
	⋮	⋮	
	S9CE	15	
SW9D0	S9CF	16	Provided for ET.NET (main module) Ethernet communications.
	S9D0	17	
	S9D1	18	
	⋮	⋮	
SW9E0	S9DE	31	Provided for ET.NET (submodule) Ethernet communications.
	S9DF	32	
	S9E0	33	
	S9E1	34	
SW690	⋮	⋮	Provided for OPTET Ethernet communications.
	S9EE	47	
	S9EF	48	
	S690	49	
SW6A0	S691	50	Provided for OPTET Ethernet communications.
	⋮	⋮	
	S69E	63	
	S69F	64	
SW6A0	S6A0	65	Provided for OPTET Ethernet communications.
	S6A1	66	
	⋮	⋮	
	S6AE	79	
	S6AF	80	



# LR, LV LADDER CONVERTER-SPECIFIC WORK REGISTERS

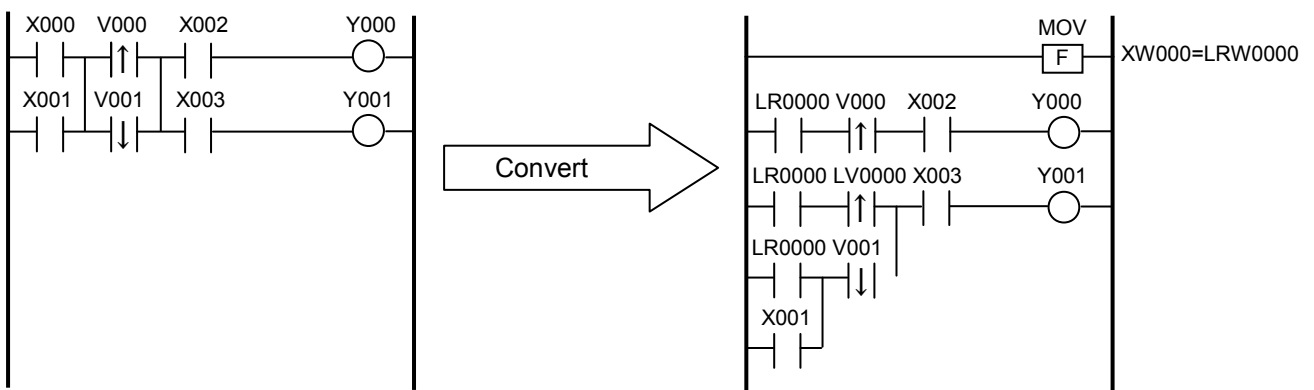
Range of numbers	0000 to 0FFF
------------------	--------------

These work registers are used by the ladder program converter during converting S10/2α Series' or S10mini Series' downward-sloping-rung ladder programs into normal-rung ladder format. Users are advised not to use these registers.

LR: Used for contacts or coils.

LV: Used for edge contacts.

● Usage example



**This Page Intentionally Left Blank**

1.6 Ladder Watchdog Timer

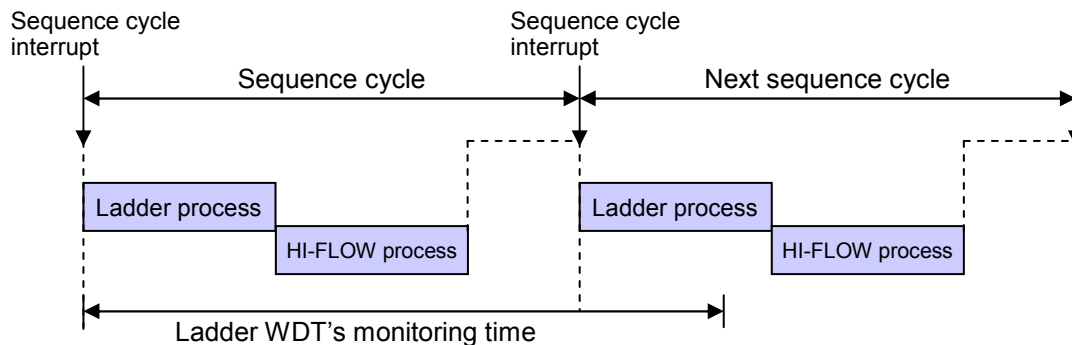
The ladder watchdog timer is used to monitor whether the execution of a ladder and any HI-FLOW process is ended within a user-set time period, called a monitoring time. If the execution is not ended within the monitoring time, it results in a ladder program watchdog timeout error (hereinafter simply called a ladder WDT error) and the following steps are automatically performed:

- The LPU's ERR LED indicator is lit, detailed information on the error is recorded, and then the LPU is stopped.
- All of the ladder, HI-FLOW, C-mode task, remote I/O communications, etc. currently in operation come to a stop, except for the RS-232C process used for communication with the S10V base system.
- The PCs' OK signal is turned off.

1.6.1 An outline of the ladder watchdog timer's operation

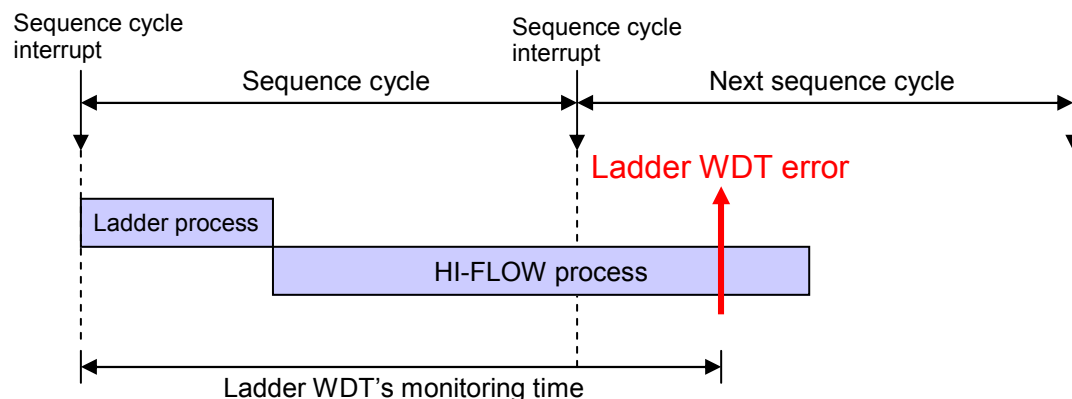
<Normal operation>

As shown below, as long as a ladder process and, if HI-FLOW is used, a HI-FLOW process as well are ended within the set monitoring time, no ladder WDT error will occur, because the ladder WDT is reset when a sequence cycle interrupt is generated.



<Operation when a timeout is detected>

If a ladder process and, if HI-FLOW is used, a HI-FLOW process as well are not ended within the set monitoring time, due to, for instance, the occurrence of an endless loop, the ladder WDT signals a timeout and this timeout is detected as a ladder WDT error, resulting in the immediate termination of the ladder and HI-FLOW processes.



### 1.6.2 Range of settable monitoring time values

The WDT's monitoring time can be set to any value within the range shown below. At shipment, it is set to the default value 2000 (ms).

Range of settable values: 50 to 10000 (ms)

Notes:

- When changing the set monitoring time, find the total processing time required for the execution of a ladder and any HI-FLOW process, add a value of 50 (ms) or larger to the obtained total, and then set the result as a new monitoring time.
- The ladder WDT is reset at the end of each sequence cycle. Therefore, if the monitoring time is set to a value smaller than the set value of sequence cycle, a ladder WDT error could occur during the normal operation of the ladder. To avoid this, any monitoring time set to a value smaller than the sequence cycle value is automatically replaced with a value of "sequence cycle + 10 ms".

### 1.6.3 Error information presented upon ladder WDT errors

When a ladder WDT error occurs, the ERR LED indicator of the LPU module comes on. This ladder WDT error can be distinguished from other types of error by using the S10V base system's error log display function. A ladder WDT error, when detected, is always notified by displaying the error code "0x1206" and message "Ladder watchdog-timer timeout error" on the S10V base system. For information on how to operate the S10V base system, refer to the "USER'S MANUAL BASIC MODULES (Manual number SVE-1-100)."

**This Page Intentionally Left Blank**

## 2 ARITHMETIC FUNCTIONS

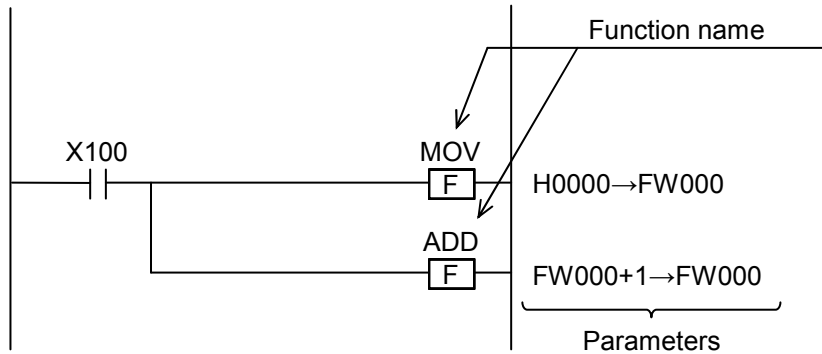
## 2 ARITHMETIC FUNCTIONS

### 2.1 Functional Overview

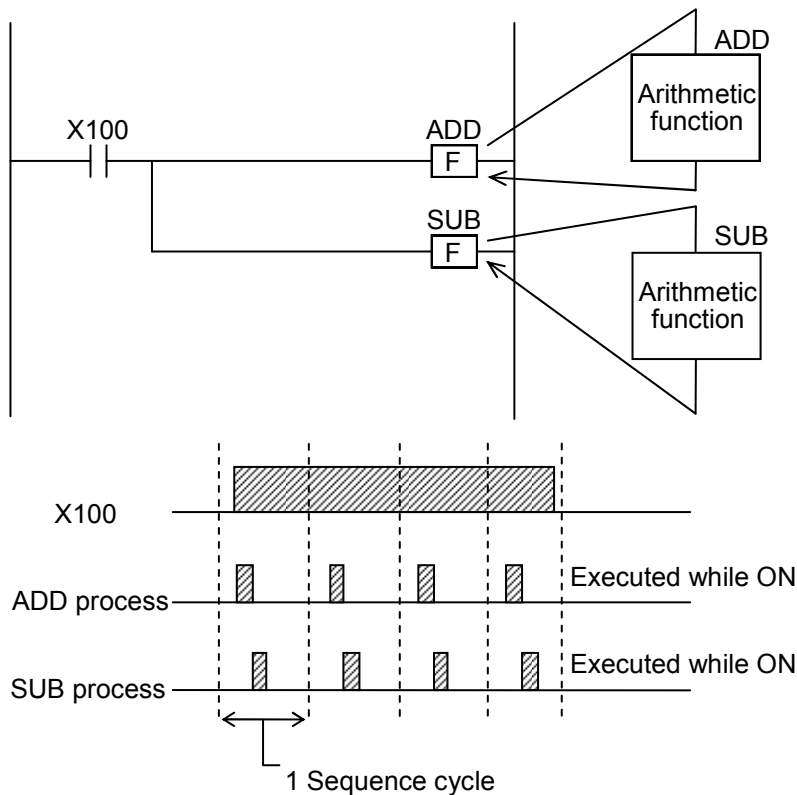
If you want to carry out arithmetic operations in a ladder program, use arithmetic functions. They will simplify your programming work.

- Operation of arithmetic functions

[Example circuit]



[Operation]



The above functions are executed during each sequence cycle as long as the input condition is met.

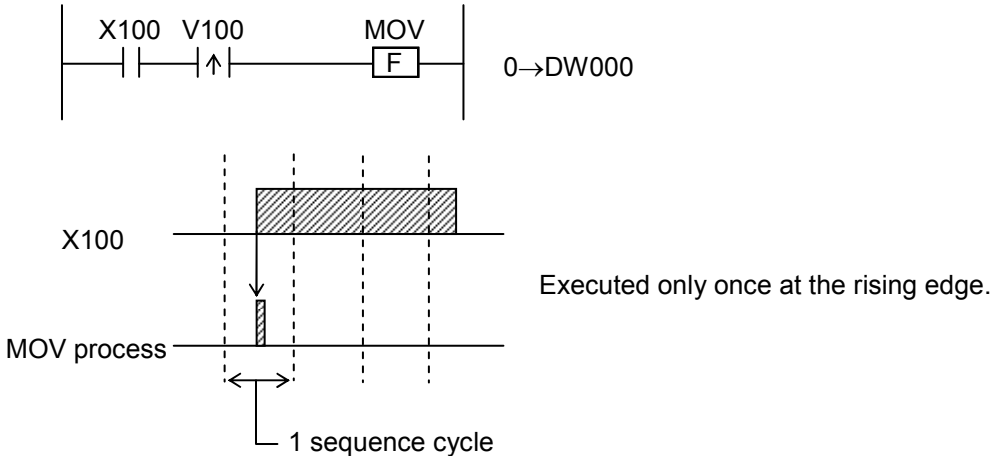
(1) Parameters

Each arithmetic function, assigned a unique name signifying its operation, takes one or more parameters. Registers and constants can be specified as parameters to arithmetic functions.

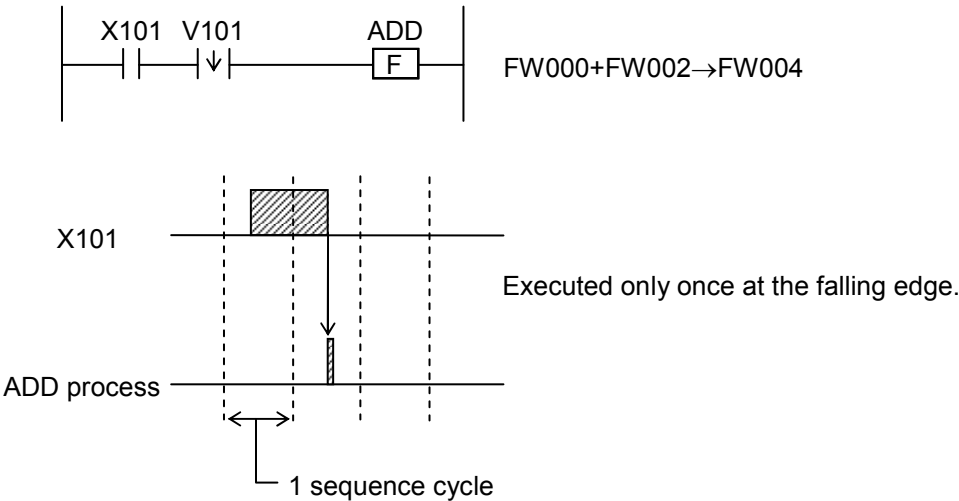
(2) Operation

Arithmetic functions are initiated during each sequence cycle as long as the coil remains energized (ON). If you want to initiate an arithmetic function only once when the coil makes a transition from OFF to ON state or from ON to OFF state, use the function in conjunction with a rising-edge or a falling-edge contact, respectively.

Example 1: Using an arithmetic function together with a rising-edge contact:



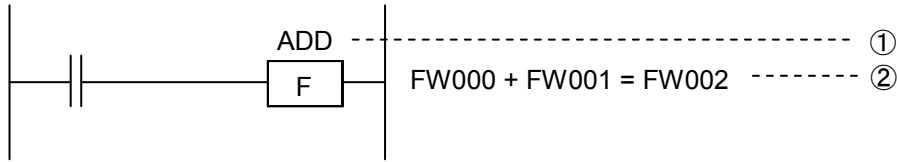
Example 2: Using an arithmetic function together with a falling-edge contact:





2.2 Functional Specifications

(1) General makeup of arithmetic functions



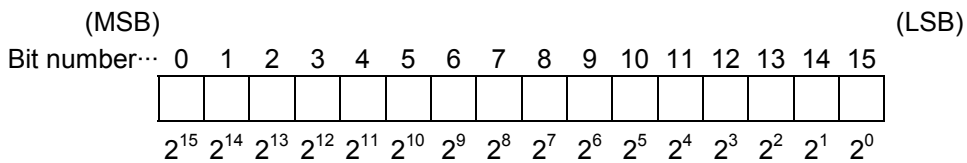
- ① Function name: Name of arithmetic function.
- ② Parameters: Each is a register or constant to be operated on.

(2) Data formats

The types of data that can be used with arithmetic functions are word, long-word, and floating:

- Word

Each piece of word data is a signed 16-bit single-precision integer. In the word format, each bit is given a bit number, as shown below.



The range of allowable word data is as follows:

In decimal: -32768 to +32767

In hexadecimal: H8000 to H7FFF, where the letter “H” denotes that the number which follows is in hexadecimal notation.

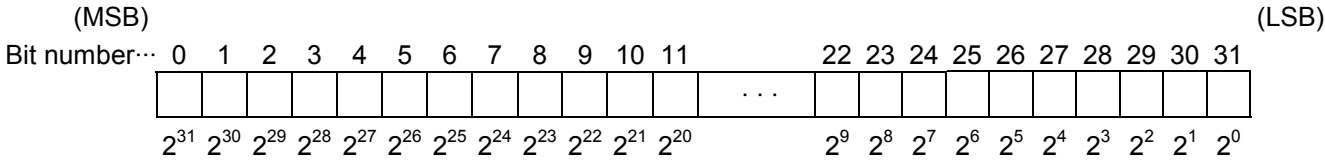
Note 1: Bit registers, such as X000 and R123, are handled as word data, where only the LSB (least significant bit) is valid as data. For details, see Subsection 2.3.2, “Handling of bit registers.”

Note 2: The values (counts) of ON-delay timers (T), one-shot timers (U), and up-down counters (C) (TC\*\*\*, UC\*\*\*, CC\*\*\*:\*\*\*=number) are all handled as word data. The same is true with their set values (TS\*\*\*, US\*\*\*, CS\*\*\*:\*\*\*=number).

- Long-word

Each piece of long-word data is a signed 32-bit double-precision integer.

In the long-word format, each bit is given a bit number, as shown below.



The range of allowable long-word data is as follows:

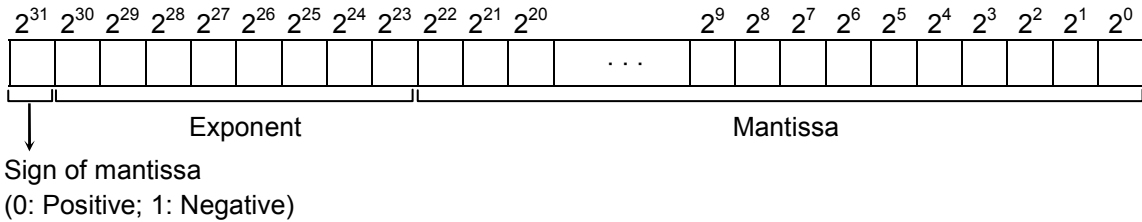
In decimal: -2147483648 to +2147483647

In hexadecimal: H80000000 to H7FFFFFFF

- Floating

Each piece of floating data is a 32-bit single-precision floating-point number.

Floating data has the following bit configuration:



The range of allowable floating data is as follows:

$$0, \pm 2^{-126} \text{ to } \pm 2^{128}$$

If one of the following errors occurs in a floating-point arithmetic operation, it is notified as described below.

Invalid operation: Of the operation result flag bits provided, the E-bit is set to 1. The content of the register to store operation results remains unchanged.

Division by zero: Of the operation result flag bits provided, the E-bit is set to 1. The content of the register to store operation results remains unchanged.

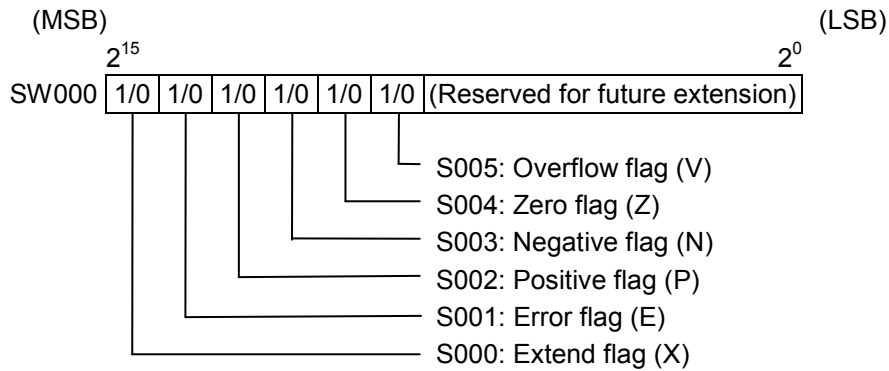
Overflow: The maximum finite number that can be represented internally as an absolute value ( $\pm 3.402823E38$ ) is returned.

Underflow: The number 0 with correct sign is returned.

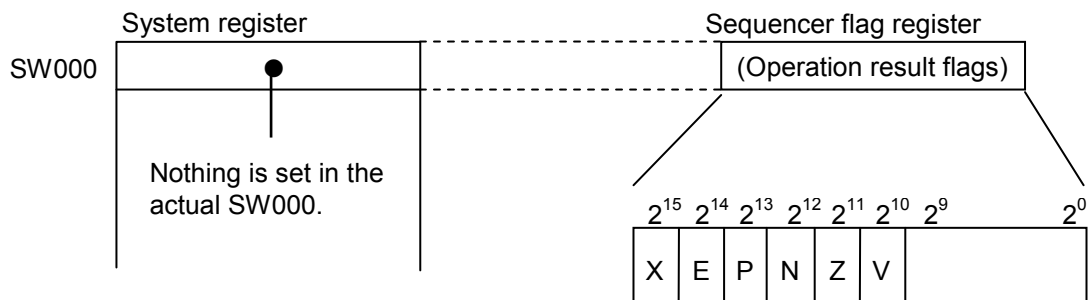
## 2 ARITHMETIC FUNCTIONS

### (3) Flag settings

Arithmetic functions set an appropriate operation result flag(s) to report on the result of the operation performed. The following description deals with the types of flags provided, where they are set, and the conditions for setting them.



Note 1: The registers S000 through S00F are read-only registers that reflect flag statuses after the execution of system arithmetic function instructions. These registers can be referenced in ladder programs, but neither in the ladder tool's MCS nor the ladder circuit monitor.



Note 2: The results of floating-point operations are also reflected in these flags.

<Conditions for flag settings>

No.	Type	Flags						Flag setting condition		
		X	E	P	N	Z	V	For word:	For long-word:	For floating:
1	ADD	-	-	-	-	-	●	V: Set to 0 if the operation result is in the range -32768 to 32767; otherwise, set to 1.	V: Set to 0 if the operation result is in the range -2147483648 to 2147483647; otherwise, set to 1.	
2	ADD (floating)	-	●	-	-	-	-			E: Set to 1 if the operation ends up with an error(*); otherwise, set to 0.
3	SUB	-	-	-	-	-	●	V: Set to 0 if the operation result is in the range -32768 to 32767; otherwise, set to 1.	V: Set to 0 if the operation result is in the range -2147483648 to 2147483647; otherwise, set to 1.	
4	SUB (floating)	-	●	-	-	-	-			E: Set to 1 if the operation ends up with an error(*); otherwise, set to 0.
5	INC	-	-	-	-	-	●	V: Set to 0 if the operation result is in the range -32768 to 32767; otherwise, set to 1.	V: Set to 0 if the operation result is in the range -2147483648 to 2147483647; otherwise, set to 1.	
6	DEC	-	-	-	-	-	●	V: Set to 0 if the operation result is in the range -32768 to 32767; otherwise, set to 1.	V: Set to 0 if the operation result is in the range -2147483648 to 2147483647; otherwise, set to 1.	
7	MUL	-	-	-	-	-	●	V: Set to 0 if the operation result is in the range -32768 to 32767; otherwise, set to 1.	V: Set to 0 if the operation result is in the range -2147483648 to 2147483647; otherwise, set to 1.	
8	MUL (floating)	-	●	-	-	-	-			E: Set to 1 if the operation ends up with an error(*); otherwise, set to 0.
9	DIV	-	●	-	-	-	●	E: Set to 1 if the divisor is zero (0); otherwise, set to 0. V: Set to 1 if the quotient is 32768; otherwise, set to 0.	E: Set to 1 if the divisor is zero (0); otherwise, set to 0. V: Set to 1 if the quotient is 2147483648; otherwise, set to 0.	
10	DIV (floating)	-	●	-	-	-	-			E: Set to 1 if the divisor is zero (0); otherwise, set to 0. Also, set to 1 if the operation ends up with an error(*); otherwise, set to 0.
11	MOD	-	●	-	-	-	●	E: Set to 1 if the divisor is zero (0); otherwise, set to 0. V: Set to 1 if the quotient is 32768; otherwise, set to 0.	E: Set to 1 if the divisor is zero (0); otherwise, set to 0. V: Set to 1 if the quotient is 2147483648; otherwise, set to 0.	
12	SCL	-	●	-	-	-	●	E: Set to 1 if the divisor is zero (0); otherwise, set to 0. V: Set to 0 if the operation result is in the range -32768 to 32767; otherwise, set to 1.		
13	TST	-	-	●	●	●	-	P: Set to 1 if data value > 0; otherwise, set to 0. N: Set to 1 if data value < 0; otherwise, set to 0. Z: Set to 1 if data value = 0; otherwise, set to 0.		
14	BTD	-	●	-	-	-	●	E: Set to 1 if data value < 0; otherwise, set to 0. V: Set to 1 if data value > 9999; otherwise, set to 0.	E: Set to 1 if data value < 0; otherwise, set to 0. V: Set to 1 if data value > 99999999; otherwise, set to 0.	
15	DTB	-	●	-	-	-	-	E: Set to 1 if a given digit (4-bit) has a value in the range HA to HF; otherwise, set to 0.		
16	APB	-	●	-	-	-	-	E: Set to 1 if a data value other than H30 thru H39 and H41 thru H46 is detected; otherwise, set to 0.		
17	AUB	-	●	-	-	-	-			
18	DTS	-	-	-	-	-	●		V: Set to 0 if the operation result is in the range -32768 to 32767; otherwise, set to 1.	
19	ABS	-	-	-	-	-	●	V: Set to 1 if data value = -32768; otherwise, set to 0.	V: Set to 1 if data value = -2147483648; otherwise, set to 0.	
20	NEG	-	-	-	-	-	●			
21	ECD	-	●	-	-	-	-	E: Set to 1 if data value = 0; otherwise, set to 0.		
22	ASL	-	-	-	-	-	●	V: Set to 1 if the sign bit's value changes at least once during the shift operation.		
23	LIM	-	●	-	-	-	-	E: Set to 1 if upper-limit value < lower-limit value; otherwise, set to 0.		
24	LIM (floating)	-	●	-	-	-	-			E: Set to 1 if upper-limit value < lower-limit value; otherwise, set to 0. Also, set to 1 if the operation ends up with an error(*); otherwise, set to 0.
25	BND	-	●	-	-	-	●	E: Set to 1 if upper-limit value < lower-limit value; otherwise, set to 0. V: Set to 0 if the operation result is in the range -32768 to 32767; otherwise, set to 1.	E: Set to 1 if upper-limit value < lower-limit value; otherwise, set to 0. V: Set to 0 if the operation result is in the range -2147483648 to 2147483647; otherwise, set to 1.	
26	BND (floating)	-	●	-	-	-	-			E: Set to 1 if upper-limit value < lower-limit value; otherwise, set to 0. Also, set to 1 if the operation ends up with an error(*); otherwise, set to 0.
27	ZON	-	●	-	-	-	●	E: Set to 1 if upper-limit value < lower-limit value; otherwise, set to 0. V: Set to 0 if the operation result is in the range -32768 to 32767; otherwise, set to 1.	E: Set to 1 if upper-limit value < lower-limit value; otherwise, set to 0. V: Set to 0 if the operation result is in the range -2147483648 to 2147483647; otherwise, set to 1.	
28	ZON (floating)	-	●	-	-	-	-			E: Set to 1 if upper-limit value < lower-limit value; otherwise, set to 0. Also, set to 1 if the operation ends up with an error(*); otherwise, set to 0.
29	TAN	-	●	-	-	-	-			E: Set to 1 if the operation ends up with an error(*); otherwise, set to 0.
30	ASIN	-	●	-	-	-	-			E: Set to 1 if a given data value is out of the range -1.0 to 1.0; otherwise, set to 0.
31	ACOS	-	●	-	-	-	-			
32	EXP	-	●	-	-	-	-			E: Set to 1 if the operation ends up with an error(*); otherwise, set to 0.
33	LOG	-	●	-	-	-	-			E: Set to 1 if specified value < 0; otherwise, set to 0. Also, set to 1 if the operation ends up with an error(*); otherwise, set to 0.
34	Other than the above	-	-	-	-	-	-	All the flags remaining unchanged.		

-: This flag's value is the same as before the execution of the function.

●: See the description of flag setting conditions in this table.

(\*) This error occurs when the result of the floating-point operation is:

Not zero (0) and outside the range  $\pm 2^{-126}$  to  $\pm 2^{128}$

## 2 ARITHMETIC FUNCTIONS

### 2.3 Registers Used in Arithmetic Functions

As mentioned in Section 2.2, “Functional Specifications,” registers can be specified as parameters to arithmetic functions. This section provides information on the registers used in arithmetic functions.

#### 2.3.1 Registers usable in arithmetic functions

Table 2-1 is a list of all registers that can be used in arithmetic functions. Each of these registers has its unique name and specific use. For efficient programming and maintenance, users are advised to use each register for its intended application. Of course, they may be used for applications other than the intended.

Table 2-1 Registers Usable in Arithmetic Functions

(1/3)

Function name	Register name (size)	Number	Use	Status after reset or power recovery
External input	X (bit)	000 to FFF	Data input from input modules connected for remote I/O operations	Cleared
	XW (word)	000 to FF0		
	XL (long-word)	000 to FE0		
External output	Y (bit)	000 to FFF	Data output to output modules connected for remote I/O operations	Cleared
	YW (word)	000 to FF0		
	YL (long-word)	000 to FE0		
Internal register	R, M, A (bit)	000 to FFF	Passing operation results between ladder instructions	Cleared
	RW, MW, AW (word)	000 to FF0		
	RL, ML, AL (long-word)	000 to FE0		
Keep relay	K (bit)	000 to FFF	Temporary retention of operation results	Remaining unchanged
	KW (word)	000 to FF0		
	KL (long-word)	000 to FE0		
ON-delay timer (contact, coil)	T (bit)	000 to 1FF	ON-delay timer	Cleared
	TW (word)	000 to 1F0		
	TL (long-word)	000 to 1E0		
ON-delay timer / set value	TS (word)	000 to 1FF		Remaining unchanged
ON-delay timer / count value	TC (word)	000 to 1FF		Cleared
One-shot timer (contact, coil)	U (bit)	000 to 0FF	One-shot timer	Cleared
	UW (word)	000 to 0F0		
	UL (long-word)	000 to 0E0		
One-shot timer / set value	US (word)	000 to 1FF		Remaining unchanged
One-shot timer / count value	UC (word)	000 to 1FF		Cleared

Table 2-1 Registers Usable in Arithmetic Functions

(2/3)

Function name	Register name (size)	Number	Use	Status after reset or power recovery
Up-down counter (contact, coil)	C (bit)	000 to 0FF	Counting if the condition is met	Remaining unchanged
	CW (word)	000 to 0F0		
	CL (long-word)	000 to 0E0		
Up-down counter / set value	CS (word)	000 to 1FF		
Up-down counter / count value	CC (word)	000 to 1FF		
Global link register	G (bit)	000 to FFF	Linkage between PLCs	Cleared
	GW (word)	000 to FF0		
	GL (long-word)	000 to FE0		
Nesting coil	N (bit)	000 to 0FF	Ladder subprogram call	Cleared
	NW (word)	000 to 0F0		
	NL (long-word)	000 to 0E0		
Process register	P (bit)	001 to 080	Task initiation	Cleared
	PW (word)	000 to 080		
	PL (long-word)	000 to 060		
Event register	E (bit)	000 to FFF	Event information output, analog/pulse counter	Cleared
	EW (word)	000 to FF0		
	EL (long-word)	000 to FE0		
Edge contact	V (bit)	000 to FFF	Edge detection	Cleared
	VW (word)	000 to FF0		
	VL (long-word)	000 to FE0		
Zee register	Z (bit)	000 to 3FF	Interrupt generation to host	Cleared
	ZW (word)	000 to 3F0		
	ZL (long-word)	000 to 3E0		
System register	S (bit)	000 to BFF	System status display	Initialized with initial value
	SW (word)	000 to BF0		
	SL (long-word)	000 to BE0		
Shared-data register between HI-FLOW and ladder	J, Q (bit)	000 to FFF	Data sharing between HI-FLOW and ladder	Cleared
	JW, QW(word)	000 to FF0		
	JL, QL (long-word)	000 to FE0		

## 2 ARITHMETIC FUNCTIONS

Table 2-1 Registers Usable in Arithmetic Functions

(3/3)

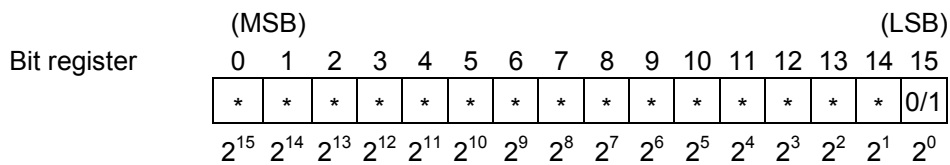
Function name	Register name (size)	Number	Use	Status after reset or power recovery
Extension internal register	LB (bit)	0000 to FFFF	Passing operation results between ladder instructions	Cleared
	LBW (word)	0000 to FFF0		
	LBL (long-word)	0000 to FFE0		
Converter-specific register	LR, LV (bit)	0000 to 0FFF	Exclusive use by the converter (users are not allowed to use it)	Cleared
	LRW, LVW (word)	0000 to 0FF0		
	LRL, LVL (long-word)	0000 to 0FE0		
Input/output register (reserved for future extension)	IW, OW (word)	000 to FFF	Future use	Cleared
Internal register	BD (long-word)	000 to 1FE	Indirect access	Remaining unchanged
	BW (word)	000 to 1FE		Depending on location
	BL (long-word)	000 to 1FE		
Function data register	DW (word)	000 to FFF	Constant data area	Remaining unchanged
	DL (long)	000 to FFE		
Function work register	FW (word)	000 to BFF	Work area	Remaining unchanged
	FL (long)	000 to BFE		
Extension function work register	LWW (word)	0000 to FFFF	Work area	Cleared
	LWL (long)	0000 to FFFE		
Long-word work register	LLL	0000 to 1FFF	Work area (long-word)	Cleared
Single-precision floating-point work register	LF	0000 to 1FFF	Floating-point arithmetics	Cleared
Backup word work register	LXW (word) (*)	0000 to 3FFF	Retention of data upon resetting	Remaining unchanged
	LXL (long)	0000 to 3FFE		
Backup long-word work register	LML (*)	0000 to 1FFF	Retention of data upon resetting	Remaining unchanged
Backup single-precision floating-point work register	LG (*)	0000 to 1FFF	Retention of data upon resetting	Remaining unchanged

(\*) The backup registers LX, LM, and LG require longer access time than do other registers, so they should be used only for the retention of initial values or data saving on error. Do not use them like ordinary non-backup registers.

### 2.3.2 Handling of bit registers

In arithmetic functions, such bit registers as X000 and RFF0 (i.e., those registers which are listed as “bit” in Table 2-1, “Registers Usable in Arithmetic Functions”) are handled as word data. In these registers, only the LSB is valid as data and all other bits are zero (0) in reading and invalid in writing.

The following is the data format of bit registers used in arithmetic functions:



\*: Each is zero (0) in reading and invalid in writing.

Example 1: MOV HFFFF -> R000, then MOV R000 -> FW000

The first transfer instruction “MOV” moves the value HFFFF (hexadecimal constant) to the bit register R000, then the second “MOV” moves the content of R000 to the word register FW000. The result is the value H0001 stored in FW000.

Example 2: MOV FW010 -> LB0000, then MOV LB0000 -> DW000

If the content of FW010 is H1234, the first “MOV” moves that content to the bit register LB0000, then the second “MOV” moves the content of LB0000 to DW000. The result is the value H0000 stored in DW000.

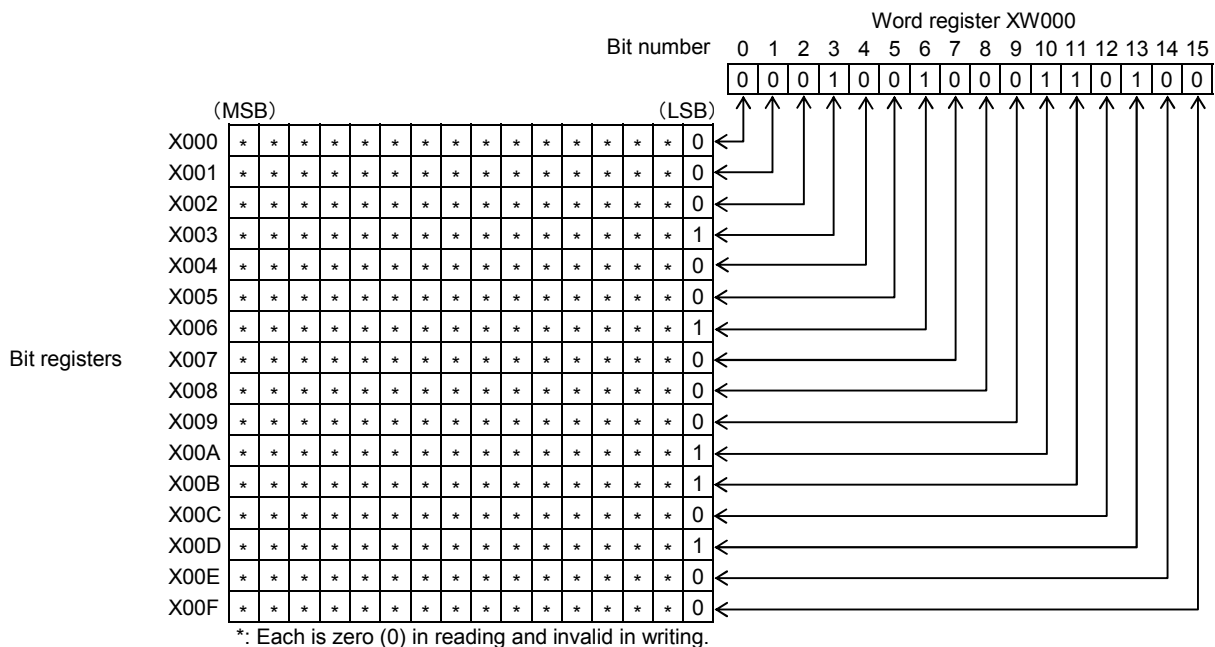


## 2 ARITHMETIC FUNCTIONS

### 2.3.3 Relationships between bit registers and word registers

Bit registers and word registers have relationships as shown below. The example below manifests that the bit registers X000 through X00F are in 16-to-1 correspondence with the word register XW000. The same correspondence also exists between X010 through X01F and XW010, between X020 through X02F and XW020, and so forth. The example below is the case of the X registers. The same is true with all other bit registers.

The bit and word registers actually share the same memory area, so the contents of a particular set of bit registers are completely synchronized with the content of the corresponding word register. This means that, right after data is written to a word register, reading some of the corresponding bit registers will bring you part of the data you have just written to that word register.



Long-word registers also have similar relationships with bit registers. For example, the bit registers X000 through X01F are in 32-to-1 correspondence with the long-word register XL000.

## 2.4 Inputs to Arithmetic Functions

Inputs to arithmetic functions are made through the input diagram of the arithmetic function. (For details on the arithmetic-function input diagram, refer to the “SOFTWARE MANUAL OPERATION S10V LADDER CHART For Windows® (Manual number SVE-3-131).”) Every input made has spaces inserted between the symbol and the first parameter, if any, and between the first and subsequent parameters, if any.

The number of parameters input depends of the type of arithmetic function. For details, see Section 2.6, “Details on the Instructions.”

function-name □ parameter □ parameter □ parameter ..... [Enter]

Example:

ADD □ RW000 □ FW000 □ FW000 [Enter]

### (1) Registers as inputs

Area in which settings can be made	Example of input	Remarks
I/O area (bit)	X000	Inputs are handled as word data in arithmetic functions. (Only LSB data is valid.)
I/O area (word)	YW000	The letter “W” denotes a word.
I/O area (long-word)	RL000	The letter “L” denotes a long word.
Function work register area	FW025	Work area
Function data register area	DW050	Constant-data area
Extension function work register area	LWW0000	The letter “W” denotes a word.
Long-word work register area	LLL0000	The letter “L” denotes a long word.
Single-precision floating-point work register area	LF0001	Used for single-precision floating-point operations.
Backup work register area (word, long-word, and floating)	LXW0000	The three types of registers, word, long-word, and floating, can be specified.
T, U, C set-value area	TS003	The letter “S” denotes a set value.
T, U, C count-value area	UC007	The letter “C” denotes a count value.
High-speed I/O (word) area	IW000	(Reserved for future extension)

- Each I/O area above is one of the registers named X, Y, R, M, A, K, T, U, C, G, N, P, E, V, Z, S, J, Q, LB, or LV.
- Numbers are input as 3- or 4-digit numbers.

## 2 ARITHMETIC FUNCTIONS

### (2) Constants as inputs (immediate)

#### (a) Input of decimal numbers

- ① Direct input of numeric value (positivedecimal number)  $\xrightarrow{\text{Positive number}}$  1 2 5 3
- ② Input of a sign (“+” or “-”) and then a numeric value  $\xrightarrow{\text{Negative number}}$  + 3 2 1 0 5  
- 1 2 5
- ③ Maximum number of digits that can be input:
- For word: 5  $\xrightarrow{\hspace{1.5cm}}$  (+/-) 3 2 7 6 7
  - For long-word: 10  $\xrightarrow{\hspace{1.5cm}}$  (+/-) 2 1 4 7 4 8 3 6 4 7

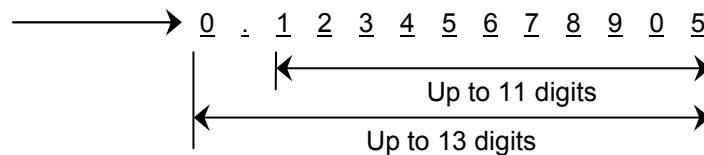
#### (b) Input of hexadecimal numbers

- ① Input of “H” and then a numeric value
- ② Maximum number of digits that can be input:
- For word: 4  $\xrightarrow{\hspace{1.5cm}}$  H 0 5 F 3
  - For long word: 8  $\xrightarrow{\hspace{1.5cm}}$  H 1 2 3 4 A B C D

#### (c) Input of floating (single-precision floating-point) data

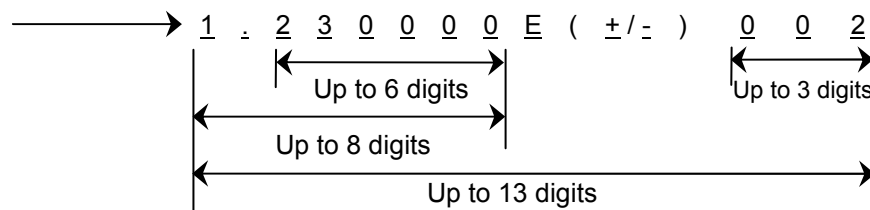
##### ① Input without exponent

Up to 13 digits (including the decimal point), and up to 11 digits after the decimal point:



##### ② Input with exponent

Up to eight digits as the mantissa, up to six digits after the decimal point in the mantissa, and up to three digits in the exponent:



Note: When you input a floating constant, be sure to enter the decimal point. If the decimal point is omitted, the input value will not be recognized as floating data and will cause an input error.

## (3) Specification of indices in arithmetic function instructions

## (a) Indexing using the “base register (index register)” format

Execution register address = base register number + index register content (expressed in units of words)

This indexing method uses as the execution address the location that is identified by the content of the index register relative to the register number of the base register.

The index registers that can be specified are all word-type registers.

Examples: DW020 (FW000), R400 (FW010)

In the case of “DW020 (FW000)”, if the content of FW000 is H0020, then:  
 $DW020 + H0020 \rightarrow DW040$ .

In the case of “R400 (FW010)”, if the content of FW010 is H0080, then:  
 $R400 + H0080 \rightarrow R480$ .

Note 1: If the content of FW000 is such a value as H0FF0 or H1200, which will result in a value greater than DWFFF (i.e., the maximum value of DW) when added to the number DW020, the normal operation of the instruction using the index is not guaranteed.

Note 2: Depending on the type of a register specified as the base register, the equation “base register number + index register content = execution register number” does not hold. For details, see the description of item (a) under “(4) Precautions in specifying an index in the arithmetic function instruction.”

When specifying the first number in a series, such as 000 or 0000, as the base register, the number may be omitted.

Examples: DW (FW000), XW (DW000)

In the case of “DW (FW000)” above, if the content of FW000 is H0020, then the indexed register points to DW020.

## (b) Indexing using the “reference type (indirect register)” format

Execution register address = indirect-register content

This indexing method is specified in the format:

Reference type (long-word register)

where the reference type is one of W (word), L (long-word), and F (floating).

The indirect register specified in this format is always a long-word register.

Examples: W (FL000), L (DL000)

In the case of “W (FL000)” above, the content of FL000 is used as an address. For example, if the content of FL000 is 000A0000, the content of the location 000A0000 is used as word data.

Note: If a given index specification causes a long-word access to a word boundary, it will result in an address error in the arithmetic function. For details, see the description of item (b) under “(4) Precautions in specifying an index in the arithmetic function instruction.”

## 2 ARITHMETIC FUNCTIONS

### (4) Precautions in specifying an index in the arithmetic function instruction

#### (a) Register numbers used in “base register (index register)” format

If any register as enumerated in the row Nos. 2, 3, 6, and 7 of the table below is used as the base register for indexing, the equation “base register number + index register content = execution register address” does not hold. Therefore, when you specify an index in an arithmetic function instruction, recall the information supplied in the table below and check that the register you have selected as the base register is actually usable.

No.	Register type	Register name	Execution register address
1	I/O register (bit)	X, Y, R, M, A, K, T, U, C, G, N, P, E, V, Z, S, J, Q, LB, LR, LV	Base register number + index register content (hexadecimal)
2	I/O register (word)	XW, YW, RW, MW, AW, KW, TW, UW, CW, GW, NW, PW, EW, VW, ZW, SW, JW, QW, LBW, LRW, LVW	Base register number + index register content (hexadecimal) × H0010 (hexadecimal)
3	I/O register (long-word)	XL, YL, RL, ML, AL, KL, TL, UL, CL, GL, NL, PL, EL, VL, ZL, SL, JL, QL, LBL, LRL, LVL	
4	Work register (word)	DW, FW, LWW, LXW	
5	Work register (long-word)	DL, FL, LWL, LXL	Base register number + index register content (hexadecimal)
6	Register used as long-word only	BD, LLL, LML	Base register number + index register content (hexadecimal) ÷ H0002 (hexadecimal)
7	Register used as floating only	LF, LG	

Examples: · G000 (DW001)

- This example results in the execution register address G010 if the content of DW001 is H0010.

- RW020 (FW000)

This example results in the execution register address RW320 if the content of FW000 is H0030.

- LLL0000 (FW000)

This example results in the execution register address LLL0020 if the content of FW000 is H0040.

## (b) Long-word/floating access using an index specification

In S10V of LPU module revision L (Ver.-Rev. 02-05) and earlier, long-word/floating accesses from arithmetic functions are limited to only the long-word boundaries. If an attempt is made to perform long-word/floating access to a word boundary by using an index specification, the LPU module detects it as an “arithmetic function address error” and goes down (i.e., the ERR LED is lit, the ladder is stopped, and the remote I/O operation is stopped), presenting a notification of the ladder program error.

Usually, the ladder chart system makes a check during circuit input for any attempt to perform long-word/floating access to a word boundary. If such an attempt is detected, the ladder chart system presents the message “A number of long register extends into word boundary.” and invalidates the input (in Ver.-Rev. 01-16 and later of the Ladder Chart System, the message “The long-word register with an odd number is specified.” appears). However, if such an attempt uses an index specification, the system cannot make a check, because the content of the index register remains indeterminate until the indexing is actually performed. Therefore, when specifying an index, care must be taken not to make long-word access to a word boundary with an inappropriate execution register address. Arithmetic function address errors can be prevented by giving due attention to programming. The table below shows preventive measures that can be used in programming.

Indexing method	Preventive measure
Base register (index register)	If a long-word register is specified as the base register, set an even number in the index register.
Reference type (indirect register)	If one of the letters “L” or “F” is specified as the reference type, be sure to use a number other than those listed in the table below, which will cause an error.

The table below is a list of all numbers that cause an arithmetic function address error at the time of long-word access (the following specifications are supported for LPU module revision M (Ver.-Rev. 02-06) and later).

Register type	Register name	Number causing an error
I/O register (long-word)	XL, YL, RL, ML, AL, KL, TL, UL, CL, GL, NL, PL, EL, VL, ZL, SL, JL, QL, LBL, LRL, LVL	Numbers that have an odd digit in the second least significant digit position, as in XL0 <b>3</b> 0 or LBL00 <b>3</b> 0 (boxed position)
Work register (long-word)	DL, FL, LWL, LXL	Numbers that have an odd digit in the least significant digit position, as in DL10 <b>7</b> or LWLFFF <b>9</b> (boxed position)

**NOTICE**

The LPU module stops due to “Invalid instruction detected.” This occurs if the long-word register having an odd number or a ladder program containing PSHO and POPO, which were created by the Ladder Chart System of Ver.-Rev. 01-16 or later, is sent to a Ladder Chart System having Ver.-Rev. 01-15 or earlier, or to an LPU having module revision L (Ver.-Rev. 02-05) or earlier using batch loading.

## 2 ARITHMETIC FUNCTIONS

---

Examples 1: Specified in “base register (index register)” format:

- XL000 (FW000)

If the content of FW000 above is H0003, the resulting execution register address is XL030, which causes an error. If, however, the content of FW000 is an even-numbered value, the resulting execution register address causes no error.

- LWL1000 (FW000)

If the content of FW000 above is H0005, the resulting execution register address is LWL1005, which causes an error. If, however, the content of FW000 is an even-numbered value, the resulting execution register address causes no error.

- LLL2000 (FW000)

If the content of FW000 above is H00FF, the resulting execution register address points to a word boundary, which causes an error. If, however, the content of FW000 is an even-numbered value, the resulting execution register address causes no error.

- LF0000 (FW000)

If the content of FW000 above is H007F, the resulting execution register address points to a word boundary, which causes an error. If, however, the content of FW000 is an even-numbered value, the resulting execution register address causes no error.

Examples 2: Specified in “reference type (indirect register)” format:

- L (RL000)

If the content of RL000 above is YW010, an error results. If, however, its content is YW000 or YW020, no error results.

When setting a YW address value in RL000 by using the AST function, specify an even digit in the second least significant digit position within the YW address value, that is, use the format YW\* $\Delta$ \*, where  $\Delta$  is an even digit. For example:

AST YW010 -> RL000: In this case, access using L (RL000) results in an error.

AST YW020 -> RL000: In this case, access using L (RL000) proceeds normally.

- L (DL004)

If the content of DL004 above is LWW0001, an error results. If, however, its content is LWW0002 or LWWFFFE, no error results.

When setting an LWW address value in DL004 by using the AST function, specify an even digit in the least significant digit position within the LWW address value, that is, use the format LWW\*\*\* $\Delta$ , where  $\Delta$  is an even digit. For example:

AST LWW0001 -> DL004: In this case, access using L (DL004) results in an error.

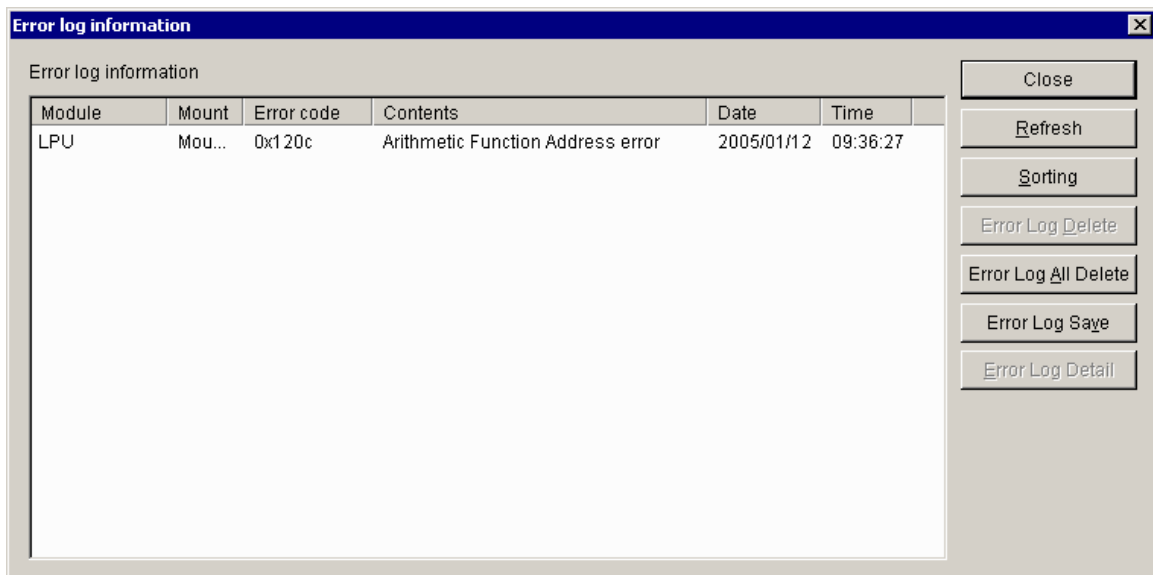
AST LWW0002 -> DL004: In this case, access using L (DL004) proceeds normally.

[Supplement] How to check if an arithmetic function address error has occurred by using the S10V base system:

If the ERR LED indicator of the LPU module is lit, you can identify the error as an arithmetic function address error or some other type of error by using the S10V base system. The procedure is as follows:

- ① Start the S10V base system.
- ② Click on the **Error Log** button.
- ③ The [Error log information] window appears. Check if the error code “0x120c” and message “Arithmetic Function Address error” are presented in an “LPU (module)” line.

An example of the [Error log information] window that reports on the occurrence of an “Arithmetic Function Address error” is given below.





## 2 ARITHMETIC FUNCTIONS

### 2.5 Arithmetic Functions

(1/5)

Major classification	Minor classification	Symbol	Unit of data processed	Process outline	Flags						Page
					X	E	P	N	Z	V	
Arithmetic instructions	Addition	ADD	Word	$(S) + (D) \rightarrow (R)$	-	-	-	-	-	●	2-26
			Long		-	-	-	-	-	●	
			Floating		-	●	-	-	-	-	
	Subtraction	SUB	Word	$(S) - (D) \rightarrow (R)$	-	-	-	-	-	●	2-30
			Long		-	-	-	-	-	●	
			Floating		-	●	-	-	-	-	
	+ 1	INC	Word	$(S) + 1 \rightarrow (S)$	-	-	-	-	-	●	2-34
			Long		-	-	-	-	-	●	
	- 1	DEC	Word	$(S) - 1 \rightarrow (S)$	-	-	-	-	-	●	2-36
			Long		-	-	-	-	-	●	
	Multiplication	MUL	Word	$(S) \times (D) \rightarrow (R)$	-	-	-	-	-	●	2-38
			Long		-	-	-	-	-	●	
			Floating		-	●	-	-	-	-	
	Division	DIV	Word	$(S) / (D) \rightarrow (R)$ (quotient)	-	●	-	-	-	●	2-42
Long			-		●	-	-	-	●		
Floating			-		●	-	-	-	-	-	
Remainder	MOD	Word	$(S) / (D) \rightarrow (R)$ (remainder)	-	●	-	-	-	●	2-46	
		Long		-	●	-	-	-	●		
Scale change	SCL	Word	$(S) \times (D1) / (D2) \rightarrow (R)$	-	●	-	-	-	●	2-48	
Logical instructions	Logical product	AND	Word	$(S) \wedge (D) \rightarrow (R)$	-	-	-	-	-	-	2-50
			Long		-	-	-	-	-	-	
	Logical sum	OR	Word	$(S) \vee (D) \rightarrow (R)$	-	-	-	-	-	-	2-52
			Long		-	-	-	-	-	-	
	Exclusive OR	EOR	Word	$(S) \oplus (D) \rightarrow (R)$	-	-	-	-	-	-	2-54
			Long		-	-	-	-	-	-	
Negation	NOT	Word	$\overline{(S)} \rightarrow (R)$	-	-	-	-	-	-	2-56	
		Long		-	-	-	-	-	-		
Comparison instructions	=	EQU	Word	When $(S) = (D)$ , $1 \rightarrow (R)$ . When $(S) \neq (D)$ , $0 \rightarrow (R)$ .	-	-	-	-	-	-	2-58
			Long		-	-	-	-	-	-	
			Floating		-	-	-	-	-	-	
	$\neq$	NEQ	Word	When $(S) \neq (D)$ , $1 \rightarrow (R)$ . When $(S) = (D)$ , $0 \rightarrow (R)$ .	-	-	-	-	-	-	2-60
			Long		-	-	-	-	-	-	
			Floating		-	-	-	-	-	-	
	>	GT	Word	When $(S) > (D)$ , $1 \rightarrow (R)$ . When $(S) \leq (D)$ , $0 \rightarrow (R)$ .	-	-	-	-	-	-	2-62
			Long		-	-	-	-	-	-	
			Floating		-	-	-	-	-	-	
	<	LT	Word	When $(S) < (D)$ , $1 \rightarrow (R)$ . When $(S) \geq (D)$ , $0 \rightarrow (R)$ .	-	-	-	-	-	-	2-64
			Long		-	-	-	-	-	-	
			Floating		-	-	-	-	-	-	

●: The value of this flag varies depending on the result of the operation performed.

-: The same value as before the performance of the operation is retained.

S: Source

D: Destination

R: Result

Major classification	Minor classification	Symbol	Unit of data processed	Process outline	Flags						Page
					X	E	P	N	Z	V	
Comparison instructions	≥	GE	Word	When (S) ≥ (D), 1 → (R) When (S) < (D), 0 → (R)	-	-	-	-	-	-	2-66
			Long								
			Floating								
	≤	LE	Word	When (S) ≤ (D), 1 → (R) When (S) > (D), 0 → (R)	-	-	-	-	-	-	2-68
			Long								
			Floating								
Test	TST	Word	Test (S) and set the P, N, and/or Z flags according the result.	-	-	●	●	●	-	2-70	
		Long									
		Floating									
Transfer instructions	Transfer	MOV	Word	(S) → (D)	-	-	-	-	-	-	2-72
			Long								
			Floating								
	Batch transfer	MOM	Word	(S, n) → (D)	-	-	-	-	-	-	2-74
			Long								
	Batch transfer of same data	INI	Word	(S) → (D, n)	-	-	-	-	-	-	2-76
			Long								
	Exchange	EXC	Word	(S) ↔ (D)	-	-	-	-	-	-	2-78
			Long								
	Write on FIFO basis	PSH	Word	(S) → FIFO table	-	-	-	-	-	-	2-80
	Read on FIFO basis	POP	Word	FIFO table → (D)	-	-	-	-	-	-	2-82
	Write on FIFO basis	PSHO	Word	(S) → FIFO table	-	-	-	-	-	-	2-84
	Read on FIFO basis	POPO	Word	FIFO table → (D)	-	-	-	-	-	-	2-86
	Address setting	AST	Long	Address of S → (D)	-	-	-	-	-	-	2-88
Search	SCH	Word	Search D for (S) in the range m (the number of steps to be searched), and matching number → (R)	-	-	-	-	-	-	-	2-90
		Long									
		Floating									
Conversion instructions	BIN → FLOAT	BTF	Word	BIN → FLOAT (S) → (R)	-	-	-	-	-	-	2-94
			Long								
	FLOAT → BIN	FTB	Word	FLOAT → BIN (S) → (R)	-	-	-	-	-	-	2-96
			Long								
	BIN → BCD	BTD	Word	BIN → BCD (S) → (R)	-	●	-	-	-	●	2-98
			Long								
	BCD → BIN	DTB	Word	BCD → BIN (S) → (R)	-	●	-	-	-	-	2-100
			Long								
BIN → 7SEG	SEG	Word	BIN → 7SEG (S) → (R)	-	-	-	-	-	-	2-102	
		Long									
BIN → ASCII	ASP	Word	BIN → ASCII (pack mode) (S) → (R)	-	-	-	-	-	-	2-104	
	ASU	Word	BIN → ASCII (unpack mode) (S) → (R)	-	-	-	-	-	-	2-106	

●: The value of this flag varies depending on the result of the operation performed.  
 -: The same value as before the performance of the operation is retained.  
 S: Source  
 D: Destination  
 R: Result  
 n: Number of words  
 m: Number of steps to be searched

## 2 ARITHMETIC FUNCTIONS

(3/5)

Major classification	Minor classification	Symbol	Unit of data processed	Process outline	Flags						Page	
					X	E	P	N	Z	V		
Conversion instructions	ASCII → BIN	APB	Word	ASCII → BIN (pack mode) (S) → (R)	-	●	-	-	-	-	2-108	
		AUB	Word	ASCII → BIN (unpack mode) (S) → (R)	-	●	-	-	-	-	2-110	
	SINGLE ↓ DOUBLE	STD	Word	(S) W → (R) L	-	-	-	-	-	-	2-112	
	DOUBLE ↓ SINGLE	DTS	Long	(S) L → (R) W	-	-	-	-	-	●	2-114	
	Absolute value	ABS	Word	(S)  → (R)	-	-	-	-	-	-	●	2-116
			Long									
			Floating									
	+/-	NEG	Word	-(S) → (R)	-	-	-	-	-	-	●	2-118
			Long									
			Floating									
Decode	DCD	Word	Numeric value n in (S) ... 1 → n-th bit in (R)	-	-	-	-	-	-	-	2-120	
		Long										
Encode	ECD	Word	Search for first 1-bit in (S), starting from MSB ... bit position n of the 1-bit found → (R)	-	●	-	-	-	-	-	2-122	
		Long										
Shift instructions	Logical shift right	LSR	Word	Shift (S) to the right by (D) bits → 0, (R)	-	-	-	-	-	-	2-124	
			Long									
	Logical shift left	LSL	Word	Shift (S) to the left by (D) bits → (R), 0	-	-	-	-	-	-	2-126	
			Long									
Arithmetic shift right	ASR	Word	Shift (S) to the right by (D) bits → MSB, (R)	-	-	-	-	-	-	2-128		
		Long										
Arithmetic shift left	ASL	Word	Shift (S) to the left by (D) bits → (R), V	-	-	-	-	-	-	●	2-130	
		Long										
Rotation instructions	Rotate right	ROR	Word	Rotate (S) to the right by (D) bits → (R)	-	-	-	-	-	-	2-132	
			Long									
Rotate left	ROL	Word	Rotate (S) to the left by (D) bits → (R)	-	-	-	-	-	-	-	2-134	
		Long										
Function processing instructions	LIMITER	LIM	Word	(D1) < (S) · (D1) → (R) (D2) ≤ (S) ≤ D1 · (S) → (R) (S) < (D2) · (D2) → (R)	-	●	-	-	-	-	2-136	
			Long									
			Floating									
	DEAD BAND	BND	Word	(D1) < (S) · (S) - (D1) → (R) (D2) ≤ (S) ≤ (D1) · 0 → (R) (S) < (D2) · (S) - (D2) → (R)	-	●	-	-	-	-	●	2-140
			Long									
			Floating									
	DEAD ZONE	ZON	Word	(S) > 0 · (S) + (D1) → (R) (S) = 0 · 0 → (R) (S) < 0 · (S) + (D2) → (R)	-	●	-	-	-	-	●	2-144
			Long									
			Floating									

●: The value of this flag varies depending on the result of the operation performed.

-: The same value as before the performance of the operation is retained.

S: Source

D: Destination

R: Result

Major classification	Minor classification	Symbol	Unit of data processed	Process outline	Flags						Page
					X	E	P	N	Z	V	
Function processing instructions	Square root	SQR	Word	$(S) \geq 0 \cdots \text{SQR}(S) \rightarrow (R)$ $(S) < 0 \cdots 0 \rightarrow (R)$	-	-	-	-	-	-	2-148
			Long								
			Floating								
	Sine	SIN	Floating	$\text{SIN}(S) \rightarrow (R)$	-	-	-	-	-	-	2-152
	Cosine	COS	Floating	$\text{COS}(S) \rightarrow (R)$	-	-	-	-	-	-	2-154
	Tangent	TAN	Floating	$\text{TAN}(S) \rightarrow (R)$	-	●	-	-	-	-	2-156
	Arc sine	ASIN	Floating	$\text{SIN}^{-1}(S) \rightarrow (R)$	-	●	-	-	-	-	2-158
	Arc cosine	ACOS	Floating	$\text{COS}^{-1}(S) \rightarrow (R)$	-	●	-	-	-	-	2-160
	Arc tangent	ATAN	Floating	$\text{TAN}^{-1}(S) \rightarrow (R)$	-	-	-	-	-	-	2-162
	Exponential	EXP	Floating	$\text{EXP}(S) \rightarrow (R)$	-	●	-	-	-	-	2-164
	Natural logarithm	LOG	Floating	$\text{LOG}(S) \rightarrow (R)$	-	●	-	-	-	-	2-166
	Maximum value	MAX	Word	$(S) \geq (R) \cdots (S) \rightarrow (R)$ $(S) < (R) \cdots (D) \rightarrow (R)$	-	-	-	-	-	-	2-168
Long											
Floating											
Minimum value	MIN	Word	$(S) \leq (R) \cdots (S) \rightarrow (R)$ $(S) > (R) \cdots (D) \rightarrow (R)$	-	-	-	-	-	-	2-170	
		Long									
		Floating									
Special instructions	Clear	XCLR	-	Clear X-area.	-	-	-	-	-	-	2-172
		YCLR	-	Clear Y-area.	-	-	-	-	-	-	2-172
		GCLR	-	Clear G-area.	-	-	-	-	-	-	2-172
		RCLR	-	Clear R-area.	-	-	-	-	-	-	2-172
		KCLR	-	Clear K-area.	-	-	-	-	-	-	2-172
		TCLR	-	Clear T-area and count value	-	-	-	-	-	-	2-172
		UCLR	-	Clear U-area and count value.	-	-	-	-	-	-	2-172
		CCLR	-	Clear C-area and count value.	-	-	-	-	-	-	2-172
		VCLR	-	Clear V-area.	-	-	-	-	-	-	2-172
		ECLR	-	Clear E-area.	-	-	-	-	-	-	2-172
		FCLR	-	Clear the operation result flags.	-	-	-	-	-	-	2-172

●: The value of this flag varies depending on the result of the operation performed.  
 -: The same value as before the performance of the operation is retained.  
 S: Source  
 D: Destination  
 R: Result

## 2 ARITHMETIC FUNCTIONS

(5/5)

Major classification	Minor classification	Symbol	Unit of data processed	Process outline	Flags						Page
					X	E	P	N	Z	V	
Jump instructions	Conditional	JT	–	If a given condition is met, jump to a specified label.	–	–	–	–	–	–	2-174
	Unconditional	JMP	–	Unconditionally jump to a specified label.	–	–	–	–	–	–	2-176
	Conditional jump to SEND	JSE	–	If a given condition is met, jump to the SEND (Sequence END) instruction.	–	–	–	–	–	–	2-178
Ethernet communication instructions	TCP communication	TOP	–	Open a TCP connection (client).	–	–	–	–	–	–	2-198
		TPOP	–	Open a TCP connection (server).	–	–	–	–	–	–	2-200
		TCLO	–	Close a TCP connection.	–	–	–	–	–	–	2-202
		TRCV	–	TCP reception.	–	–	–	–	–	–	2-204
		TSND	–	TCP transmission.	–	–	–	–	–	–	2-208
	UDP communication	UOP	–	Open UDP.	–	–	–	–	–	–	2-210
		UCLO	–	Close UDP.	–	–	–	–	–	–	2-212
		URCV	–	UDP reception.	–	–	–	–	–	–	2-214
USND		–	UDP transmission.	–	–	–	–	–	–	2-218	

–: The same value as before the performance of the operation is retained.

## 2.6 Details on the Instructions

Information in this section is concerning all available standard arithmetic function instructions and is organized as follows.

- (1) Input format  
Under this heading is shown the input format of each instruction.
- (2) Function  
Under this heading is provided a description of each instruction's function.
- (3) Data types  
Under this heading are listed the types of data that can be specified as parameters to each instruction.  
Example:

This portion shows whether such registers as DW000 and such constants as H0001 may be used with the instruction or not.

This portion shows whether such registers as LLL0000 and such constants as H04231556 may be used with the instruction or not.

This portion shows whether such registers as LF0000 and such constants as 1.12E-002 may be used with the instruction or not.

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D	√	√	√	√	√	√	√
R	√	–	√	–	√	–	√

√: May be specified.  
–: May not be specified.

If a register may be specified, this portion shows whether an index may be specified or not.

According to the above sample table, users can specify, as S (Source) and D (Destination), addresses of such data as word, long-word, and floating, including index specifications, and can specify constants of those types. In addition, they can also specify as R (result) such data registers as word, long-word, and floating, including index specifications.

Note: Bit I/O areas, such as R000 and Y1FF, are handled as word data in arithmetic functions. In these cases, only the LSB is valid and all the other bits are zero (0) in reading and invalid in writing. For details, see Subsection 2.3.2, "Handling of bit registers."

- (4) Example program  
Under this heading is shown a simple ladder program using each instruction and its operation.
- (5) Error handling  
Under this heading is described what processing will be done if an error occurs. The operation result flag(s) reflecting the error are also shown under this heading.

# ADD ADDITION

---

## (1) Input format

ADD S + D -> R
----------------

where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register or a constant.

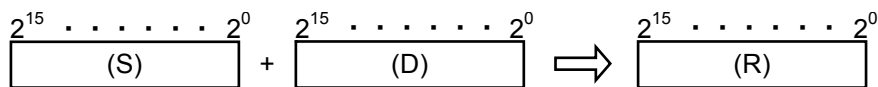
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “+” and “->” may be omitted.

## (2) Function

### ● Addition of word data

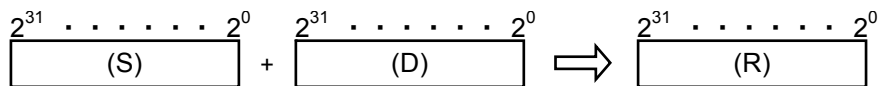
The ADD instruction adds a 16-bit data value specified in Source (S) and another 16-bit data value specified in Destination (D) together and stores the result in Result (R):



where the values that may be specified in Source (S) and Destination (D) and stored in Result (R) are in the range -32768 to 32767.

### ● Addition of long-word data

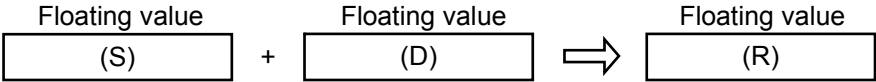
The ADD instruction adds a 32-bit data value specified in Source (S) and another 32-bit data value specified in Destination (D) together and stores the result in Result (R):



where the values that may be specified in Source (S) and Destination (D) and stored in Result (R) are in the range -2147483648 to 2147483647.

● Addition of floating data

The ADD instruction adds a floating data value specified in Source (S) and another floating data value specified in Destination (D) together and stores the result in Result (R):



where the values that may be specified in Source (S) and Destination (D) and stored in Result (R) are in the following range:

$$0, \pm 2^{-126} \text{ to } \pm 2^{128}$$

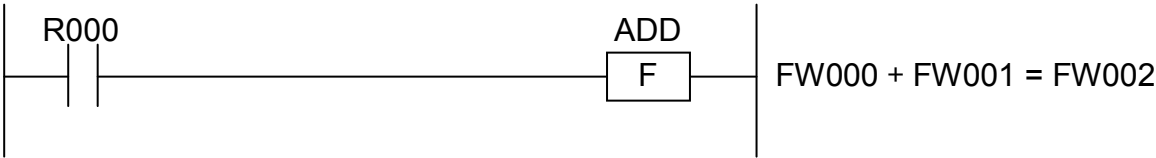
(3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D	√	√	√	√	√	√	√
R	√	–	√	–	√	–	√

√: May be specified.  
 –: May not be specified.

The types of S, D, and R must be the same (i.e., either word, long-word, or floating). If any one of them is of a different type, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the ADD instruction adds the contents of FW000 and FW001 together and stores the result in FW002.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	↕

where:

V: When the type of given data is word:

- Set to 0 if Result (R) is in the range -32768 to 32767; otherwise, set to 1.

When it is long-word:

- Set to 0 if Result (R) is in the range -2147483648 to 2147483647; otherwise, set to 1.

When it is floating:

- Not affected by the result of the operation performed; it remains unchanged.

E: When the type of given data is word or long-word:

- Not affected by the result of the operation performed; it remains unchanged.

When it is floating:

- Set to 1 if Result (R) is a non-zero value and out of the range shown below; otherwise, set to 0.

$$\pm 2^{-126} \text{ to } \pm 2^{128}$$

All the other flags then V and E remain unchanged.

- If an overflow occurs in the operation, one of the following full-scale values will be stored in Result (R):

	In case of a positive overflow:	In case of a negative overflow:
Word	H7FFF	H8000
Long-word	H7FFFFFFF	H80000000
Floating	+3.402823E38	-3.402823E38

If a floating value causes an overflow, the V-flag is not set. (The V-flag is set only if a word or long-word value causes an overflow.)

- If the E-flag is set, the content of Result (R) remains unchanged.
- If a floating value causes an underflow, a value of zero (0) with correct sign will be stored in Result (R), the operation result flags remaining unchanged.

**This Page Intentionally Left Blank**

# SUB SUBTRACTION

## (1) Input format

SUB S - D -> R
----------------

where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register or a constant.

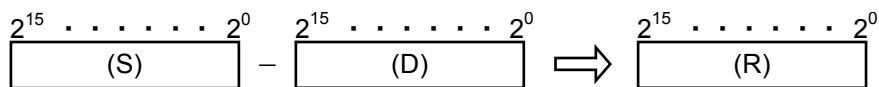
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “-” and “->” may be omitted.

## (2) Function

### ● Subtraction of word data

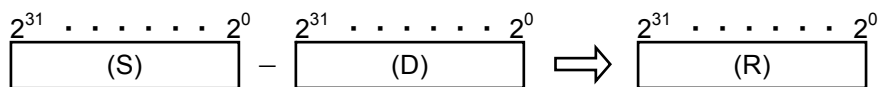
The SUB instruction subtracts a 16-bit data value specified in Destination (D) from another 16-bit data value specified in Source (S) and stores the result in Result (R):



where the values that may be specified in Source (S) and Destination (D) and stored in Result (R) are in the range -32768 to 32767.

### ● Subtraction of long-word data

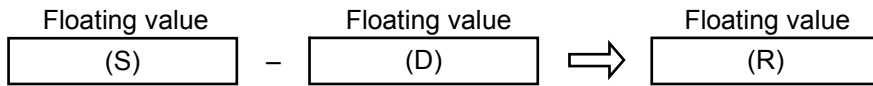
The SUB instruction subtracts a 32-bit data value specified in Destination (D) from another 32-bit data value specified in Source (S) and stores the result in Result (R):



where the values that may be specified in Source (S) and Destination (D) and stored in Result (R) are in the range -2147483648 to 2147483647.

● Subtraction of floating data

The SUB instruction subtracts a floating data value specified in Destination (D) from another floating data value specified in Source (S) and stores the result in Result (R):



where the values that may be specified in Source (S) and Destination (D) and stored in Result (R) are in the following range:

$$0, \pm 2^{-126} \text{ to } \pm 2^{128}$$

(3) Data types

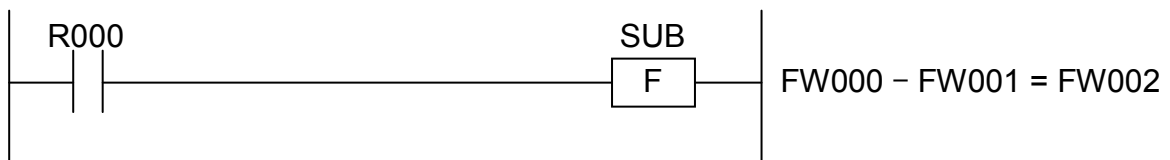
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D	√	√	√	√	√	√	√
R	√	–	√	–	√	–	√

√: May be specified.

–: May not be specified.

The types of S, D, and R must be the same (i.e., either word, long-word, or floating). If any one of them is of a different type, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the SUB instruction subtracts the content of FW001 from that of FW000 and stores the result in FW002.

## SUB SUBTRACTION

---

### (5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	↕

where:

V: When the type of given data is word:

- Set to 0 if Result (R) is in the range -32768 to 32767; otherwise, set to 1.

When it is long-word:

- Set to 0 if Result (R) is in the range -2147483648 to 2147483647; otherwise, set to 1.

When it is floating:

- Not affected by the result of the operation performed; it remains unchanged.

E: When the type of given data is word or long-word:

- Not affected by the result of the operation performed; it remains unchanged.

When it is floating:

- Set to 1 if Result (R) is a non-zero value and out of the range shown below; otherwise, set to 0.

$$\pm 2^{-126} \text{ to } \pm 2^{128}$$

All the other flags then V and E remain unchanged.

- If an overflow occurs in the operation, one of the following full-scale values will be stored in Result (R):

	In case of a positive overflow:	In case of a negative overflow:
Word	H7FFF	H8000
Long-word	H7FFFFFFF	H80000000
Floating	+3.402823E38	-3.402823E38

If a floating value causes an overflow, the V-flag is not set. (The V-flag is set only if a word or long-word value causes an overflow.)

- If the E-flag is set, the content of Result (R) remains unchanged.
- If a floating value causes an underflow, a value of zero (0) with correct sign will be stored in Result (R), the operation result flags remaining unchanged.

**This Page Intentionally Left Blank**

# INC +1 (INCREMENTATION)

## (1) Input format



where:

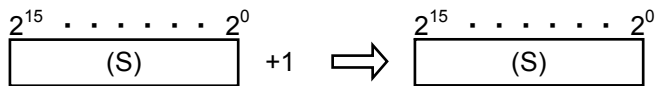
S: A data storage register to store a data value to be incremented.

Note: Spaces must be inserted between the function name and parameter.

## (2) Function

- Incrementation of word data

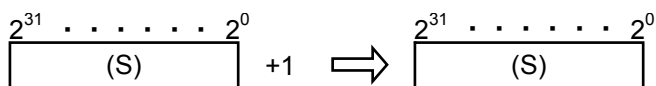
The INC instruction increments a 16-bit data value specified in Source (S) by one (1):



where the values that may be specified and stored in Source (S) are in the range -32768 to 32767.

- Incrementation of long-word data

The INC instruction increments a 32-bit data value specified in Source (S) by one (1):



where the values that may be specified and stored in Source (S) are in the range -2147483648 to 2147483647.

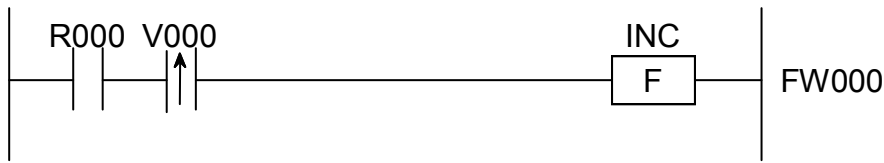
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	–	√	–	–	–	√

√: May be specified.

–: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) makes a transition from OFF to ON state, the INC instruction increments the content of FW000 by one (1) only once.

(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	↕

where:

V: When the type of given data is word:

- Set to 0 if Result (R) is in the range -32768 to 32767; otherwise, set to 1.

When it is long-word:

- Set to 0 if Result (R) is in the range -2147483648 to 2147483647; otherwise, set to 1.

All the other flags then V remain unchanged.

- If an overflow occurs in the operation, one of the following full-scale values will be stored in Result (R):

Word	Long-word
H7FFF	H7FFFFFFF



# DEC -1 (DECREMENTATION)

## (1) Input format



where:

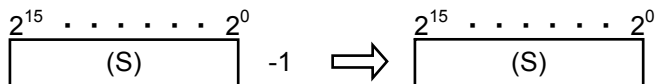
S: A data storage register to store a data value to be decremented.

Note: Spaces must be inserted between the function name and parameter.

## (2) Function

### ● Decrementation of word data

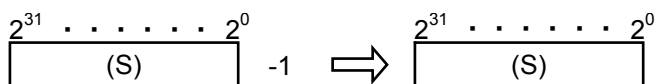
The DEC instruction decrements a 16-bit data value specified in Source (S) by one (1):



where the values that may be specified and stored in Source (S) are in the range -32768 to 32767.

### ● Decrementation of long-word data

The DEC instruction decrements a 32-bit data value specified in Source (S) by one (1):



where the values that may be specified and stored in Source (S) are in the range -2147483648 to 2147483647.

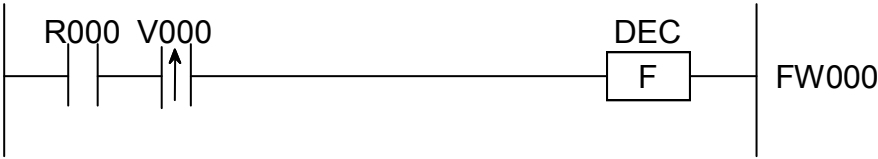
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	–	√	–	–	–	√

√: May be specified.

–: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) makes a transition from OFF to ON state, the DEC instruction decrements the content of FW000 by one (1) only once.

(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	↕

where:

V: When the type of given data is word:

- Set to 0 if Result (R) is in the range -32768 to 32767; otherwise, set to 1.

When it is long-word:

- Set to 0 if Result (R) is in the range -2147483648 to 2147483647; otherwise, set to 1.

All the other flags then V remain unchanged.

- If an overflow occurs in the operation, one of the following full-scale values will be stored in Result (R):

Word	Long-word
H8000	H80000000

# MUL MULTIPLICATION

## (1) Input format

MUL S \* D -> R

where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register or a constant.

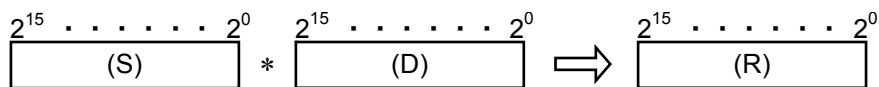
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “\*” and “->” may be omitted.

## (2) Function

### ● Multiplication of word data

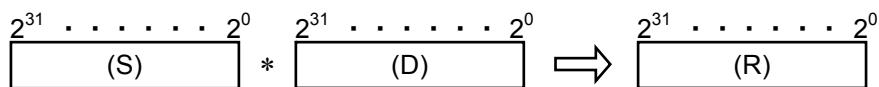
The MUL instruction multiplies a 16-bit data value specified in Source (S) and another 16-bit data value specified in Destination (D) and stores the result in Result (R):



where the values that may be specified in Source (S) and Destination (D) and stored in Result (R) are in the range -32768 to 32767.

### ● Multiplication of long-word data

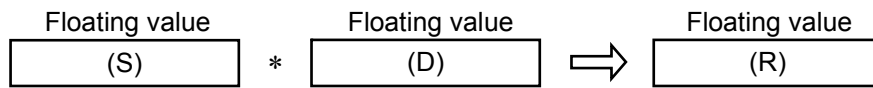
The MUL instruction multiplies a 32-bit data value specified in Source (S) and another 32-bit data value specified in Destination (D) and stores the result in Result (R):



where the values that may be specified in Source (S) and Destination (D) and stored in Result (R) are in the range -2147483648 to 2147483647.

● Multiplication of floating data

The MUL instruction multiplies a floating data value specified in Source (S) and another floating data value specified in Destination (D) and stores the result in Result (R):



where the values that may be specified in Source (S) and Destination (D) and stored in Result (R) are in the following range:

$$0, \pm 2^{-126} \text{ to } \pm 2^{128}$$

(3) Data types

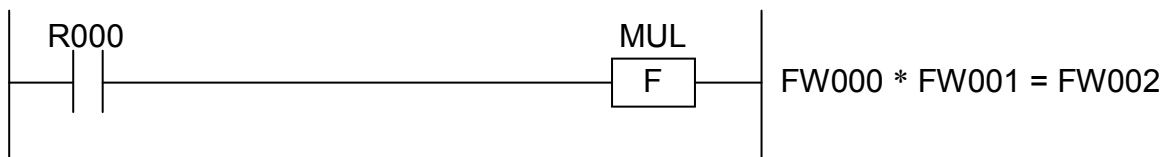
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D	√	√	√	√	√	√	√
R	√	–	√	–	√	–	√

√: May be specified.

–: May not be specified.

The types of S, D, and R must be the same (i.e., either word, long-word, or floating). If any one of them is of a different type, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the MUL instruction multiplies the content of FW000 and that of FW001 and stores the result in FW002.

## MUL MULTIPLICATION

---

### (5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	↕

where:

V: When the type of given data is word:

- Set to 0 if Result (R) is in the range -32768 to 32767; otherwise, set to 1.

When it is long-word:

- Set to 0 if Result (R) is in the range -2147483648 to 2147483647; otherwise, set to 1.

When it is floating:

- Not affected by the result of the operation performed; it remains unchanged.

E: When the type of given data is word or long-word:

- Not affected by the result of the operation performed; it remains unchanged.

When it is floating:

- Set to 1 if Result (R) is a non-zero value and out of the range shown below; otherwise, set to 0.

$$\pm 2^{-126} \text{ to } \pm 2^{128}$$

All the other flags then V and E remain unchanged.

- If an overflow occurs in the operation, one of the following full-scale values will be stored in Result (R):

	In case of a positive overflow:	In case of a negative overflow:
Word	H7FFF	H8000
Long-word	H7FFFFFFF	H80000000
Floating	+3.402823E38	-3.402823E38

If a floating value causes an overflow, the V-flag is not set. (The V-flag is set only if a word or long-word value causes an overflow.)

- If the E-flag is set, the content of Result (R) remains unchanged.
- If a floating value causes an underflow, a value of zero (0) with correct sign will be stored in Result (R), the operation result flags remaining unchanged.

**This Page Intentionally Left Blank**

# DIV DIVISION

## (1) Input format



where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register or a constant.

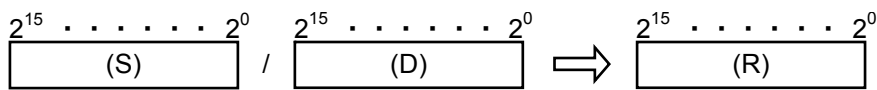
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “/” and “->” may be omitted.

## (2) Function

### ● Division of word data

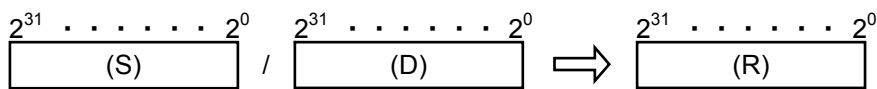
The DIV instruction divides a 16-bit data value specified in Source (S) by another 16-bit data value specified in Destination (D) and stores the result (quotient) in Result (R):



where the values that may be specified in Source (S) and Destination (D) and stored in Result (R) are in the range -32768 to 32767.

### ● Division of long-word data

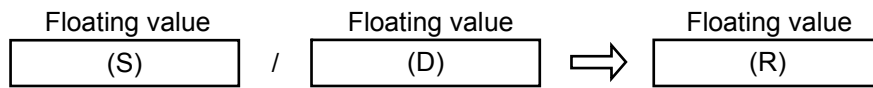
The DIV instruction divides a 32-bit data value specified in Source (S) by another 32-bit data value specified in Destination (D) and stores the result (quotient) in Result (R):



where the values that may be specified in Source (S) and Destination (D) and stored in Result (R) are in the range -2147483648 to 2147483647.

● Multiplication of floating data

The DIV instruction divides a floating data value specified in Source (S) by another floating data value specified in Destination (D) and stores the result in Result (R):



where the values that may be specified in Source (S) and Destination (D) and stored in Result (R) are in the following range:

$$0, \pm 2^{-126} \text{ to } \pm 2^{128}$$

(3) Data types

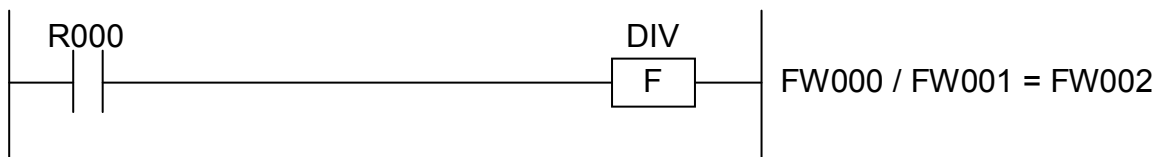
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D	√	√	√	√	√	√	√
R	√	–	√	–	√	–	√

√: May be specified.

–: May not be specified.

The types of S, D, and R must be the same (i.e., either word, long-word, or floating). If any one of them is of a different type, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the DIV instruction divides the content of FW000 by that of FW001 and stores the result (quotient) in FW002.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	↕

where:

V: When the type of given data is word:

- Set to 1 if Result (R) equals 32768; otherwise, set to 0.

When it is long-word:

- Set to 1 if Result (R) equals 2147483648; otherwise, set to 0.

When it is floating:

- Not affected by the result of the operation performed; it remains unchanged.

E: When the type of given data is word or long-word:

- Set to 1 if D equals 0; otherwise, set to 0.

When it is floating:

- Set to 1 if Result (R) is a non-zero value and out of the range shown below; otherwise, set to 0.

$$\pm 2^{-126} \text{ to } \pm 2^{128}$$

All the other flags then V and E remain unchanged.

- If an attempt is made to divide a value by zero (0), the error (E) flag is set, with the overflow (V) flag reset. The content of Result (R) remains unchanged.
- If an overflow occurs in the operation, one of the following full-scale values will be stored in Result (R):

	In case of a positive overflow:	In case of a negative overflow:
Word	H7FFF	H8000
Long-word	H7FFFFFFF	H80000000
Floating	+3.402823E38	-3.402823E38

If a floating value causes an overflow, the V-flag is not set. (The V-flag is set only if a word or long-word value causes an overflow.)

- If the E-flag is set, the content of Result (R) remains unchanged.
- If a floating value causes an underflow, a value of zero (0) with correct sign will be stored in Result (R), the operation result flags remaining unchanged.

# MOD REMAINDER

## (1) Input format

MOD S % D -> R
----------------

where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register or a constant.

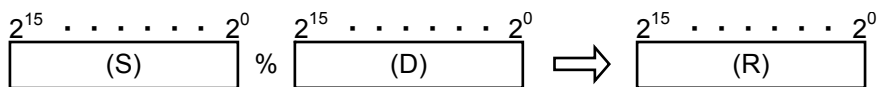
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “%” and “->” may be omitted.

## (2) Function

### ● Remainder division of word data

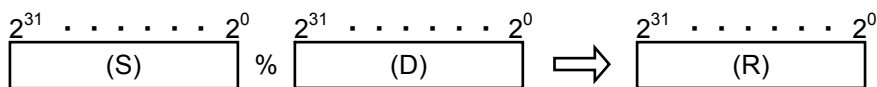
The MOD instruction divides a 16-bit data value specified in Source (S) by another 16-bit data value specified in Destination (D) and stores the resulting remainder in Result (R):



where the values that may be specified in Source (S) and Destination (D) and stored in Result (R) are in the range -32768 to 32767.

### ● Remainder division of long-word data

The MOD instruction divides a 32-bit data value specified in Source (S) by another 32-bit data value specified in Destination (D) and stores the resulting remainder in Result (R):



where the values that may be specified in Source (S) and Destination (D) and stored in Result (R) are in the range -2147483648 to 2147483647.

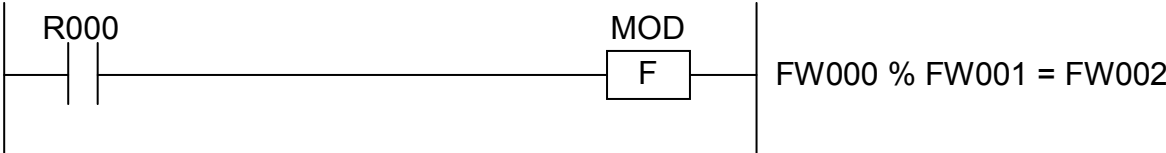
(3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	-	-	√
D	√	√	√	√	-	-	√
R	√	-	√	-	-	-	√

√: May be specified.  
 -: May not be specified.

The types of S, D, and R must be the same (i.e., either word or long-word). If any one of them is of a different type, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the MOD instruction divides the content of FW000 by that of FW001 and stores the resulting remainder in FW002.

(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	↕

where:

- V: When the type of given data is word:
    - Set to 1 if the resulting quotient equals 32768; otherwise, set to 0.
 When it is long-word:
    - Set to 1 if the resulting quotient equals 2147483648; otherwise, set to 0.
  - E: Set to 1 if Destination (D) equals 0; otherwise, set to 0.
- All the other flags then V and E remain unchanged.

- If an attempt is made to divide a value by zero (0), the error (E) flag is set, with the overflow (V) flag reset. The content of Result (R) remains unchanged.
- If an overflow occurs in the operation, a value of zero (0) will be stored in Result (R).

# SCL SCALE CHANGE

## (1) Input format

SCL S : D1 : D2 -> R

where:

S: (Source) is a source storage register or a constant.

D1, D2: (Destination 1, Destination 2) each is a destination storage register or a constant.

R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

The SCL instruction multiplies a 16-bit data value specified in Source (S) and the value of Destination 1 (D1) divided by Destination 2 (D2) and stores the result in Result (R):

$$\begin{array}{|c|} \hline 2^{15} \dots 2^0 \\ \hline \boxed{\text{(S)}} \\ \hline \end{array} * \begin{array}{|c|} \hline 2^{15} \dots 2^0 \\ \hline \boxed{\text{(D1)}} \\ \hline \end{array} / \begin{array}{|c|} \hline 2^{15} \dots 2^0 \\ \hline \boxed{\text{(D2)}} \\ \hline \end{array} \Rightarrow \begin{array}{|c|} \hline 2^{15} \dots 2^0 \\ \hline \boxed{\text{(R)}} \\ \hline \end{array}$$

where the values that may be specified in Source (S), Destination 1 (D1), and Destination 2 (D2) and stored in Result (R) are in the range -32768 to 32767.

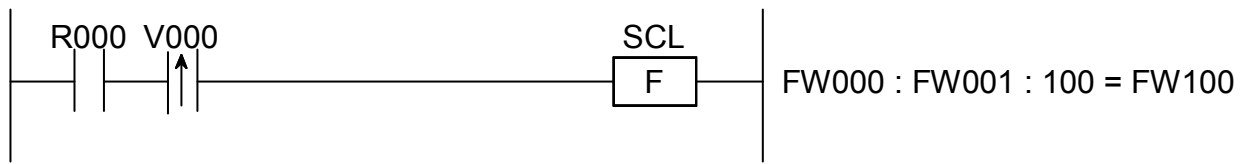
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	–	–	–	–	√
D1	√	√	–	–	–	–	√
D2	√	√	–	–	–	–	√
R	√	–	–	–	–	–	√

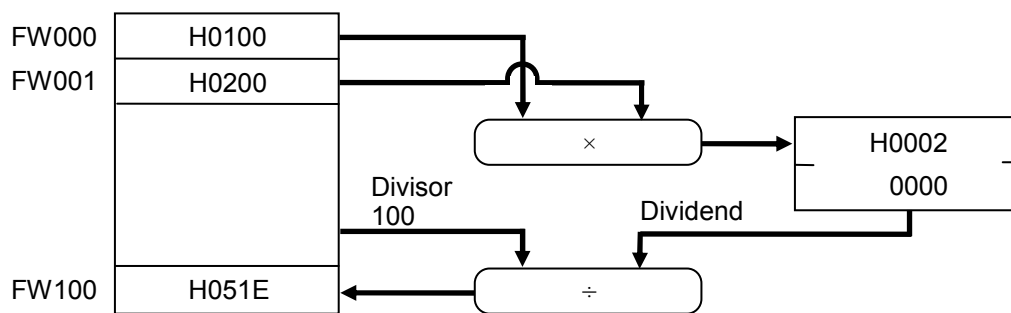
√: May be specified.

–: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) makes a transition from OFF to ON state, the SCL instruction changes the scale for the content of FW000 only once and stores the result in FW100.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	↕

where:

V: Set to 0 if Result (R) is in the range -32768 to 32767; otherwise, set to 1.

E: Set to 1 if Destination 2 (D2) equals 0; otherwise, set to 0.

All the other flags then V and E remain unchanged.

- If an attempt is made to divide a value by zero (0), the error (E) flag is set, with the overflow (V) flag reset. The content of Result (R) remains unchanged.
- If an overflow occurs in the operation, one of the following full-scale values will be stored in Result (R):

In case of a positive overflow:	H7FFF
In case of a negative overflow:	H8000

# AND LOGICAL PRODUCT

---

## (1) Input format

AND S : D -> R
----------------

where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register or a constant.

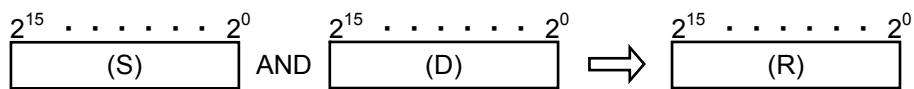
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

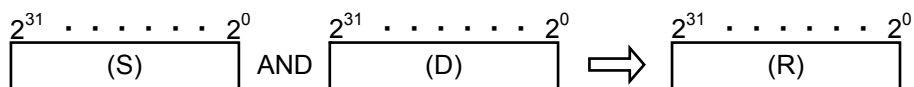
### ● Logical product of word data

The AND instruction computes the logical product of a 16-bit data value specified in Source (S) and another 16-bit data value specified in Destination (D) and stores the result in Result (R):



### ● Logical product of long-word data

The AND instruction computes the logical product of a 32-bit data value specified in Source (S) and another 32-bit data value specified in Destination (D) and stores the result in Result (R):



(3) Data types

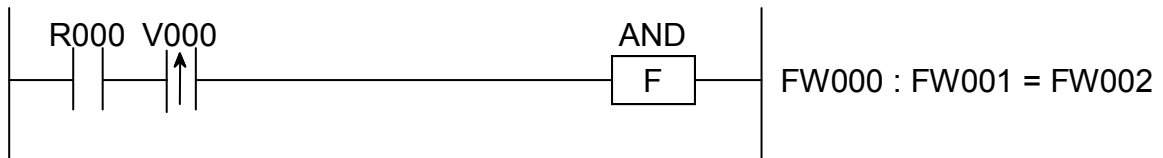
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	-	-	√
D	√	√	√	√	-	-	√
R	√	-	√	-	-	-	√

√: May be specified.

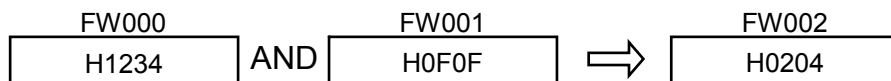
-: May not be specified.

The types of S, D, and R must be the same (i.e., either word or long-word). If any one of them is of a different type, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) makes a transition from OFF to ON state, the AND instruction computes the logical product of the contents of FW000 and FW001 only once and stores the result in FW002.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.



# OR LOGICAL SUM

---

## (1) Input format



where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register or a constant.

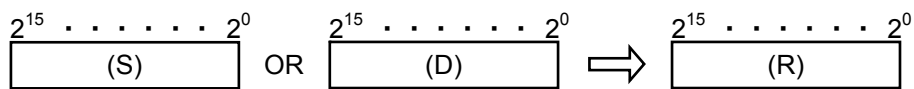
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

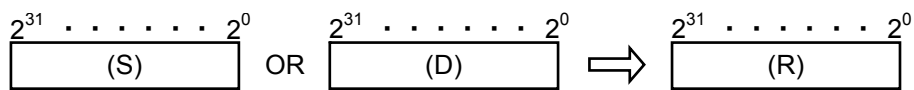
### ● Logical sum of word data

The OR instruction computes the logical sum of a 16-bit data value specified in Source (S) and another 16-bit data value specified in Destination (D) and stores the result in Result (R):



### ● Logical sum of long-word data

The OR instruction computes the logical sum of a 32-bit data value specified in Source (S) and another 32-bit data value specified in Destination (D) and stores the result in Result (R):



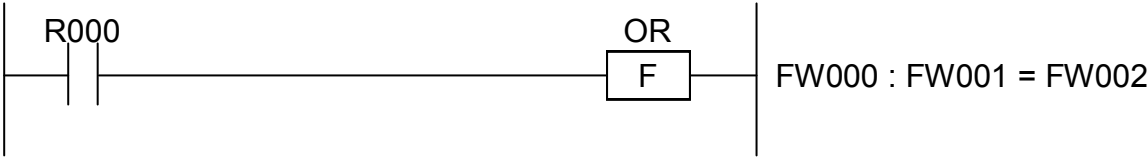
(3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	-	-	√
D	√	√	√	√	-	-	√
R	√	-	√	-	-	-	√

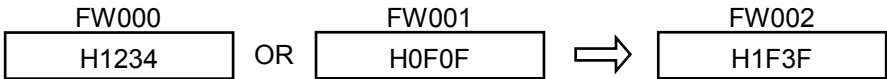
√: May be specified.  
 -: May not be specified.

The types of S, D, and R must be the same (i.e., either word or long-word). If any one of them is of a different type, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the OR instruction computes the logical sum of the contents of FW000 and FW001 and stores the result in FW002.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# EOR EXCLUSIVE OR

## (1) Input format

EOR S : D -> R
----------------

where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register or a constant.

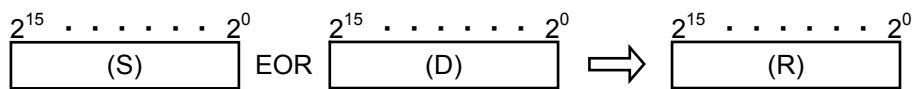
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

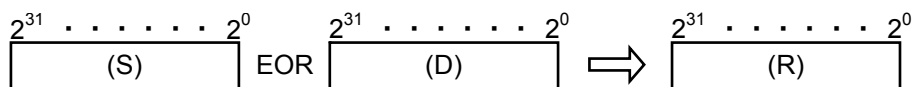
- Exclusive logical sum of word data

The EOR instruction computes the exclusive logical sum of a 16-bit data value specified in Source (S) and another 16-bit data value specified in Destination (D) and stores the result in Result (R):



- Exclusive logical sum of long-word data

The EOR instruction computes the exclusive logical sum of a 32-bit data value specified in Source (S) and another 32-bit data value specified in Destination (D) and stores the result in Result (R):



(3) Data types

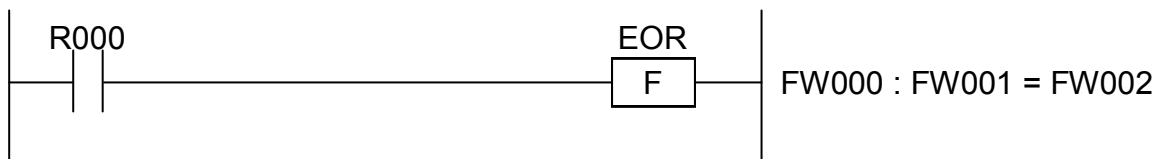
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	-	-	√
D	√	√	√	√	-	-	√
R	√	-	√	-	-	-	√

√: May be specified.

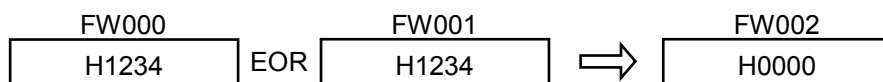
-: May not be specified.

The types of S, D, and R must be the same (i.e., either word or long-word). If any one of them is of a different type, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the EOR instruction computes the exclusive logical sum of the contents of FW000 and FW001 and stores the result in FW002.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# NOT NEGATION

---

## (1) Input format

NOT S -> R
------------

where:

S: (Source) is a source storage register.

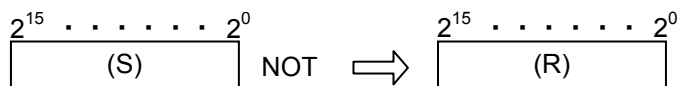
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

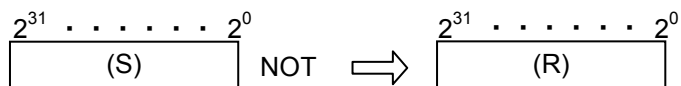
- Negation of word data

The NOT instruction inverts the bits of a 16-bit data value specified in Source (S) and stores the result in Result (R):



- Negation of long-word data

The NOT instruction inverts the bits of a 32-bit data value specified in Source (S) and stores the result in Result (R):



(3) Data types

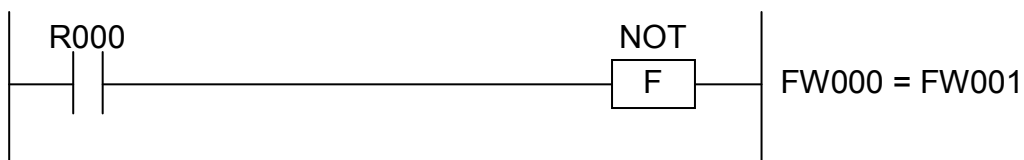
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	–	√	–	–	–	√
R	√	–	√	–	–	–	√

√: May be specified.

–: May not be specified.

The types of S and R must be the same (i.e., either word or long-word). If the two are of different types, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the NOT instruction inverts the content bits of FW000 and stores the inverted bits in FW001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
–	–	–	–	–	–

All the above flags remain unchanged.

# EQU = (EQUAL)

## (1) Input format



where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register or a constant.

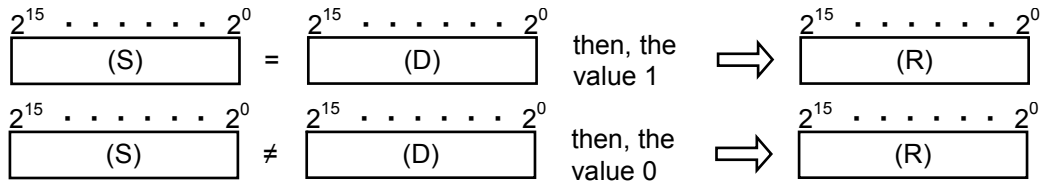
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

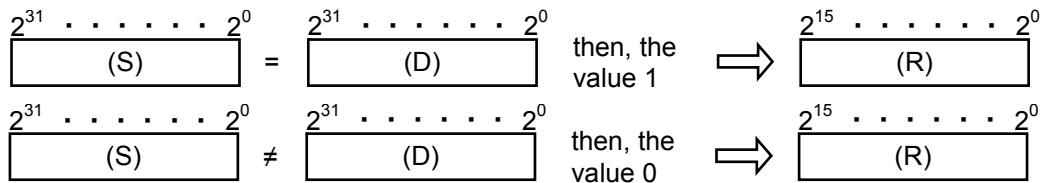
### ● Comparison of word data

The EQU instruction compares two 16-bit data values specified in Source (S) and Destination (D), respectively. If they are equal, the instruction then stores the value 1 in Result (R); otherwise, it stores the value 0 in it.



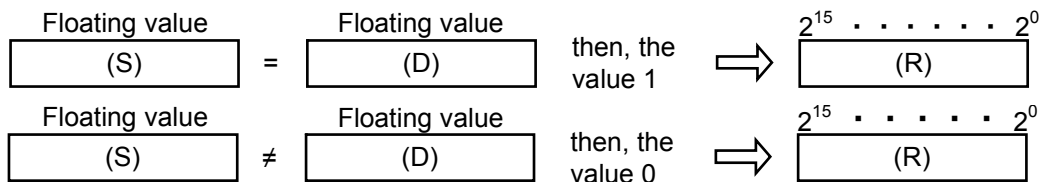
### ● Comparison of long-word data

The EQU instruction compares two 32-bit data values specified in Source (S) and Destination (D), respectively. If they are equal, the instruction then stores the value 1 in Result (R); otherwise, it stores the value 0 in it.



### ● Comparison of floating data

The EQU instruction compares two floating data values specified in Source (S) and Destination (D), respectively. If they are equal, the instruction then stores the value 1 in Result (R); otherwise, it stores the value 0 in it.



Note: Care must be taken when using this instruction for comparison of floating data values. Any two such values, which are actually equal, may be compared as not equal, due to error contained in those values.

(3) Data types

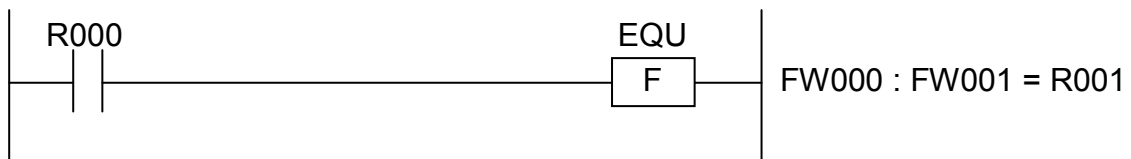
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D	√	√	√	√	√	√	√
R	√	–	–	–	–	–	√

√: May be specified.

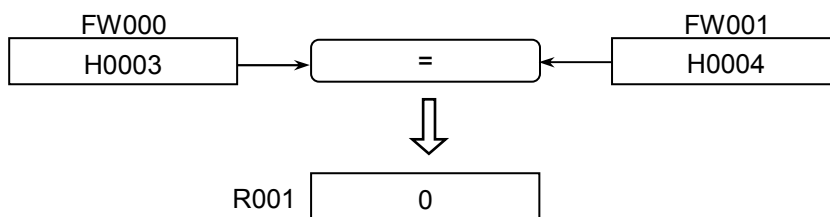
–: May not be specified.

The types of S and D must be the same (i.e., either word, long-word, or floating). If the two are of different types, an input error will result. The type of R must always be word.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the EQU instruction compares the contents of FW000 and FW001 and stores the result in R001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
–	–	–	–	–	–

All the above flags remain unchanged.



# NEQ $\neq$ (NOT EQUAL)

## (1) Input format

NEQ S : D -> R

where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register or a constant.

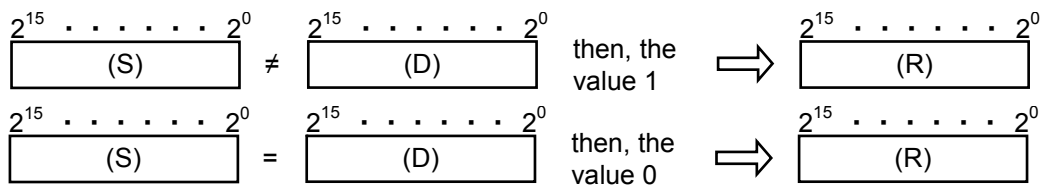
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

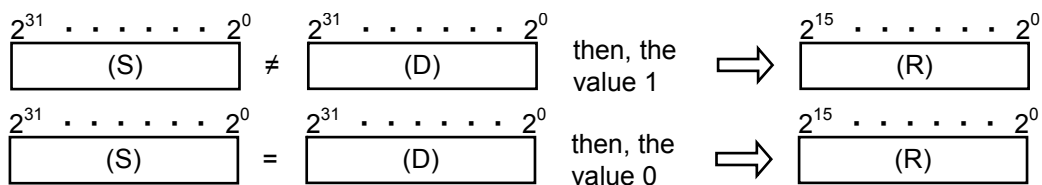
### ● Comparison of word data

The NEQ instruction compares two 16-bit data values specified in Source (S) and Destination (D), respectively. If they are not equal, the instruction then stores the value 1 in Result (R); if they are equal, it stores the value 0 in it.



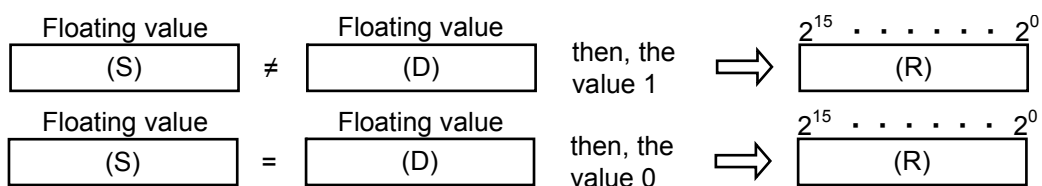
### ● Comparison of long-word data

The NEQ instruction compares two 32-bit data values specified in Source (S) and Destination (D), respectively. If they are not equal, the instruction then stores the value 1 in Result (R); if they are equal, it stores the value 0 in it.



### ● Comparison of floating data

The NEQ instruction compares two floating data values specified in Source (S) and Destination (D), respectively. If they are not equal, the instruction then stores the value 1 in Result (R); if they are equal, it stores the value 0 in it.



(3) Data types

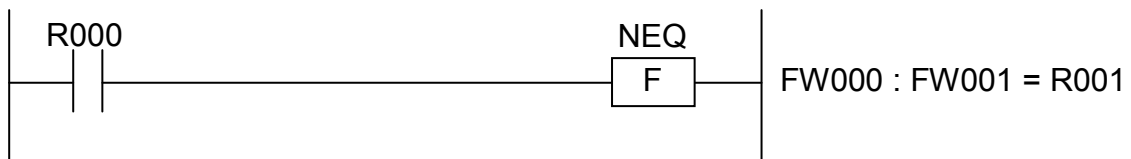
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D	√	√	√	√	√	√	√
R	√	–	–	–	–	–	√

√: May be specified.

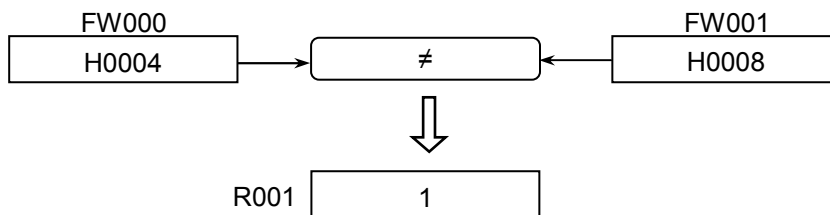
–: May not be specified.

The types of S and D must be the same (i.e., either word, long-word, or floating). If the two are of different types, an input error will result. The type of R must always be word.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the NEQ instruction compares the contents of FW000 and FW001 and stores the result in R001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
–	–	–	–	–	–

All the above flags remain unchanged.

# GT > (GREATER THAN)

## (1) Input format



where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register or a constant.

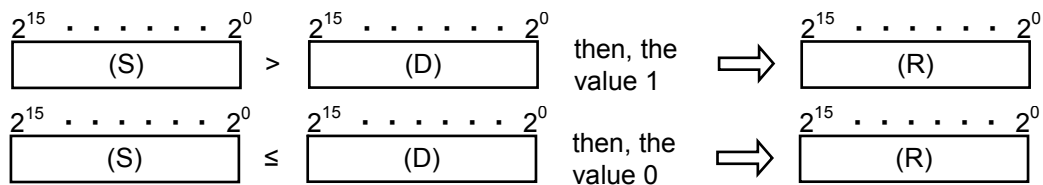
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

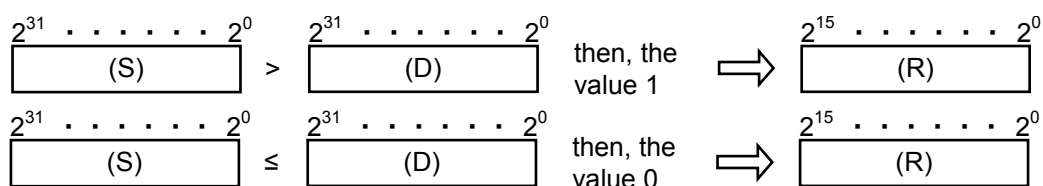
### ● Comparison of word data

The GT instruction compares two 16-bit data values specified in Source (S) and Destination (D), respectively. If Source (S) is greater than Destination (D), the instruction then stores the value 1 in Result (R); otherwise, it stores the value 0 in it.



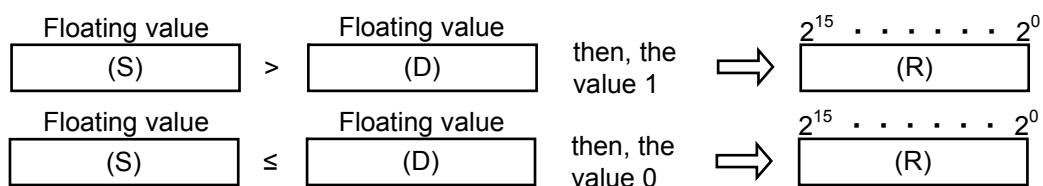
### ● Comparison of long-word data

The GT instruction compares two 32-bit data values specified in Source (S) and Destination (D), respectively. If Source (S) is greater than Destination (D), the instruction then stores the value 1 in Result (R); otherwise, it stores the value 0 in it.



### ● Comparison of floating data

The GT instruction compares two floating data values specified in Source (S) and Destination (D), respectively. If Source (S) is greater than Destination (D), the instruction then stores the value 1 in Result (R); otherwise, it stores the value 0 in it.



(3) Data types

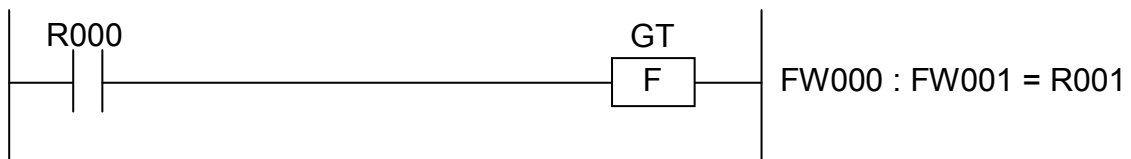
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D	√	√	√	√	√	√	√
R	√	–	–	–	–	–	√

√: May be specified.

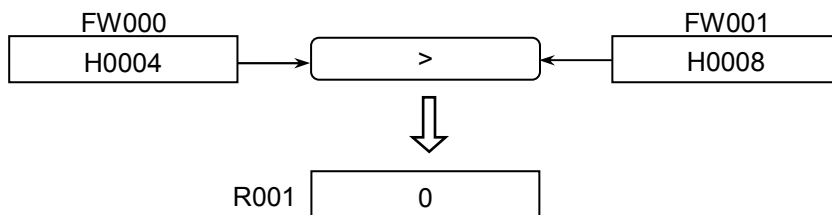
–: May not be specified.

The types of S and D must be the same (i.e., either word, long-word, or floating). If the two are of different types, an input error will result. The type of R must always be word.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the GT instruction compares the contents of FW000 and FW001 and stores the result in R001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
–	–	–	–	–	–

All the above flags remain unchanged.

# LT < (LESS THAN)

## (1) Input format



where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register or a constant.

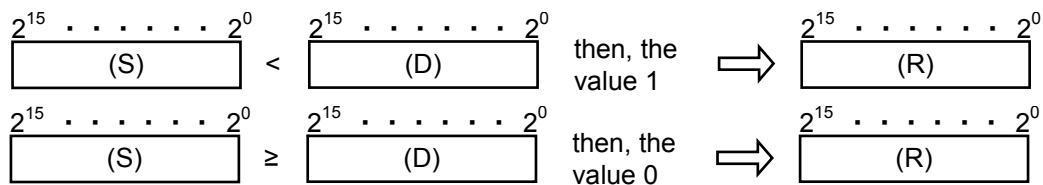
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

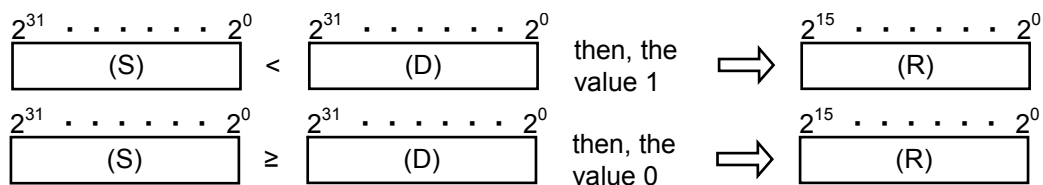
### ● Comparison of word data

The LT instruction compares two 16-bit data values specified in Source (S) and Destination (D), respectively. If Source (S) is less than Destination (D), the instruction then stores the value 1 in Result (R); otherwise, it stores the value 0 in it.



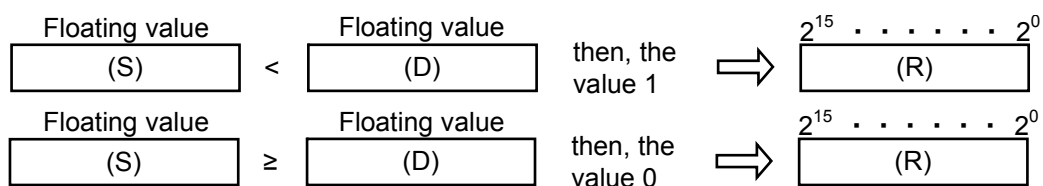
### ● Comparison of long-word data

The LT instruction compares two 32-bit data values specified in Source (S) and Destination (D), respectively. If Source (S) is less than Destination (D), the instruction then stores the value 1 in Result (R); otherwise, it stores the value 0 in it.



### ● Comparison of floating data

The LT instruction compares two floating data values specified in Source (S) and Destination (D), respectively. If Source (S) is less than Destination (D), the instruction then stores the value 1 in Result (R); otherwise, it stores the value 0 in it.



(3) Data types

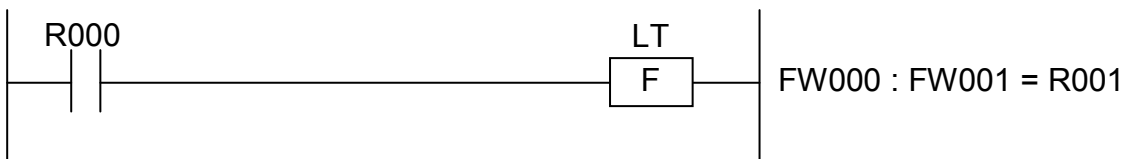
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D	√	√	√	√	√	√	√
R	√	–	–	–	–	–	√

√: May be specified.

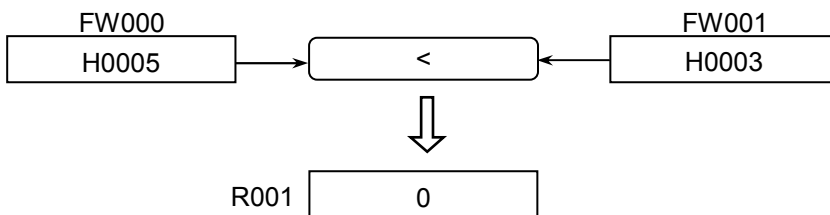
–: May not be specified.

The types of S and D must be the same (i.e., either word, long-word, or floating). If the two are of different types, an input error will result. The type of R must always be word.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the LT instruction compares the contents of FW000 and FW001 and stores the result in R001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
–	–	–	–	–	–

All the above flags remain unchanged.

# GE ≥ (GREATER OR EQUAL)

## (1) Input format



where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register or a constant.

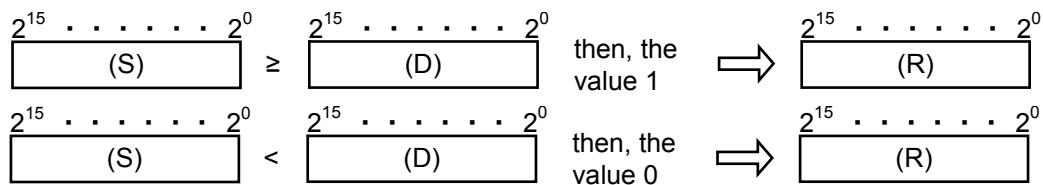
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

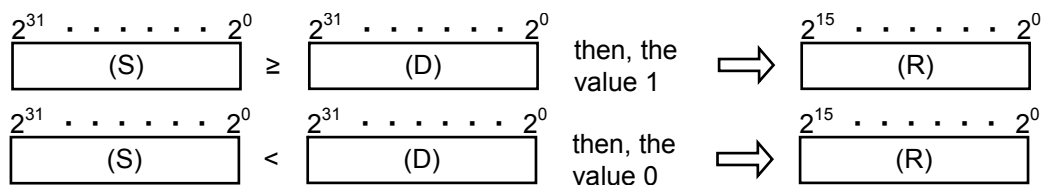
### ● Comparison of word data

The GE instruction compares two 16-bit data values specified in Source (S) and Destination (D), respectively. If Source (S) is greater than or equal to Destination (D), the instruction then stores the value 1 in Result (R); otherwise, it stores the value 0 in it.



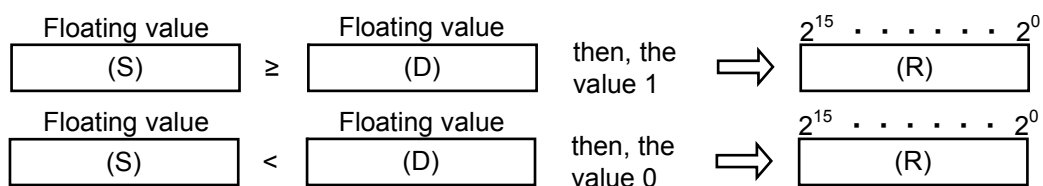
### ● Comparison of long-word data

The GE instruction compares two 32-bit data values specified in Source (S) and Destination (D), respectively. If Source (S) is greater than or equal to Destination (D), the instruction then stores the value 1 in Result (R); otherwise, it stores the value 0 in it.



### ● Comparison of floating data

The GE instruction compares two floating data values specified in Source (S) and Destination (D), respectively. If Source (S) is greater than or equal to Destination (D), the instruction then stores the value 1 in Result (R); otherwise, it stores the value 0 in it.



(3) Data types

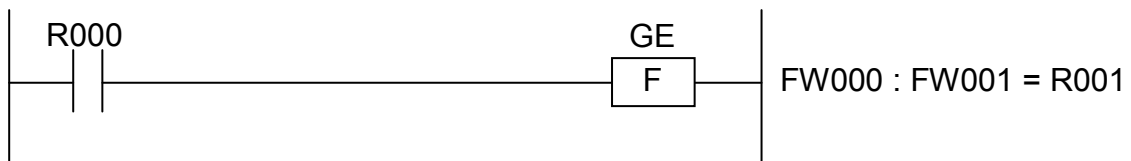
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D	√	√	√	√	√	√	√
R	√	–	–	–	–	–	√

√: May be specified.

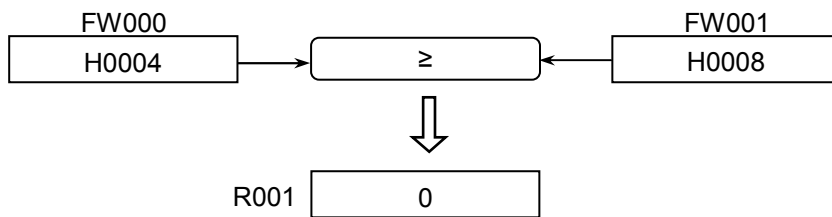
–: May not be specified.

The types of S and D must be the same (i.e., either word, long-word, or floating). If the two are of different types, an input error will result. The type of R must always be word.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the GE instruction compares the contents of FW000 and FW001 and stores the result in R001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
–	–	–	–	–	–

All the above flags remain unchanged.



# LE $\geq$ (LESS OR EQUAL)

## (1) Input format



where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register or a constant.

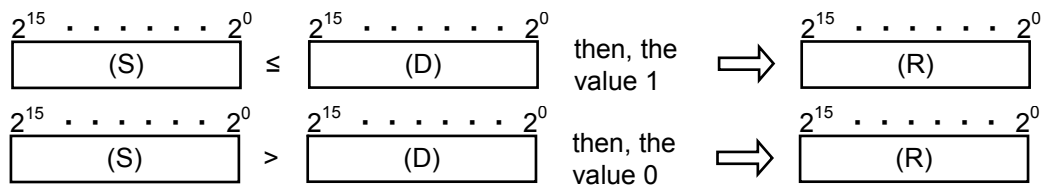
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

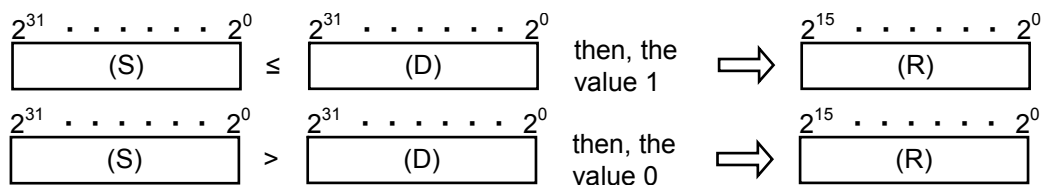
### ● Comparison of word data

The LE instruction compares two 16-bit data values specified in Source (S) and Destination (D), respectively. If Source (S) is less than or equal to Destination (D), the instruction then stores the value 1 in Result (R); otherwise, it stores the value 0 in it.



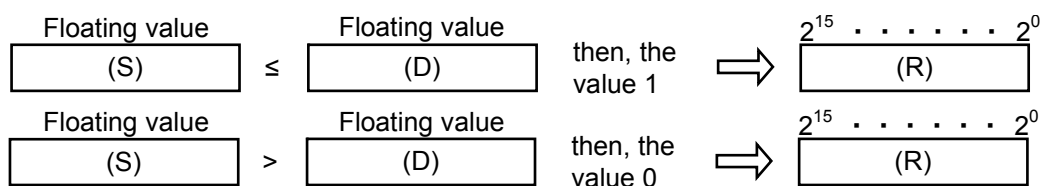
### ● Comparison of long-word data

The LE instruction compares two 32-bit data values specified in Source (S) and Destination (D), respectively. If Source (S) is less than or equal to Destination (D), the instruction then stores the value 1 in Result (R); otherwise, it stores the value 0 in it.



### ● Comparison of floating data

The LE instruction compares two floating data values specified in Source (S) and Destination (D), respectively. If Source (S) is less than or equal to Destination (D), the instruction then stores the value 1 in Result (R); otherwise, it stores the value 0 in it.



(3) Data types

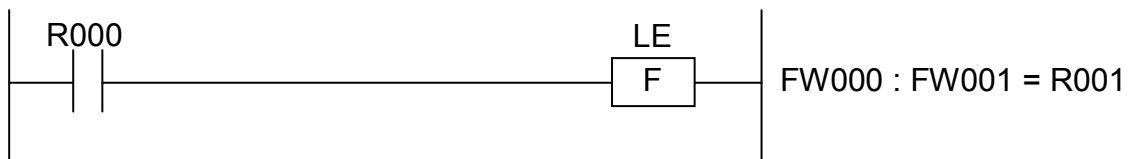
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D	√	√	√	√	√	√	√
R	√	–	–	–	–	–	√

√: May be specified.

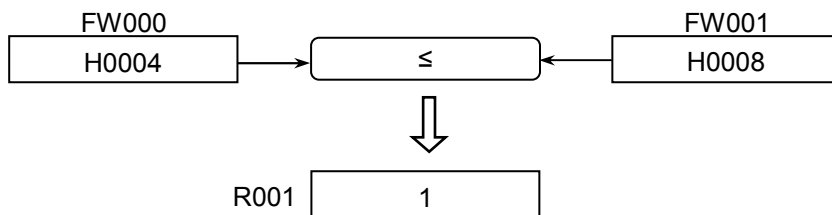
–: May not be specified.

The types of S and D must be the same (i.e., either word, long-word, or floating). If the two are of different types, an input error will result. The type of R must always be word.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the LE instruction compares the contents of FW000 and FW001 and stores the result in R001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
–	–	–	–	–	–

All the above flags remain unchanged.

# TST TEST

(1) Input format

TST S

where:

S: (Source) is a source storage register.

Note: At least one space must be inserted between the function name and parameter.

(2) Function

The TST instruction tests the content of Source (S) for polarity and sets the positive (p), negative (N), or zero (Z) flag, depending on the polarity found. All the other flags remain unchanged.

<Operation result flags>

X	E	P	N	Z	V
-	-	↕	↕	↕	-

● Test of word data

$$\boxed{\begin{matrix} 2^{15} & \dots & 2^0 \\ (S) \end{matrix}} > 0 \quad :P \text{ ON (N, Z OFF)}$$

$$\boxed{\begin{matrix} 2^{15} & \dots & 2^0 \\ (S) \end{matrix}} = 0 \quad :Z \text{ ON (P, N OFF)}$$

$$\boxed{\begin{matrix} 2^{15} & \dots & 2^0 \\ (S) \end{matrix}} < 0 \quad :N \text{ ON (P, Z OFF)}$$

● Test of long-word data

$$\boxed{\begin{matrix} 2^{31} & \dots & 2^0 \\ (S) \end{matrix}} > 0 \quad :P \text{ ON (N, Z OFF)}$$

$$\boxed{\begin{matrix} 2^{31} & \dots & 2^0 \\ (S) \end{matrix}} = 0 \quad :Z \text{ ON (P, N OFF)}$$

$$\boxed{\begin{matrix} 2^{31} & \dots & 2^0 \\ (S) \end{matrix}} < 0 \quad :N \text{ ON (P, Z OFF)}$$

● Test of floating data

$$\boxed{\begin{matrix} \text{Floating value} \\ (S) \end{matrix}} > 0 \quad :P \text{ ON (N, Z OFF)}$$

$$\boxed{\begin{matrix} \text{Floating value} \\ (S) \end{matrix}} = 0 \quad :Z \text{ ON (P, N OFF)}$$

$$\boxed{\begin{matrix} \text{Floating value} \\ (S) \end{matrix}} < 0 \quad :N \text{ ON (P, Z OFF)}$$

## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	–	√	–	√	–	√

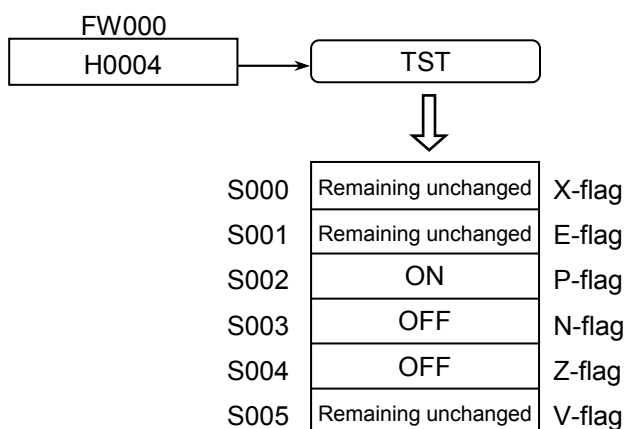
√: May be specified.

–: May not be specified.

## (4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the TST instruction tests the content of FW000 for polarity and sets the appropriate flag.



## (5) Error handling

- If a non-numeric value or infinity is specified in Source (S) for a floating-value test operation, the operation result flags set are as follows:

Source (S)	Operation result flag
Non-numeric value	N ON (P, Z OFF)
+ infinity	P ON (N, Z OFF)
- infinity	N ON (P, Z OFF)

# MOV TRANSFER

---

## (1) Input format

```
MOV S -> D
```

where:

S: (Source) is a source storage register or a constant.

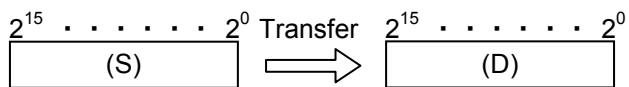
D: (Destination) is a destination storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

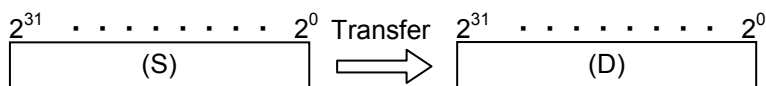
## (2) Function

The MOV (move) instruction transfers a data value specified in Source (S) to Destination (D).

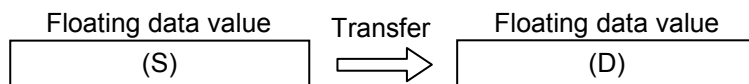
- Transfer of word data



- Transfer of long-word data



- Transfer of floating data



(3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D	√	-	√	-	√	-	√

√: May be specified.

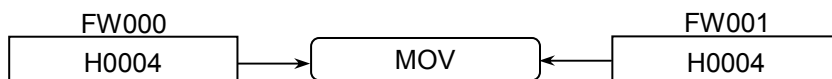
-: May not be specified.

The types of S and D must be the same (i.e., either word, long-word, or floating). If the two are of different types, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the MOV instruction transfers the content of FW000 to FW001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# MOM BATCH TRANSFER

## (1) Input format

MOM S : n -> D

where:

S: (Source) is a source storage register.

n: A count (constant) of the number of words or long words to be transferred.

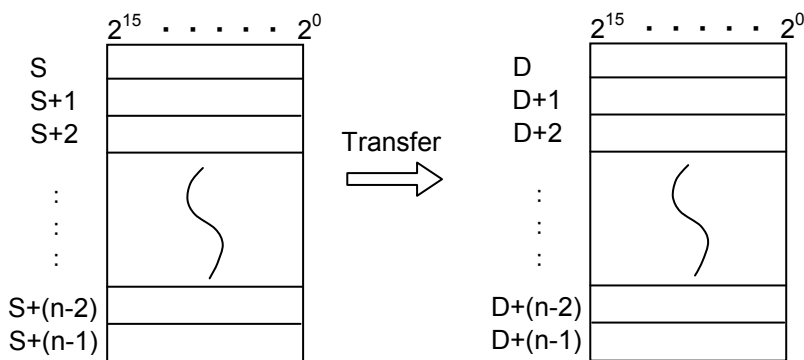
D: (Destination) is a destination storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

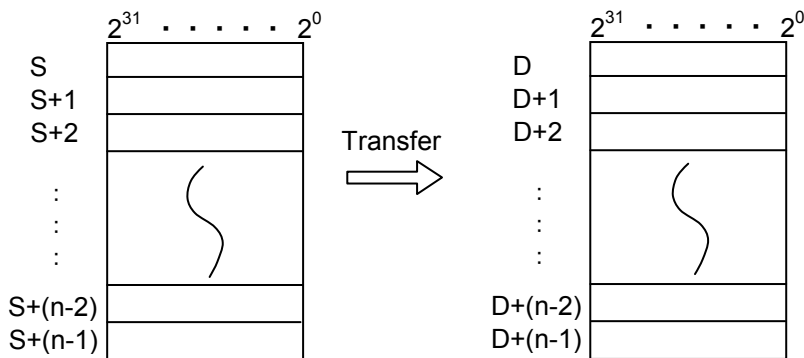
## (2) Function

The MOM (move multi) instruction transfers the contents of the first n steps in Source (S) to the corresponding steps in Destination (D), where n is an integer in the range 1 to 256. (If any integer outside that range is given, the instruction performs nothing.)

### ● Batch transfer of word data



### ● Batch transfer of long-word data



(3) Data types

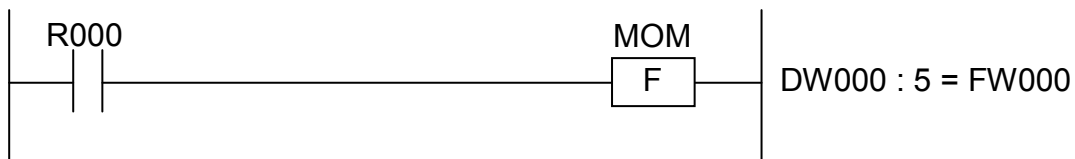
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	–	√	–	–	–	√
n	–	√	–	–	–	–	–
D	√	–	√	–	–	–	√

√: May be specified.

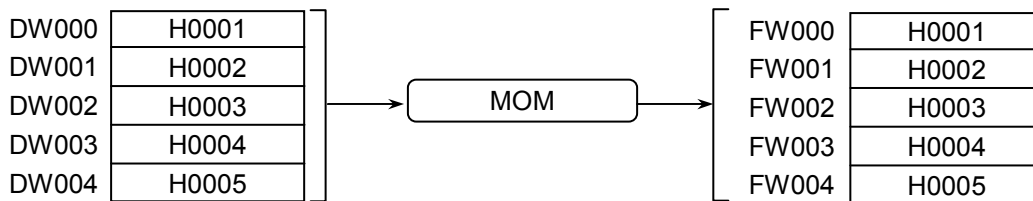
–: May not be specified.

The types of S and D must be the same (i.e., either word or long-word). If the two are of different types, an input error will result. The type of n must always be word.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the MOM instruction transfers the contents of the first five steps in DW000 to the corresponding steps in FW000.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
–	–	–	–	–	–

All the above flags remain unchanged.



# INI BATCH TRANSFER OF SAME DATA

## (1) Input format

INI S : n -> D

where:

S: (Source) is a source storage register or a constant.

n: A count (constant) of the number of words or long words to be transferred.

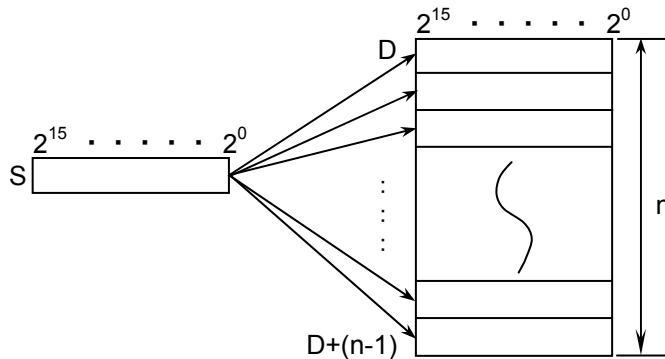
D: (Destination) is a destination storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

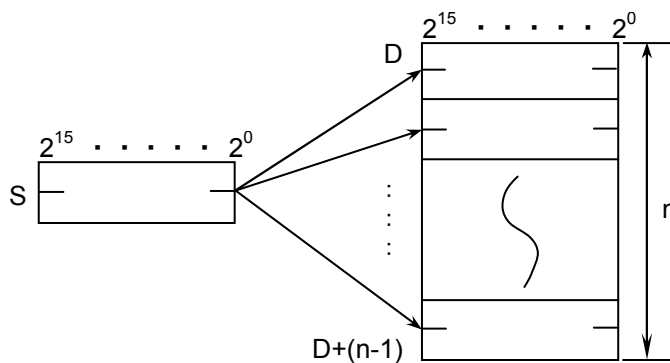
## (2) Function

The INI (initial) instruction transfers the content of Source (S) repeatedly to the first n steps in Destination (D), where n is an integer in the range 1 to 256. (If any integer outside that range is given, the instruction performs nothing.)

- Batch transfer of same word data



- Batch transfer of same long-word data



(3) Data types

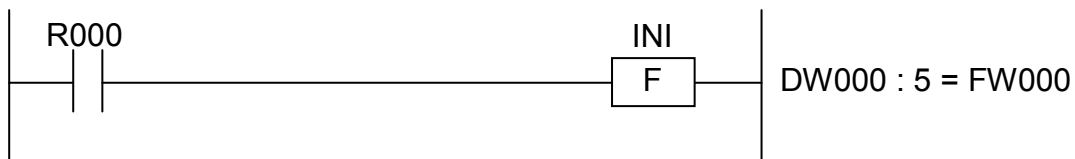
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	-	-	√
n	-	√	-	-	-	-	-
D	√	-	√	-	-	-	√

√: May be specified.

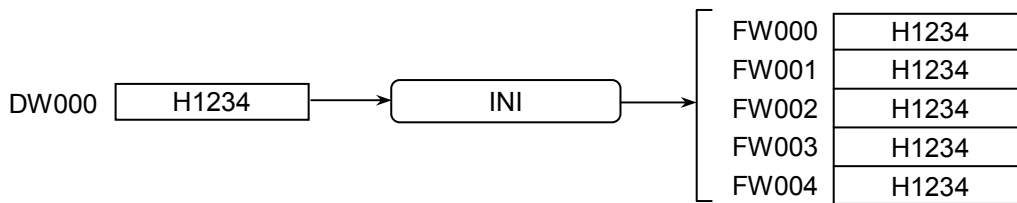
-: May not be specified.

The types of S and D must be the same (i.e., either word or long-word). If the two are of different types, an input error will result. The type of n must always be word.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the INI instruction transfers the content of DW000 repeatedly to the first five steps in FW000.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# EXC EXCHANGE

## (1) Input format



where:

S: (Source) is a source storage register.

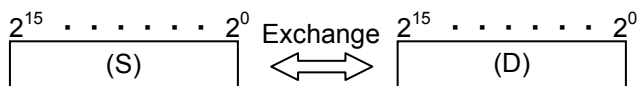
D: (Destination) is a destination storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “:” may be omitted.

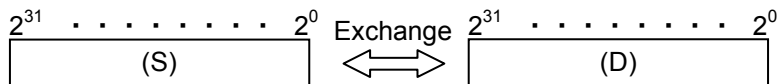
## (2) Function

The EXC instruction exchanges the contents of Source (S) and Destination (D).

- Exchange of word data



- Exchange of long-word data



## (3) Data types

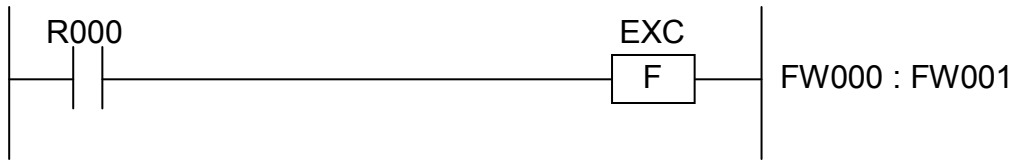
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	–	√	–	–	–	√
D	√	–	√	–	–	–	√

√: May be specified.

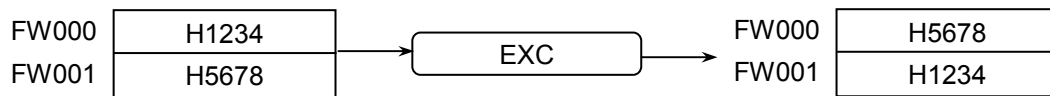
–: May not be specified.

The types of S and D must be the same (i.e., either word or long-word). If the two are of different types, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the EXC instruction exchanges the contents of FW000 and FW001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# PSH WRITE ON FIFO BASIS

## (1) Input format

```
PSH S -> TB
```

where:

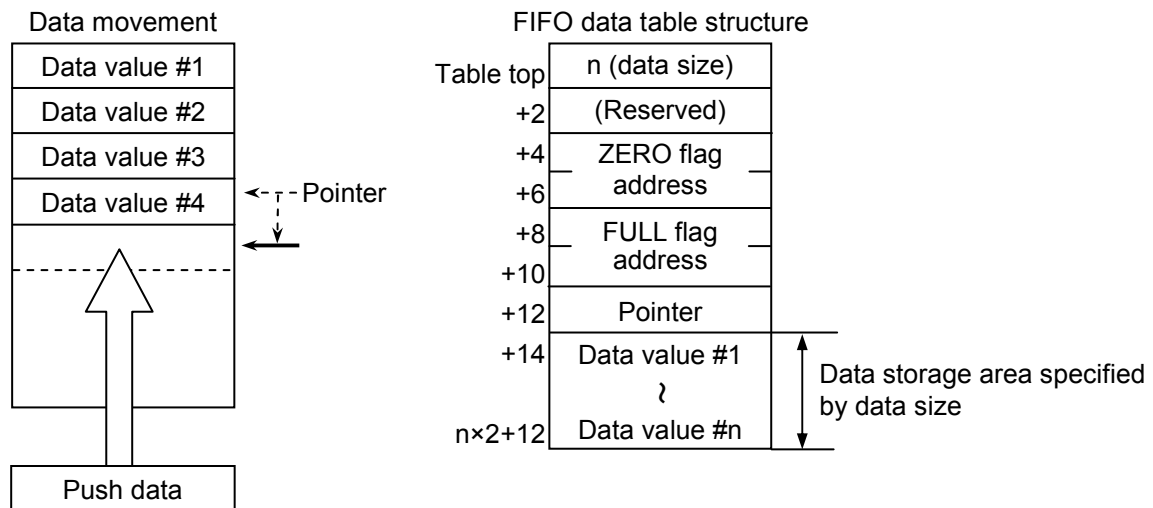
S: (Source) is a source storage register.

TB: Is the starting register of an FIFO table.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

The PSH (FIFO push) instruction writes the content of Source (S) to a specified FIFO table.



Notes:

- If the pointer has a value of n before pushing, this instruction sets the FULL flag and does not perform pushing (the ZERO flag is reset). The instruction also sets the FULL flag if the pointer incremented after pushing reaches n. In any other case, the FULL flag is reset.
- This instruction resets the ZERO flag at the end of its operation, except when it performs nothing in the cases described below.
- If data size n is smaller than or equal to 0 or greater than 256, this instruction performs nothing.
- If the pointer has a value smaller than 0 or greater than n, this instruction performs nothing.

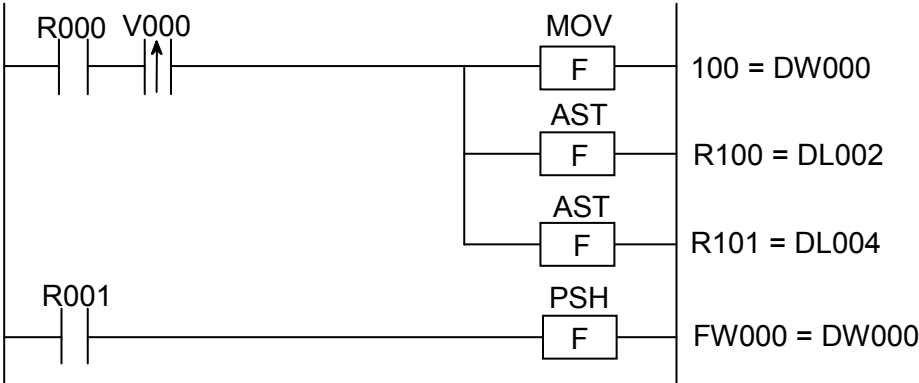
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	–	–	–	–	–	√
TB	√	–	–	–	–	–	√

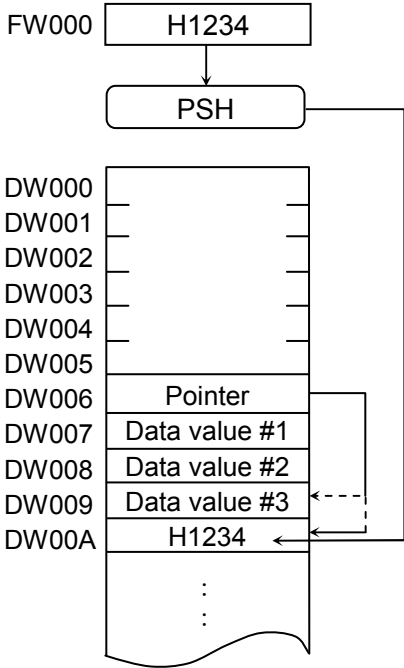
√: May be specified.

–: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) makes a transition from OFF to ON state, the MOV (transfer) and the AST (address setting) instructions set a data size (100) and the addresses of a ZERO flag (R100) and a FULL flag (R101), respectively, only once. Then, if the contact R001 (input condition) is closed (ON), the PSH instruction writes the content of FW000 to a specified FIFO table beginning with DW000. (Data size n is defined by MOV with an immediate data value.)



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# POP READ ON FIFO BASIS

## (1) Input format

```
POP TB -> D
```

where:

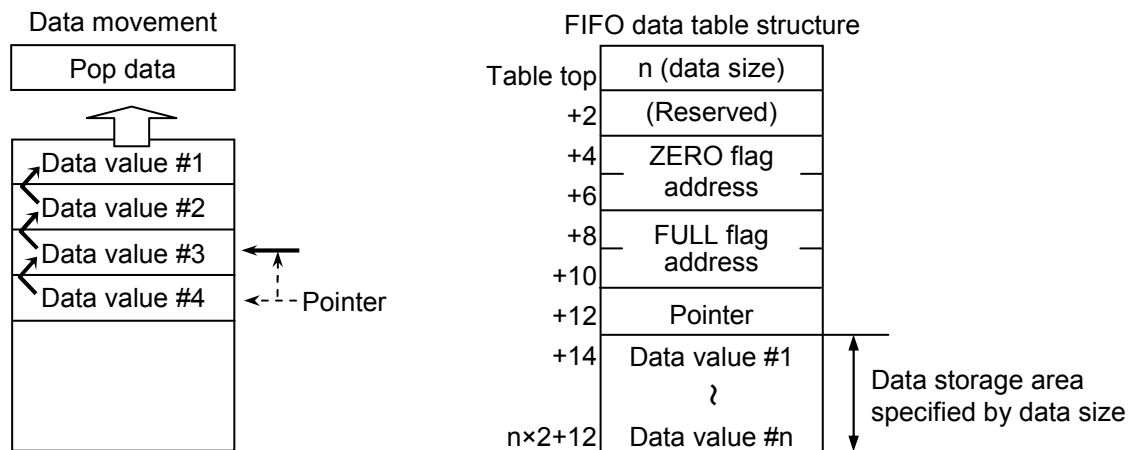
TB: Is the starting address of an FIFO table (register).

D: (Destination) is a destination storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

The POP (FIFO pop) instruction reads a data value from a specified FIFO table and stores the value in Destination (D).



Notes:

- If the pointer has a value of 0 before popping, this instruction sets the ZERO flag and does not perform popping (the FULL flag is reset). The instruction also sets the ZERO flag if the pointer decremented after popping reaches 0. In any other case, the ZERO flag is reset.
- This instruction resets the FULL flag at the end of its operation, except when it performs nothing in the cases described below.
- If data size n is smaller than or equal to 0 or greater than 256, this instruction performs nothing.
- If the pointer has a value smaller than 0 or greater than n, this instruction performs nothing.

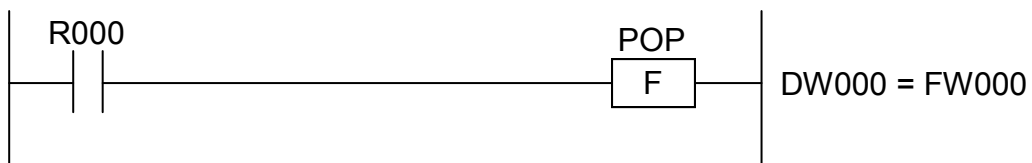
(3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
D	√	-	√	-	-	-	√
TB	√	-	√	-	-	-	√

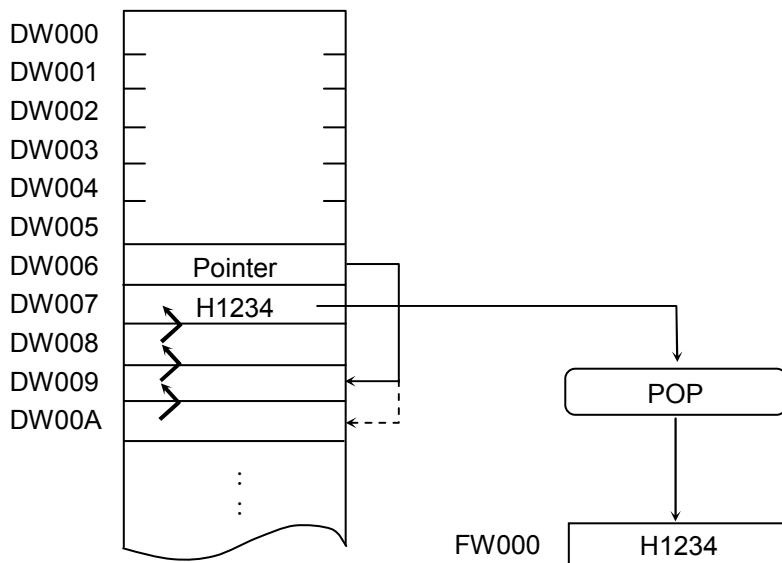
√: May be specified.

-: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the POP instruction reads a data value from a specified FIFO table beginning with DW000 and stores the value in FW000.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.



# PSHO WRITE ON FIFO BASIS

## (1) Input format

```
PSHO S -> TB
```

where:

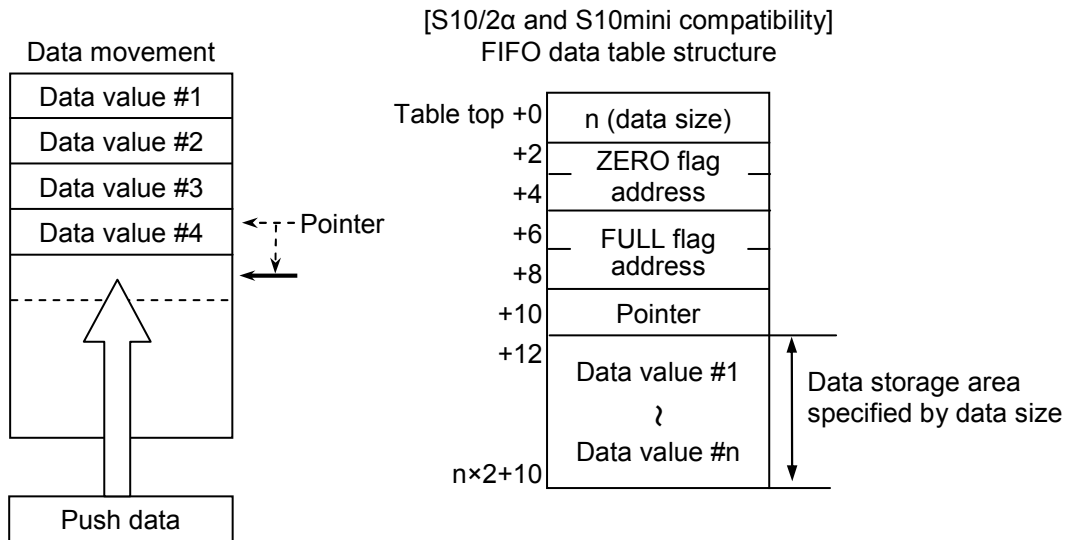
S: (Source) is a source storage register.

TB: Is the starting register of an FIFO table.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

This instruction pushes the contents of the source (S) to the S10/2α and S10mini-compatible FIFO table.



Notes:

- If the pointer has a value of n before pushing, this instruction sets the FULL flag and does not perform pushing (the ZERO flag is reset). The instruction also sets the FULL flag if the pointer incremented after pushing reaches n. In any other case, the FULL flag is reset.
- This instruction resets the ZERO flag at the end of its operation, except when it performs nothing in the cases described below.
- If data size n is smaller than or equal to 0 or greater than 256, this instruction performs nothing.
- If the pointer has a value smaller than 0 or greater than n, this instruction performs nothing.

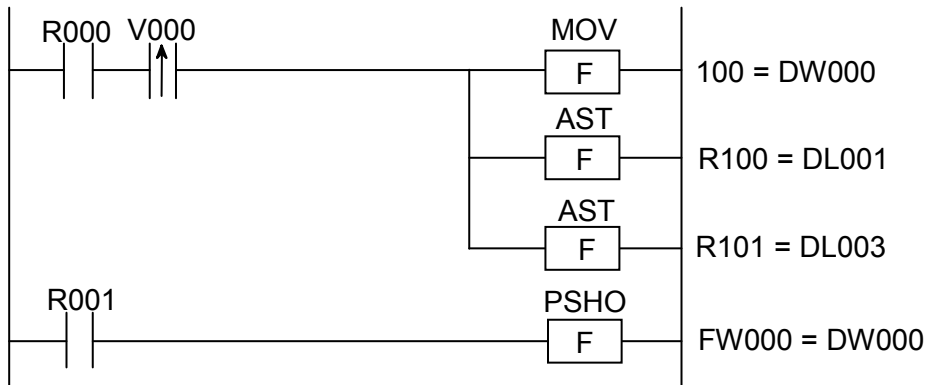
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	–	–	–	–	–	√
TB	√	–	–	–	–	–	√

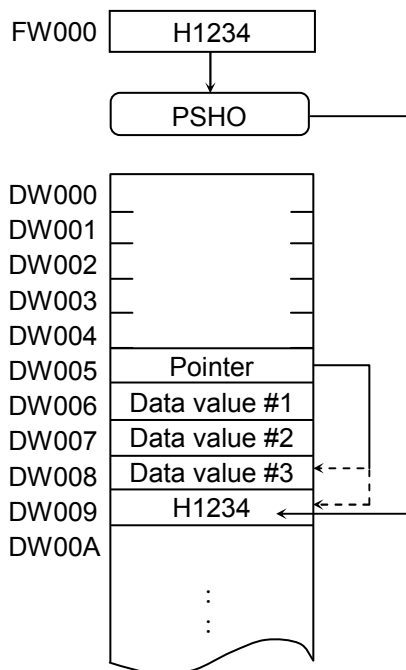
√: May be specified.

–: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) makes a transition from OFF to ON state, the MOV (transfer) and the AST (address setting) instructions set a data size (100) and the addresses of a ZERO flag (R100) and a FULL flag (R101), respectively, only once. Then, if the contact R001 (input condition) is closed (ON), the PSH instruction writes the content of FW000 to a specified FIFO table beginning with DW000. (Data size n is defined by MOV with an immediate data value.)



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# POPO READ ON FIFO BASIS

## (1) Input format

POPO TB -> D

where:

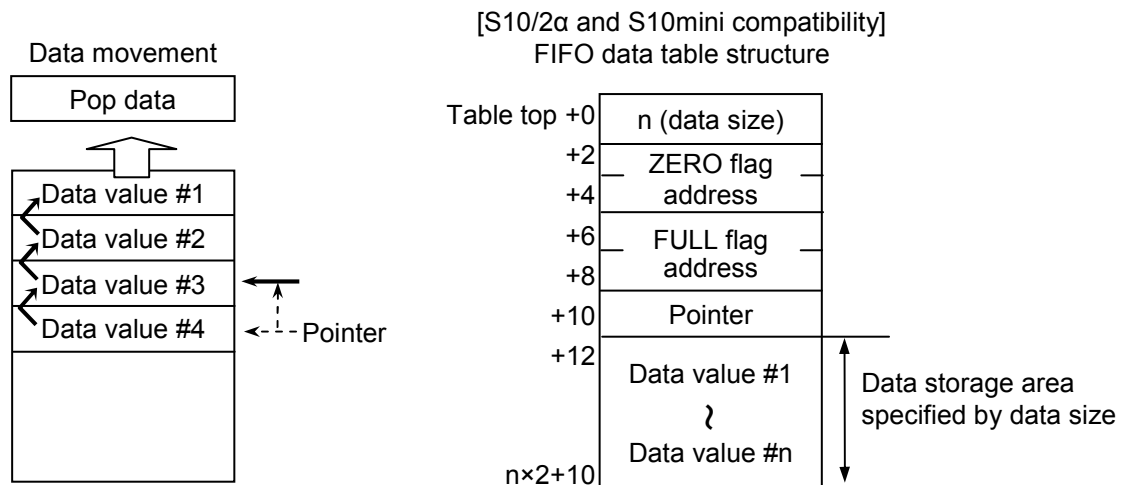
TB: Is the starting address of an FIFO table (register).

D: (Destination) is a destination storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

This instruction performs popping for the S10/2α and S10mini-compatible FIFO table and stores pop data in the destination (D).



Notes:

- If the pointer has a value of 0 before popping, this instruction sets the ZERO flag and does not perform popping (the FULL flag is reset). The instruction also sets the ZERO flag if the pointer decremented after popping reaches 0. In any other case, the ZERO flag is reset.
- This instruction resets the FULL flag at the end of its operation, except when it performs nothing in the cases described below.
- If data size n is smaller than or equal to 0 or greater than 256, this instruction performs nothing.
- If the pointer has a value smaller than 0 or greater than n, this instruction performs nothing.

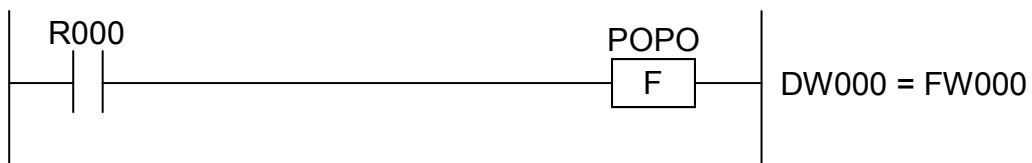
(3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
D	√	-	√	-	-	-	√
TB	√	-	√	-	-	-	√

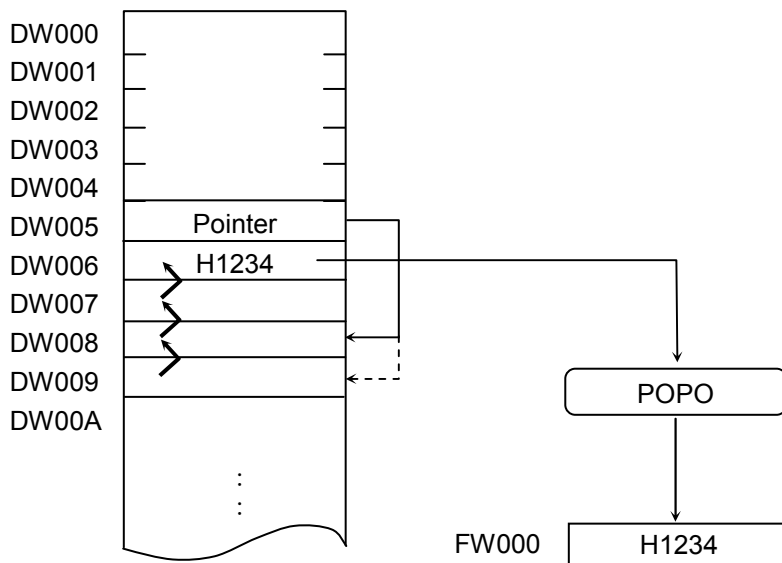
√: May be specified.

-: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the POP instruction reads a data value from a specified FIFO table beginning with DW000 and stores the value in FW000.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# AST ADDRESS SETTING

## (1) Input format

AST S -> D

where:

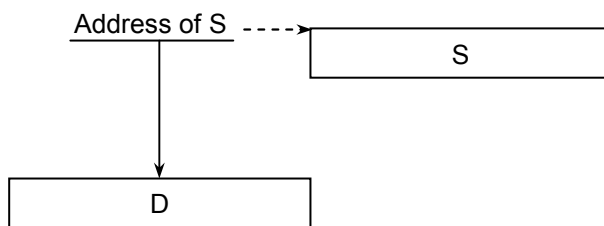
S: (Source) is a source storage register.

D: (Destination) is a destination storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

The AST instruction stores the address of Source (S) in Destination (D).



## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	–	–	–	–	–	√
D	–	–	√	–	–	–	√

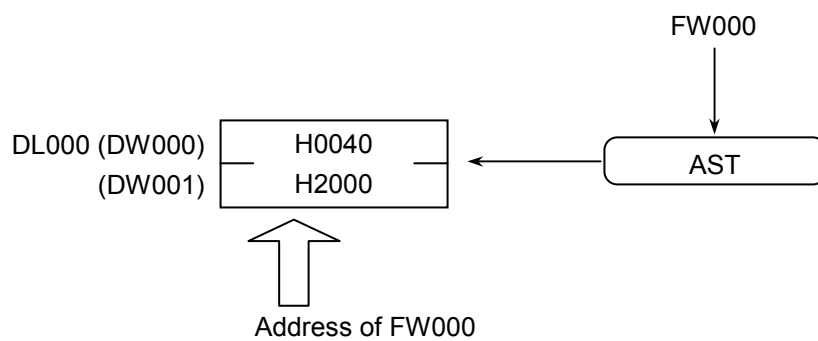
√: May be specified.

–: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) makes a transition from OFF to ON state, the AST instruction stores the address of FW000 in DL000.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# SCH SEARCH

## (1) Input format

```
SCH S : D : m -> R
```

where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register.

m: Is the number of steps to be searched.

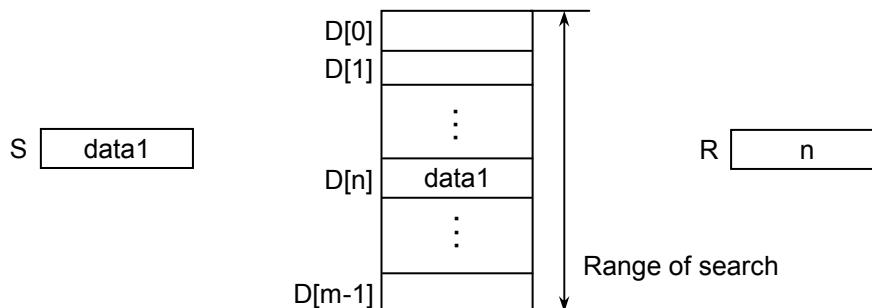
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

### ● Search for word data

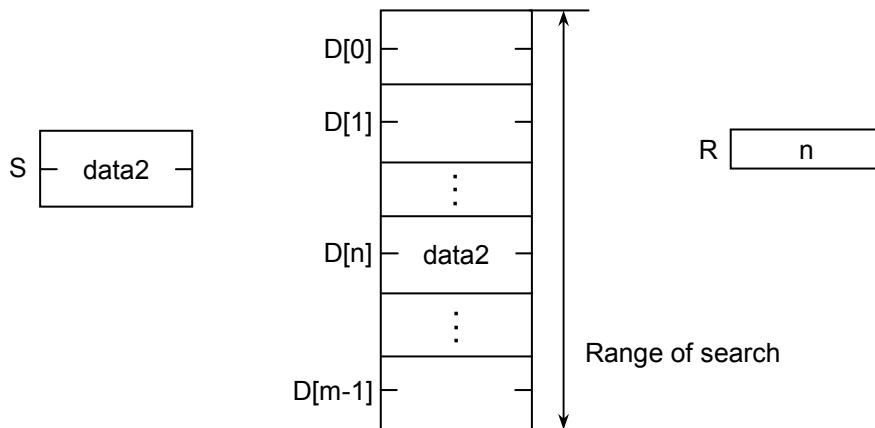
The SCH instruction searches the first m steps in a specified destination (D) for a word data value specified in Source (S) and, if it is found, stores in Result (R) the step number of the step containing the matching value.



- The matching value is the first value that is found in a specified range of search, starting from the beginning of that range.
- If there is no matching value in a specified range of search, this instruction stores the value -1 (HFFFF) in Result (R).
- If a value specified as m (the number of steps to be searched) is not within the range 1 to 256, this instruction performs nothing.

- Search for long-word data

The SCH instruction searches the first  $m$  steps in a specified destination (D) for a long-word data value specified in Source (S) and, if it is found, stores in Result (R) the step number of the step containing the matching value.

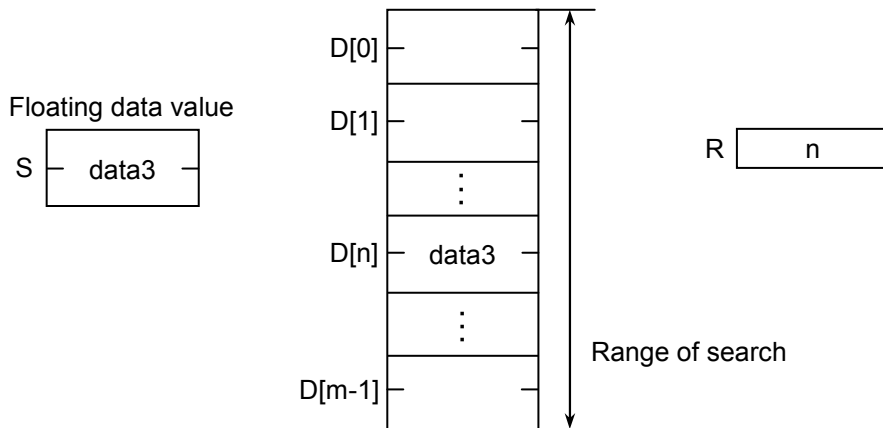


- The matching value is the first value that is found in a specified range of search, starting from the beginning of that range.
- If there is no matching value in a specified range of search, this instruction stores the value -1 (HFFFF) in Result (R).
- If a value specified as  $m$  (the number of steps to be searched) is not within the range 1 to 256, this instruction performs nothing.



- Search for floating data

The SCH instruction searches the first m steps in a specified destination (D) for a floating data value specified in Source (S) and, if it is found, stores in Result (R) the step number of the step containing the matching value.



- The matching value is the first value that is found in a specified range of search, starting from the beginning of that range.
- If there is no matching value in a specified range of search, this instruction stores the value -1 (HFFFF) in Result (R).
- If a value specified as m (the number of steps to be searched) is not within the range 1 to 256, this instruction performs nothing.

Note: Care must be taken when using this instruction for searching for floating data values. Any data value in storage, which is actually equal to a given data value, may be skipped as not matching, due to error contained in those values.

(3) Data types

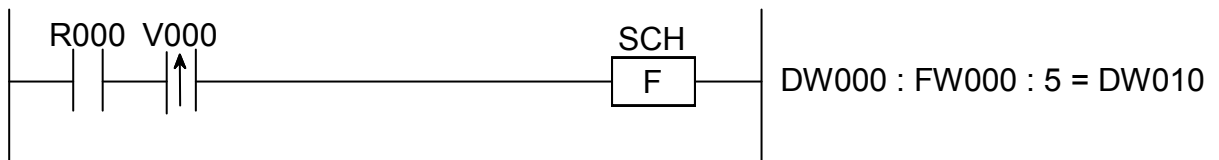
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D	√	–	√	–	√	–	√
m	–	√	–	–	–	–	–
R	√	–	–	–	–	–	√

√: May be specified.

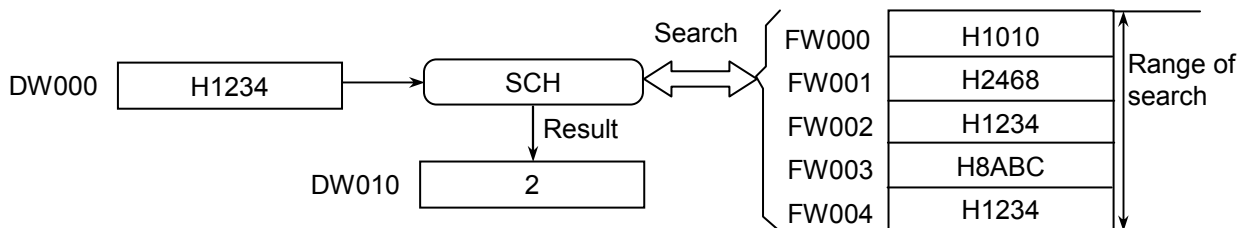
–: May not be specified.

The types of S and D must be the same (i.e., either word, long-word, or floating). If the two are of different types, an input error will result. The types of m and R must always be word.

(4) Example program



In this example, if the contact R000 (input condition) makes a transition from OFF to ON state, the SCH instruction searches the first five steps FW000 through FW004 for the same data value as the content of DW000 only once and stores the result in DW010.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
–	–	–	–	–	–

All the above flags remain unchanged.

# BTF BINARY-TO-FLOATING CONVERSION

## (1) Input format

BTF S -> R

where:

S: (Source) is a binary-data storage register or a binary constant.

R: (Result) is an operation result (floating data) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

### ● Conversion of word data

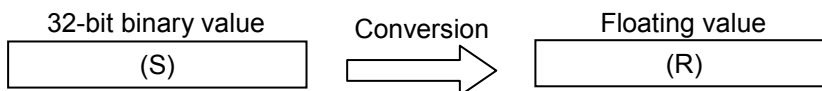
The BTF instruction converts a 16-bit binary data value specified in Source (S) to floating format and stores the result in Result (R).



The values that may be specified in Source (S) are in the range -32768 to 32767.

### ● Conversion of long-word data

The BTF instruction converts a 32-bit binary data value specified in Source (S) to floating format and stores the result in Result (R).



The values that may be specified in Source (S) are in the range -2147483648 to 2147483647.

## (3) Data types

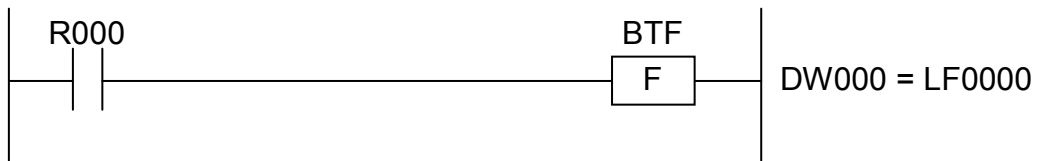
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	–	–	√
R	–	–	–	–	√	–	√

√: May be specified.

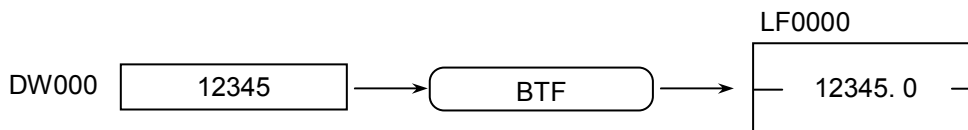
–: May not be specified.

The type of R must always be floating.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the BTF instruction converts the content of DW000 to floating data format and stores the result in LF0000.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

Note: Any floating data value, processed in 32-bit single-precision form, has a total of 24 significant bits when expressed in binary, and about seven significant digits when expressed in decimal. Therefore, if an integer outside the range -16777216 to 16777215 (24-bit binary values) is converted by using this instruction, the resulting value will contain error, because it is rounded off at its 25th significant bit position.

# FTB FLOATING-TO-BINARY CONVERSION

## (1) Input format

FTB S -> R

where:

S: (Source) is a floating-data storage register or a floating constant.

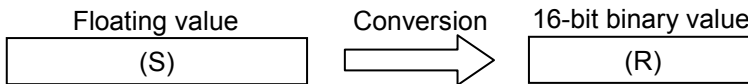
R: (Result) is an operation result (binary data) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

### ● Conversion to binary word data format

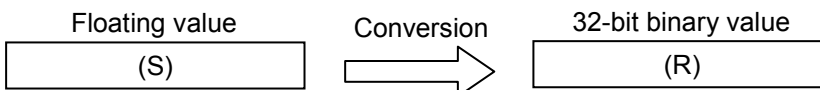
The FTB instruction converts a floating data value specified in Source (S) to 16-bit binary data format and stores the result in Result (R).



- The values that may be specified in Source (S) are in the range -32768 to 32767.
- The resulting value is one that is rounded off at the first decimal place in the floating data value.

### ● Conversion to binary long-word data format

The FTB instruction converts a floating data value specified in Source (S) to 32-bit binary data format and stores the result in Result (R).



- The values that may be specified in Source (S) are in the range -2147483648 to 2147483647.
- The resulting value is one that is rounded off at the first decimal place in the floating data value.

## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	-	-	-	-	√	√	√
R	√	-	√	-	-	-	√

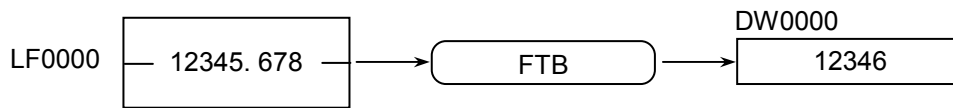
√: May be specified.

-: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the FTB instruction converts the content of LF0000 to binary data format and stores the result in DW0000.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	↕

where:

V: When the type of given data is word:

- Set to 0 if Result (R) is in the range -32768 to 32767; otherwise, set to 1.

When it is long-word:

- Set to 0 if Result (R) is in the range -2147483648 to 2147483647; otherwise, set to 1.

All the other flags then V remain unchanged.

- If an overflow occurs in the operation, one of the following full-scale values will be stored in Result (R):

	In case of a positive overflow:	In case of a negative overflow:
Word	32767	-32768
Long-word	2147483647	-2147483648

# BTD BINARY-TO-BCD CONVERSION

## (1) Input format

BTD S -> R

where:

S: (Source) is a binary-data storage register or a binary constant.

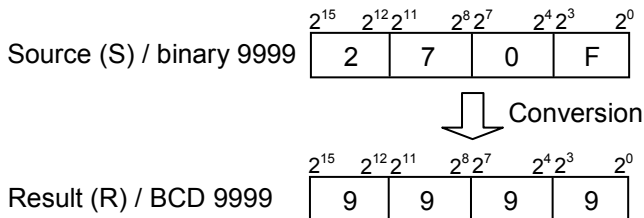
R: (Result) is an operation result (BCD data) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

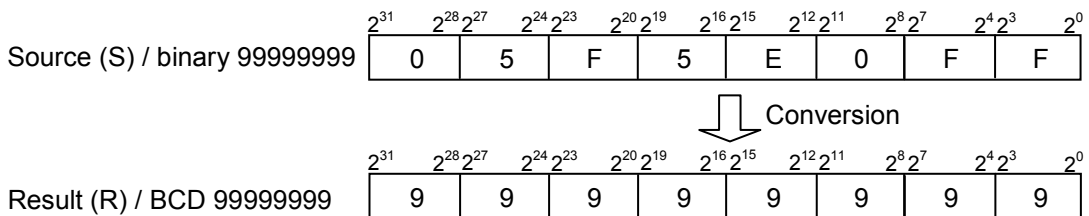
### ● Conversion of word data

The BTD instruction converts a binary data value (0 to 9999) specified in Source (S) to BCD (Binary Coded Decimal) data format and stores the result in Result (R).



### ● Conversion of long-word data

The BTD instruction converts a binary data value (0 to 99999999) specified in Source (S) to BCD data format and stores the result in Result (R).



## (3) Data types

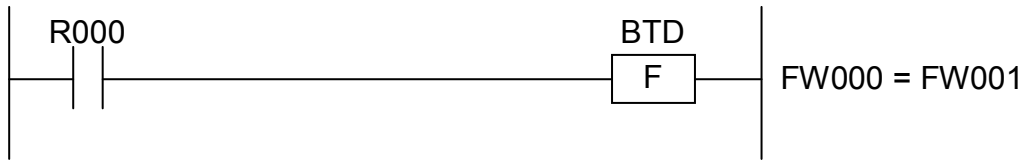
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	–	–	√
R	√	–	√	–	–	–	√

√: May be specified.

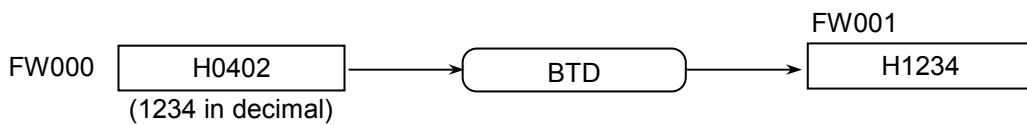
–: May not be specified.

The types of S and R must be the same (i.e., either word or long-word). If the two are of different types, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the BTD instruction converts the binary data content of FW000 to BCD data format and stores the result in FW001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	↕

where:

E: Set to 1 if Source (S) is smaller than 0; otherwise, set to 0.

V: When the type of given data is word:

- Set to 1 if Source (S) is greater than 9999; otherwise, set to 0.

When it is long-word:

- Set to 1 if Source (S) is greater than 99999999; otherwise, set to 0.

All the other flags then V and E remain unchanged.

- If Source (S) is smaller than 0, this instruction sets the E-flag of the operation result flags (the V-flag is reset) and performs nothing. The value of Result (R) remains unchanged.
- If an overflow occurs in the operation (the V-flag is set), one of the following full-scale values will be stored in Result (R):

Word	Long-word
H9999	H99999999



# DTB BCD-TO-BINARY CONVERSION

## (1) Input format

DTB S -> R

where:

S: (Source) is a BCD data storage register or a BCD constant.

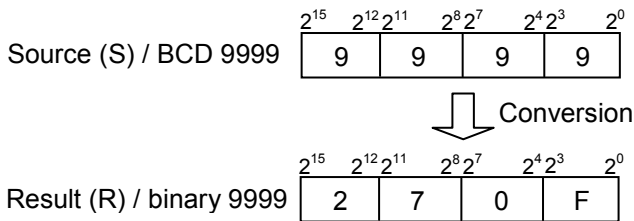
R: (Result) is an operation result (binary data) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

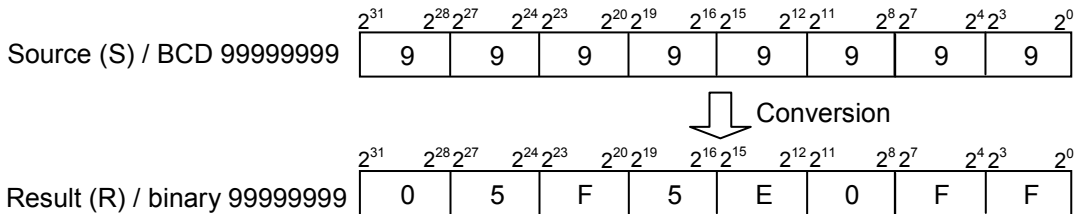
### ● Conversion of word BCD data

The DTB instruction converts a BCD data value (0 to 9999) specified in Source (S) to binary data format and stores the result in Result (R).



### ● Conversion of long-word BCD data

The DTB instruction converts a long-word BCD data value (0 to 99999999) specified in Source (S) to binary data format and stores the result in Result (R).



## (3) Data types

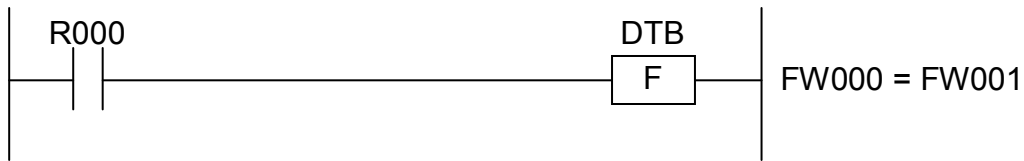
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	–	–	√
R	√	–	√	–	–	–	√

√: May be specified.

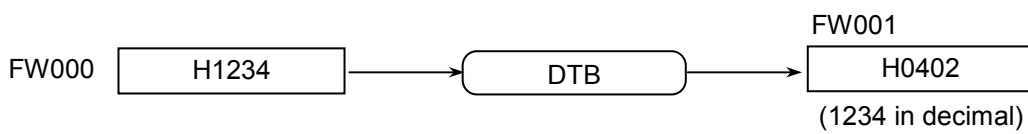
–: May not be specified.

The types of S and R must be the same (i.e., either word or long-word). If the two are of different types, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the DTB instruction converts the BCD data content of FW000 to binary data format and stores the result in FW001.



(5) Error handling

- Operation result flags

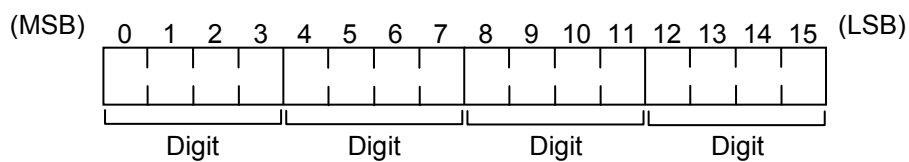
X	E	P	N	Z	V
-	↕	-	-	-	-

where:

E: Set to 0 if each digit (4-bit) (\*) specified in Source (S) is in the range 0 to 9; otherwise, set to 1.

All the other flags then E remain unchanged.

(\*) The digit is as shown below.



# SEG BINARY-TO-SEGMENT CONVERSION

---

## (1) Input format



where:

S: (Source) is a binary data storage register or a binary constant.

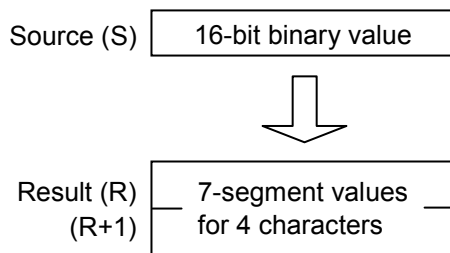
R: (Result) is an operation result (7-segment data) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

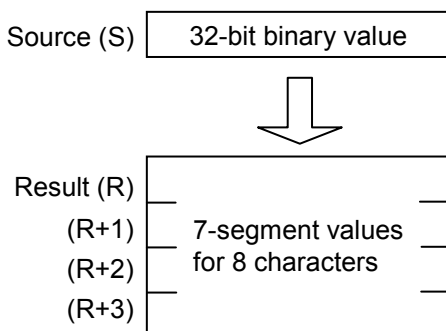
### ● Conversion of word data

The SEG instruction converts a 16-bit binary data value specified in Source (S) to 7-segment data format and stores the result in Result (R).

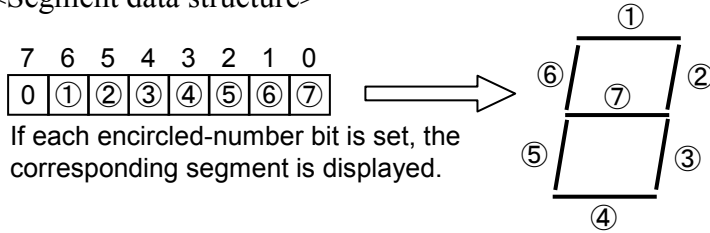


### ● Conversion of long-word data

The SEG instruction converts a 32-bit binary data value specified in Source (S) to 7-segment data format and stores the result in Result (R).



<Segment data structure>



If each encircled-number bit is set, the corresponding segment is displayed.

Correspondence between displays and segment data:

No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Display																
Data value	H7E	H30	H6D	H79	H33	H5B	H5F	H70	H7F	H7B	H77	H1F	H4E	H3D	H4F	H47

(3) Data types

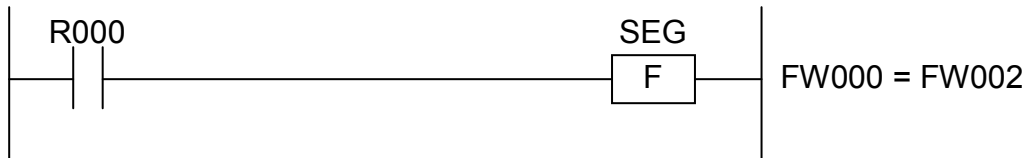
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	-	-	√
R	√	-	√	-	-	-	√

√: May be specified.

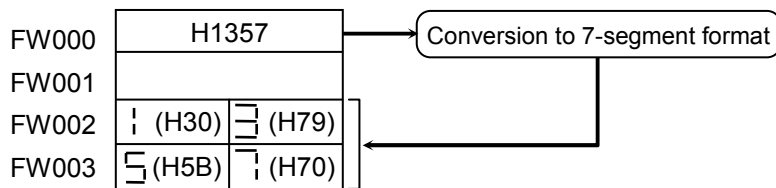
-: May not be specified.

The types of S and R must be the same (i.e., either word or long-word). If the two are of different types, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the SEG instruction converts the binary data content of FW000 to 7-segment data format (four characters) and stores the result in FW002.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# ASP BINARY-TO-ASCII CONVERSION IN PACK MODE

## (1) Input format

ASP S -> R

where:

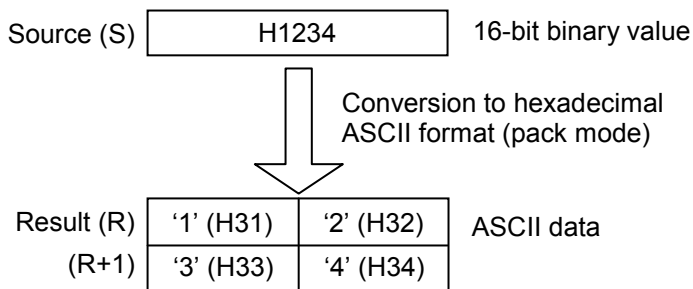
S: (Source) is a binary data storage register or a binary constant.

R: (Result) is an operation result (ASCII data) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

The ASP instruction converts a 16-bit binary data value specified in Source (S) to hexadecimal ASCII data format in pack mode and stores the result in Result (R).



<Correspondence between binary and ASCII data>

Binary	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ASCII	H30	H31	H32	H33	H34	H35	H36	H37	H38	H39	H41	H42	H43	H44	H45	H46

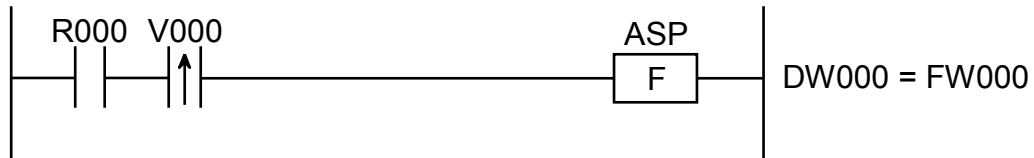
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	–	–	–	–	√
R	√	–	–	–	–	–	√

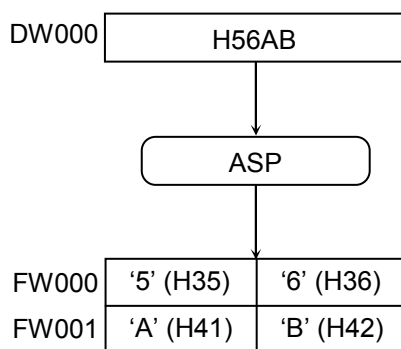
√: May be specified.

–: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) makes a transition from OFF to ON state, the ASP instruction converts the binary data content of DW000 to hexadecimal ASCII data format in pack mode only once and stores the result in FW000.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# ASU BINARY-TO-ASCII CONVERSION IN UNPACK MODE

## (1) Input format



where:

S: (Source) is a binary data storage register or a binary constant.

R: (Result) is an operation result (ASCII data) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

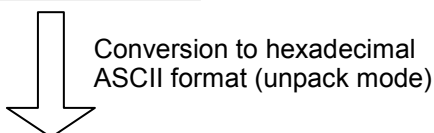
## (2) Function

The ASU instruction converts a 16-bit binary data value specified in Source (S) to hexadecimal ASCII data format in unpack mode and stores the result in Result (R).

Source (S) 

H1234
-------

 16-bit binary value



Result (R)	'0' (H30)	'1' (H31)	ASCII data
(R+1)	'0' (H30)	'2' (H32)	
(R+2)	'0' (H30)	'3' (H33)	
(R+3)	'0' (H30)	'4' (H34)	

The result is stored byte-by-byte in the lower bytes at (R), (R+1), (R+2), and (R+3), starting from the high-order digit. The upper bytes at (R) through (R+3) are set to an ASCII value of 0 (H30).

<Correspondence between binary and ASCII data>

Binary	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ASCII	H30	H31	H32	H33	H34	H35	H36	H37	H38	H39	H41	H42	H43	H44	H45	H46

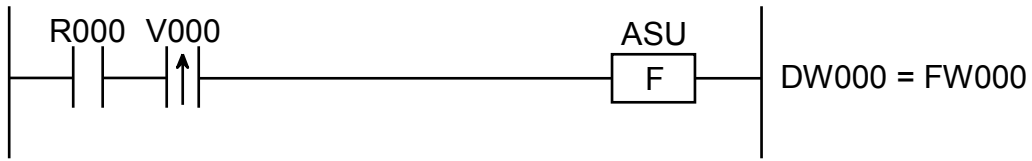
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	-	-	-	-	√
R	√	-	-	-	-	-	√

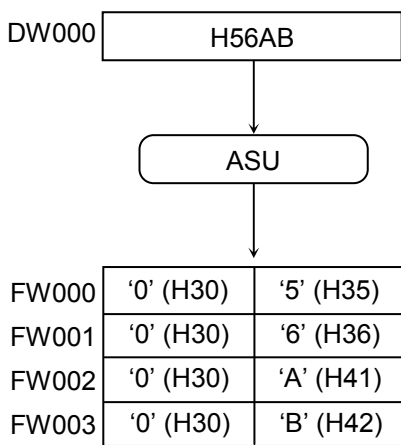
√: May be specified.

-: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) makes a transition from OFF to ON state, the ASU instruction converts the binary data content of DW000 to hexadecimal ASCII data format in unpack mode only once and stores the result in FW000.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.



# APB ASCII-TO-BINARY CONVERSION IN PACK MODE

## (1) Input format



where:

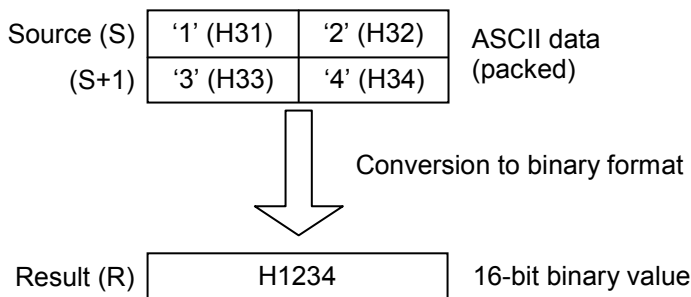
S: (Source) is an ASCII data storage register.

R: (Result) is an operation result (binary data) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

The APB instruction converts a packed hexadecimal ASCII data value specified in Source (S) to 16-bit binary data format and stores the result in Result (R).



<Correspondence between binary and ASCII data>

Binary	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ASCII	H30	H31	H32	H33	H34	H35	H36	H37	H38	H39	H41	H42	H43	H44	H45	H46

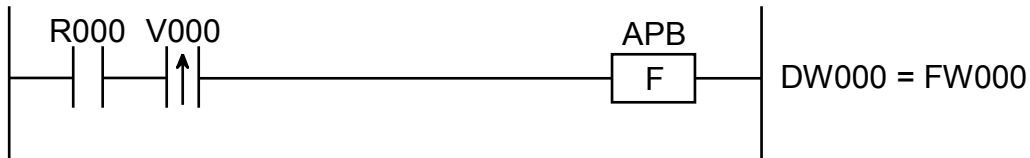
## (3) Data types

\	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	-	-	-	-	-	√
R	√	-	-	-	-	-	√

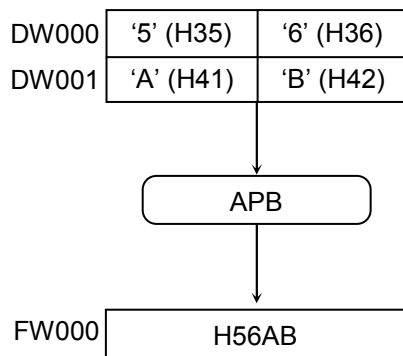
√: May be specified.

-: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) makes a transition from OFF to ON state, the APB instruction converts the unpacked hexadecimal ASCII data content of DW000 to binary data format only once and stores the result in FW000.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	-

where:

E: Set to 1 if data other than hexadecimal ASCII data (H30 through H39 and H41 through H46) is detected in Source (S); otherwise, set to 0.

All the other flags then E remain unchanged.

- If the E-flag is set, Result (R) remains unchanged.

# AUB ASCII-TO-BINARY CONVERSION IN UNPACK MODE

## (1) Input format

AUB S -> R
------------

where:

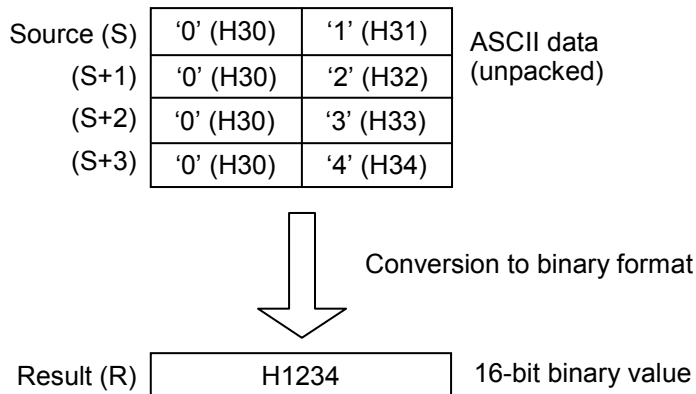
S: (Source) is an ASCII data storage register.

R: (Result) is an operation result (binary data) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

The AUB instruction converts an unpacked hexadecimal ASCII data value specified in Source (S) to 16-bit binary data format and stores the result in Result (R).



The upper bytes at Source (S) and subsequent locations up to (S+3) may be set to any value.

<Correspondence between binary and ASCII data>

Binary	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ASCII	H30	H31	H32	H33	H34	H35	H36	H37	H38	H39	H41	H42	H43	H44	H45	H46

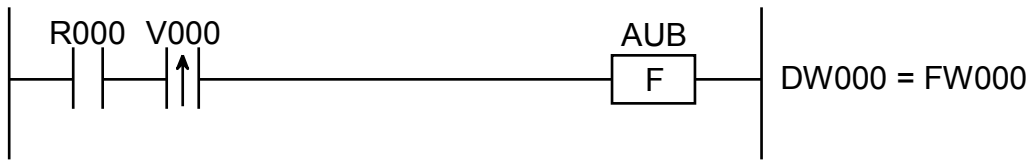
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	–	–	–	–	–	√
R	√	–	–	–	–	–	√

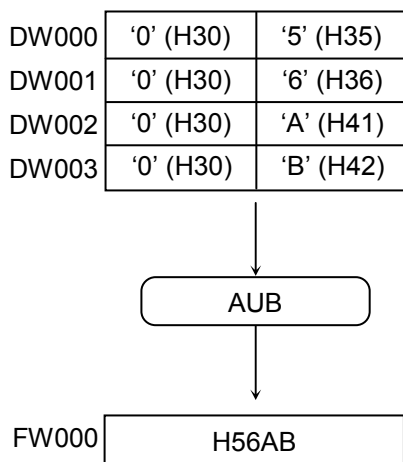
√: May be specified.

–: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) makes a transition from OFF to ON state, the AUB instruction converts the unpacked hexadecimal ASCII data content of DW000 to binary data format only once and stores the result in FW000.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	-

where:

E: Set to 1 if data other than hexadecimal ASCII data (H30 through H39 and H41 through H46) is detected in Source (S); otherwise, set to 0.

All the other flags then E remain unchanged.

- If the E-flag is set, Result (R) remains unchanged.

# STD SINGLE-TO-DOUBLE CONVERSION

## (1) Input format

STD S -> R

where:

S: (Source) is a 16-bit binary data storage register.

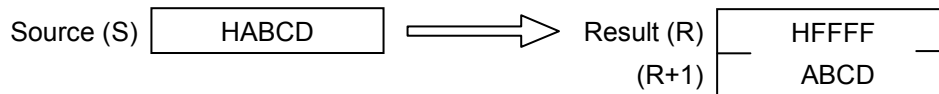
R: (Result) is an operation result (32-bit binary data) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

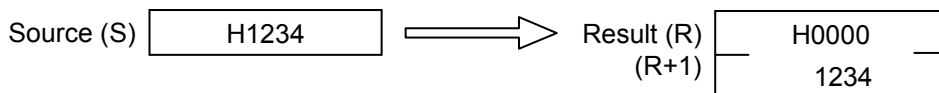
## (2) Function

The STD instruction converts a signed 16-bit binary data value specified in Source (S) to 32-bit binary data format, expanding the sign bit assignment, and stores the result in Result (R).

- When the sign bit is set:



- When the sign bit is reset:



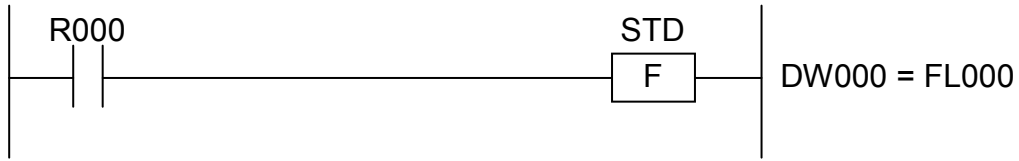
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	–	–	–	–	√
R	–	–	√	–	–	–	√

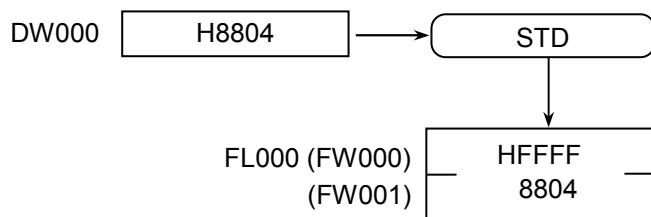
√: May be specified.

–: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the STD instruction converts the 16-bit binary data content of DW000 to 32-bit binary data format, extending the sign bit assignment, and stores the result in FL000.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# DTS DOUBLE-TO-SINGLE CONVERSION

## (1) Input format

DTS S -> R

where:

S: (Source) is a 32-bit binary data storage register or a 32-bit binary constant.

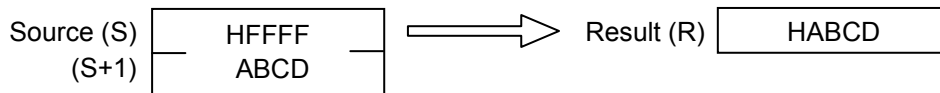
R: (Result) is an operation result (16-bit binary data) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

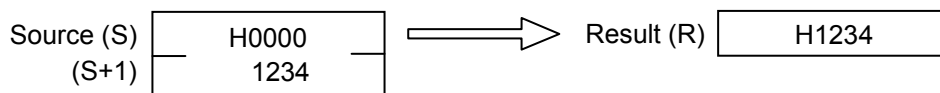
## (2) Function

The DTS instruction converts a 32-bit binary data value specified in Source (S) to 16-bit binary data format and stores the result in Result (R).

- When the sign bit is set:



- When the sign bit is reset:



## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	-	-	√	√	-	-	√
R	√	-	-	-	-	-	√

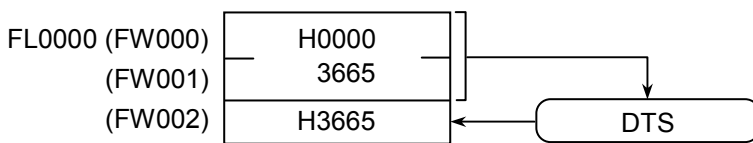
√: May be specified.

-: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the DTS instruction converts the 32-bit binary data content of FL000 to 16-bit binary data format and stores the result in FW002.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	↕

where:

V: Set to 0 if Source (S) is in the range -32768 to 32767; otherwise, set to 1.

All the other flags then V remain unchanged.

- If an overflow occurs in the operation (the V-flag is set), one of the following full-scale values will be stored in Result (R):

When Source (S) > 32767:	H7FFF
When Source (S) < -32767:	H8000



# ABS ABSOLUTE VALUE

## (1) Input format



where:

S: (Source) is a source storage register or a constant.

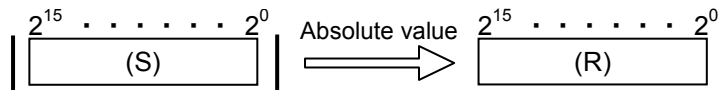
R: (Result) is an operation result (absolute value) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

### ● Absolute value of word data

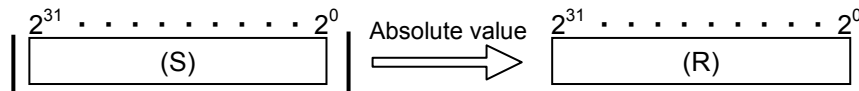
The ABS instruction obtains the absolute value of a 16-bit data value specified in Source (S) and stores the result in Result (R).



where the values that may be specified in Source (S) and stored in Result (R) are in the range -32768 to 32767.

### ● Absolute value of long-word data

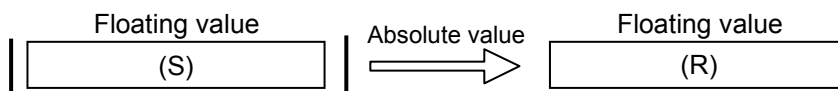
The ABS instruction obtains the absolute value of a 32-bit data value specified in Source (S) and stores the result in Result (R).



where the values that may be specified in Source (S) and stored in Result (R) are in the range -2147483648 to 2147483647.

### ● Absolute value of floating data

The ABS instruction obtains the absolute value of a floating data value specified in Source (S) and stores the result in Result (R).



where the values that may be specified in Source (S) and stored in Result (R) are in the range:

$$0, \pm 2^{-126} \text{ to } \leq \pm 2^{128}$$

## (3) Data types

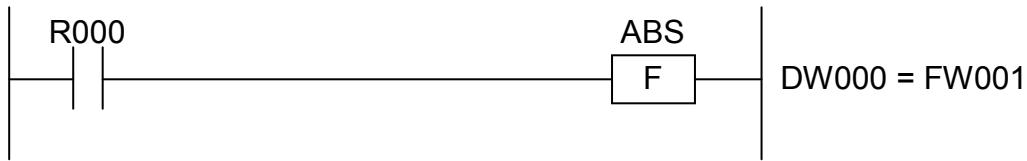
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
R	√	-	√	-	√	-	√

√: May be specified.

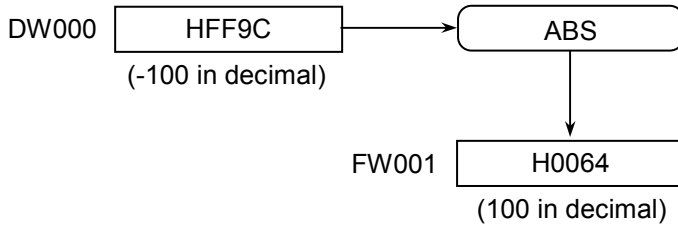
-: May not be specified.

The types of S and R must be the same (i.e., either word, long-word, or floating). If the two are of different types, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the ABS instruction obtains the absolute value of the content of DW000 and stores the result in FW001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	↕

where:

V: When the type of given data is word:

- Set to 1 if Source (S) equals -32768; otherwise, set to 0.

When it is long-word:

- Set to 1 if Source (S) equals -2147483648; otherwise, set to 0.

When it is floating:

- Not affected by the result of the operation performed; it remains unchanged.

All the other flags then V remain unchanged.

- If an overflow occurs in the operation (the V-flag is set), one of the following full-scale values will be stored in Result (R):

Word	Long-word
H7FFF	H7FFFFFFF

- If a non-numeric value or infinity is specified in Source (S) for a floating operation, one of the following values will be stored in Result (R) -- the E-flag remains reset in this case:

Source (S)	Result (R)
Non-numeric value	Non-numeric value
+ infinity	+ infinity
- infinity	- infinity

# NEG SIGN CHANGE

## (1) Input format



where:

S: (Source) is a data storage register or constant whose sign is to be changed.

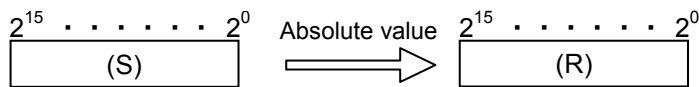
R: (Result) is an operation result (sign-changed value) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

### ● Changing the sign of word data

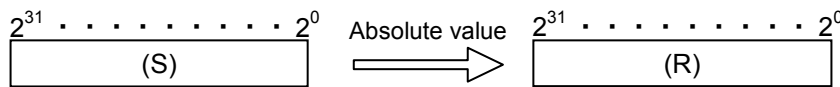
The NEG instruction changes the sign of a 16-bit data value specified in Source (S) and stores the result in Result (R).



where the values that may be specified in Source (S) and stored in Result (R) are in the range -32768 to 32767.

### ● Changing the sign of long-word data

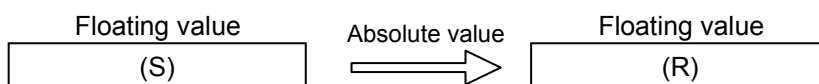
The NEG instruction changes the sign of a 32-bit data value specified in Source (S) and stores the result in Result (R).



where the values that may be specified in Source (S) and stored in Result (R) are in the range -2147483648 to 2147483647.

### ● Changing the sign of floating data

The NEG instruction changes the sign of a floating data value specified in Source (S) and stores the result in Result (R).



where the values that may be specified in Source (S) and stored in Result (R) are in the range:

$$0, \pm 2^{-126} \text{ to } \pm 2^{128}$$

## (3) Data types

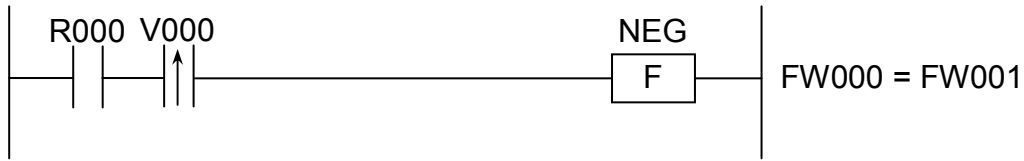
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
R	√	-	√	-	√	-	√

√: May be specified.

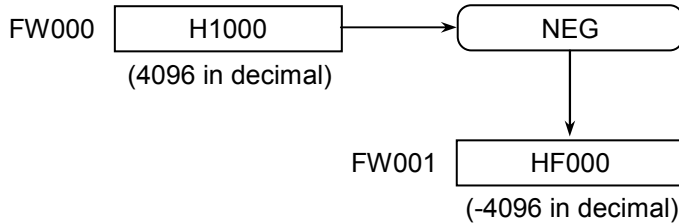
-: May not be specified.

The types of S and R must be the same (i.e., either word, long-word, or floating). If the two are of different types, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) makes a transition from OFF to ON state, the NEG instruction changes the sign of the content of FW000 only once and stores the result in FW001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	↕

where:

V: When the type of given data is word:

- Set to 1 if Source (S) equals -32768; otherwise, set to 0.

When it is long-word:

- Set to 1 if Source (S) equals -2147483648; otherwise, set to 0.

When it is floating:

- Not affected by the result of the operation performed; it remains unchanged.

All the other flags then V remain unchanged.

- If an overflow occurs in the operation (the V-flag is set), one of the following full-scale values will be stored in Result (R):

Word	Long-word
H7FFF	H7FFFFFFF

- If a non-numeric value or infinity is specified in Source (S) for a floating operation, one of the following values will be stored in Result (R) -- the E-flag remains reset in this case:

Source (S)	Result (R)
Non-numeric value	Non-numeric value
+ infinity	- infinity
- infinity	+ infinity

## (1) Input format

DCD S -> R

where:

S: (Source) is a data storage register or constant to be decoded.

R: (Result) is an operation result (decoded value) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

- Decoding of word data
  - The DCD instruction decodes the low-order 4 bits of a data value specified in Source (S) and sets the resulting bit in Result (R).
  - Only the low-order 4 bits of a data value specified in Source (S) are effective.
  - The values that may be specified in Source (S) are in the range 0 to 15.
  
- Decoding of long-word data
  - The DCD instruction decodes the low-order 5 bits of a data value specified in Source (S) and sets the resulting bit in Result (R).
  - Only the low-order 5 bits of a data value specified in Source (S) are effective.
  - The values that may be specified in Source (S) are in the range 0 to 31.

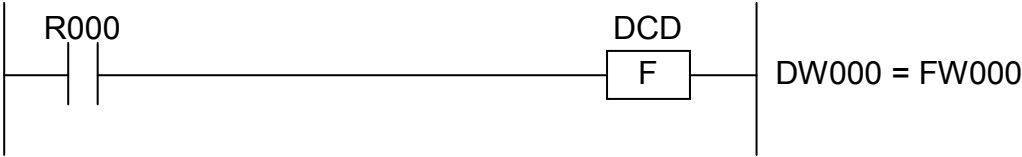
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	–	–	–	–	√
R	√	–	√	–	–	–	√

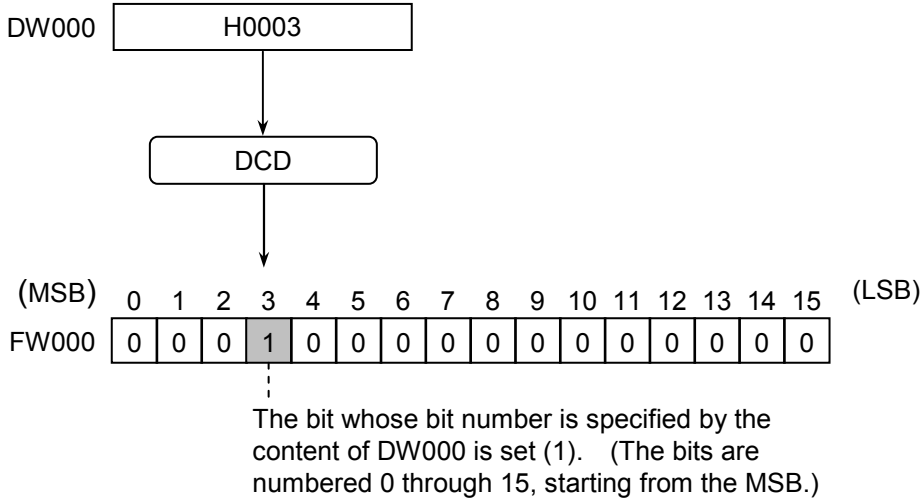
√: May be specified.

–: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the DCD instruction decodes the content of DW000 and sets the resulting bit in FW000.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# ECD ENCODE

---

## (1) Input format

ECD S -> R
------------

where:

S: (Source) is a data storage register or constant to be encoded.

R: (Result) is an operation result (encoded value) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

- The ECD instruction encodes a data value specified in Source (S) and stores the resulting value in Result (R).
- If Source (S) equals 0, this instruction performs nothing, the content of Result (R) remaining unchanged.

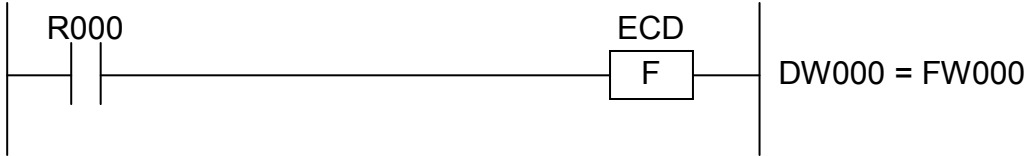
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	–	–	√
R	√	–	–	–	–	–	√

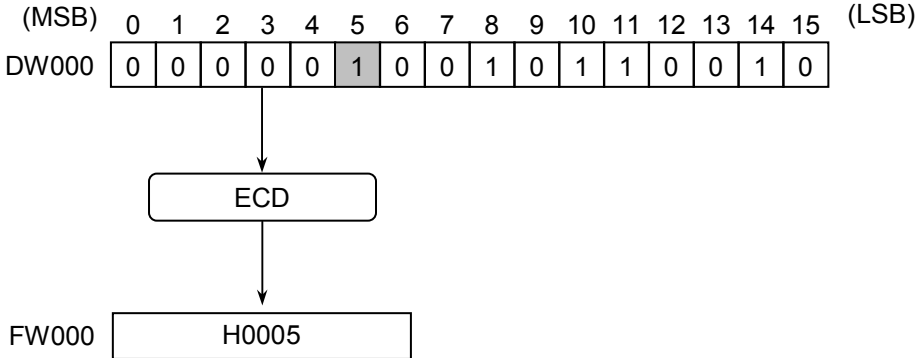
√: May be specified.

–: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the ECD instruction encodes the content of DW000 and stores the resulting value in FW000.



The bit number of the first 1-bit that is found in DW000 by scanning its content from MSB towards LSB is stored in FW000.

(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	-

where:

E: Set to 1 if Source (S) equals 0; otherwise, set to 0.

All the other flags then E remain unchanged.



# LSR LOGICAL SHIFT RIGHT

## (1) Input format

LSR S : D -> R

where:

S: (Source) is a data storage register or constant to be shifted.

D: (Destination) is a shift-bit-count storage register or a constant.

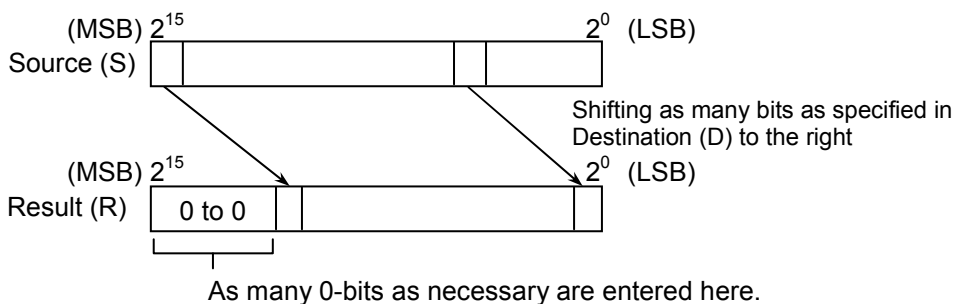
R: (Result) is an operation result (shifted value) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

### ● Shifting word data right

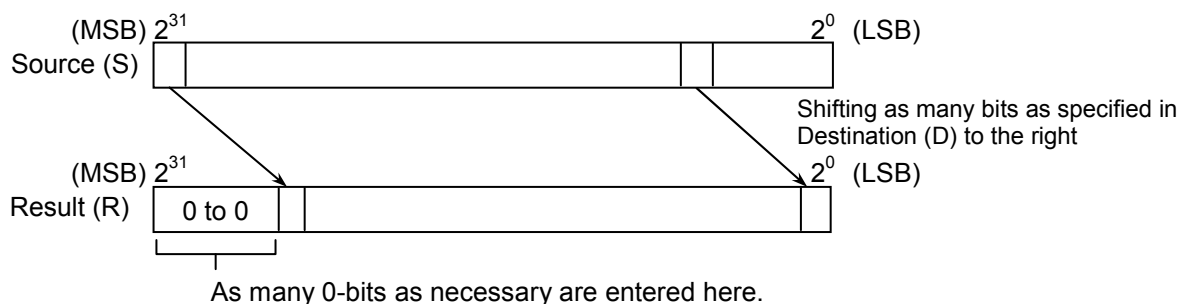
The LSR instruction shifts a 16-bit data value specified in Source (S) as many bits as specified in Destination (D) to the right and stores the resulting value in Result (R).



- As the shift-bit-count, only the low-order 4 bits of a data value specified in Destination (D) are effective.
- The values that may be specified in Destination (D) are in the range 0 to 15.

### ● Shifting long-word data right

The LSR instruction shifts a 32-bit data value specified in Source (S) as many bits as specified in Destination (D) to the right and stores the resulting value in Result (R).



- As the shift-bit-count, only the low-order 5 bits of a data value specified in Destination (D) are effective.
- The values that may be specified in Destination (D) are in the range 0 to 31.

(3) Data types

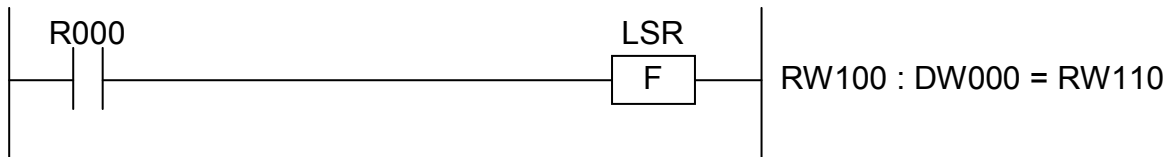
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	-	-	√
D	√	√	-	-	-	-	√
R	√	-	√	-	-	-	√

√: May be specified.

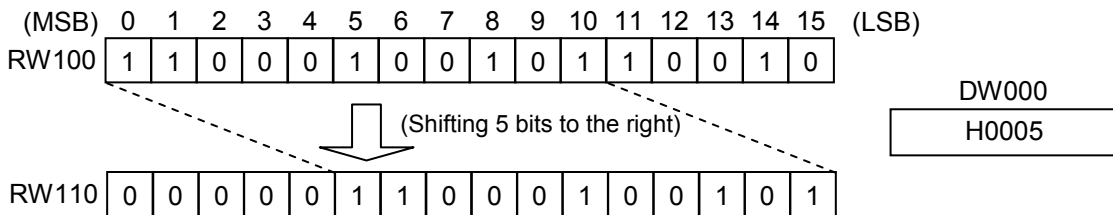
-: May not be specified.

The types of S and R must be the same (i.e., either word or long-word). If the two are of different types, an input error will result. The type of D must always be word.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the LSR instruction shifts the content of RW100 as many bits as specified in DW000 to the right and stores the resulting value in RW110.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# LSL LOGICAL SHIFT LEFT

## (1) Input format

```
LSL S : D -> R
```

where:

S: (Source) is a data storage register or constant to be shifted.

D: (Destination) is a shift-bit-count storage register or a constant.

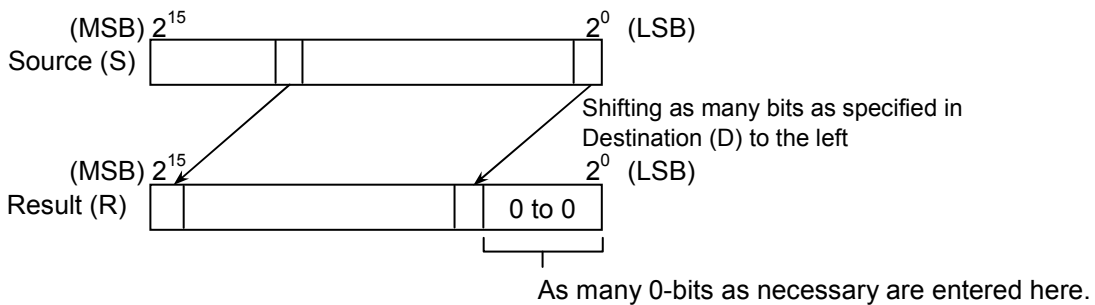
R: (Result) is an operation result (shifted value) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

### ● Shifting word data left

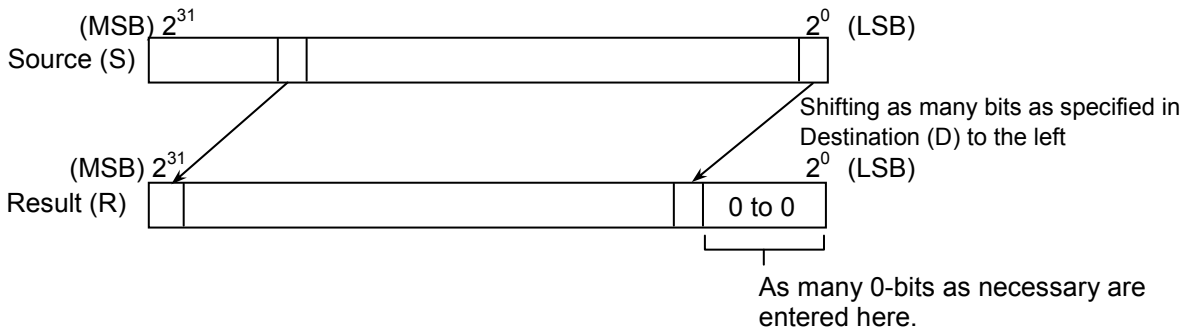
The LSL instruction shifts a 16-bit data value specified in Source (S) as many bits as specified in Destination (D) to the left and stores the resulting value in Result (R).



- As the shift-bit-count, only the low-order 4 bits of a data value specified in Destination (D) are effective.
- The values that may be specified in Destination (D) are in the range 0 to 15.

### ● Shifting long-word data left

The LSL instruction shifts a 32-bit data value specified in Source (S) as many bits as specified in Destination (D) to the left and stores the resulting value in Result (R).



- As the shift-bit-count, only the low-order 5 bits of a data value specified in Destination (D) are effective.
- The values that may be specified in Destination (D) are in the range 0 to 31.

(3) Data types

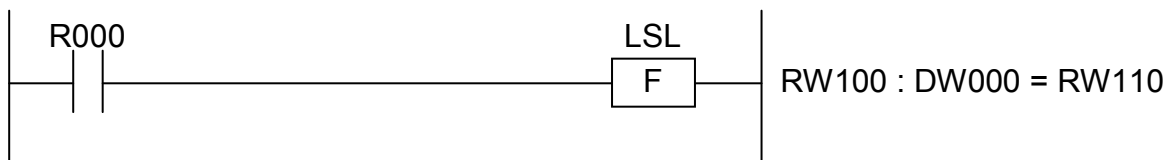
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	-	-	√
D	√	√	-	-	-	-	√
R	√	-	√	-	-	-	√

√: May be specified.

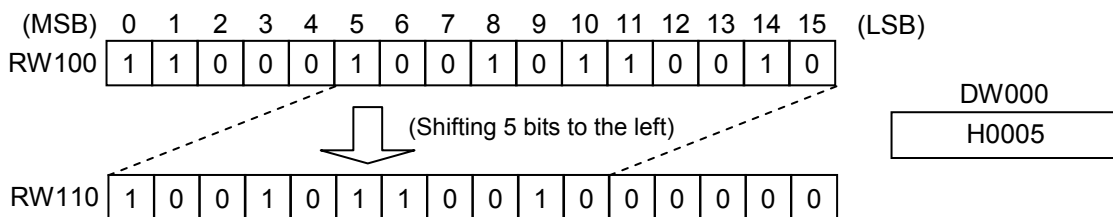
-: May not be specified.

The types of S and R must be the same (i.e., either word or long-word). If the two are of different types, an input error will result. The type of D must always be word.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the LSL instruction shifts the content of RW100 as many bits as specified in DW000 to the left and stores the resulting value in RW110.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# ASR ARITHMETIC SHIFT RIGHT

## (1) Input format

ASR S : D -> R
----------------

where:

S: (Source) is a data storage register or constant to be shifted.

D: (Destination) is a shift-bit-count storage register or a constant.

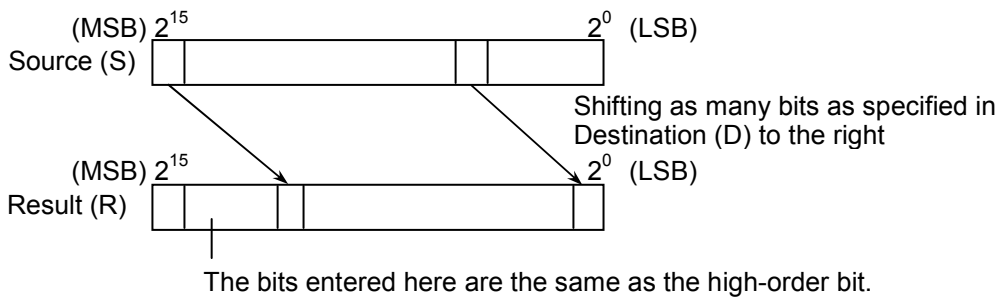
R: (Result) is an operation result (shifted value) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

### ● Shifting word data right

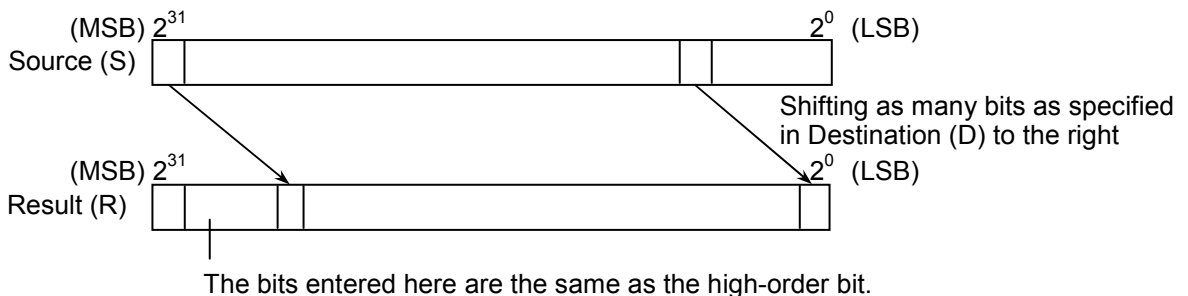
The ASR instruction shifts a 16-bit data value specified in Source (S) as many bits as specified in Destination (D) to the right, keeping the sign bit as is, and stores the resulting value in Result (R).



- As the shift-bit-count, only the low-order 4 bits of a data value specified in Destination (D) are effective.
- The values that may be specified in Destination (D) are in the range 0 to 15.

### ● Shifting long-word data right

The ASR instruction shifts a 32-bit data value specified in Source (S) as many bits as specified in Destination (D) to the right, keeping the sign bit as is, and stores the resulting value in Result (R).



- As the shift-bit-count, only the low-order 5 bits of a data value specified in Destination (D) are effective.
- The values that may be specified in Destination (D) are in the range 0 to 31.

(3) Data types

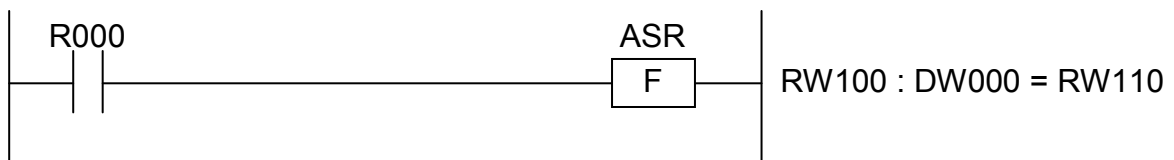
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	-	-	√
D	√	√	-	-	-	-	√
R	√	-	√	-	-	-	√

√: May be specified.

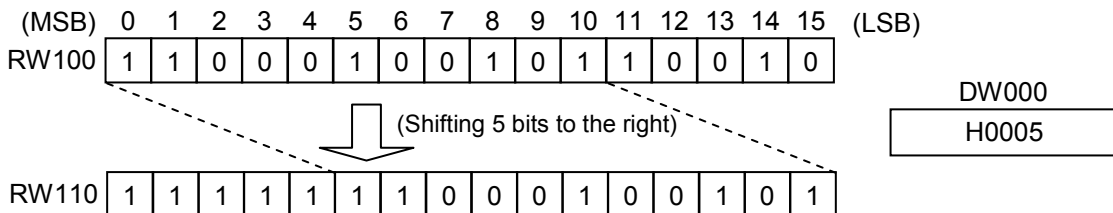
-: May not be specified.

The types of S and R must be the same (i.e., either word or long-word). If the two are of different types, an input error will result. The type of D must always be word.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the ASR instruction shifts the content of RW100 as many bits as specified in DW000 to the right and stores the resulting value in RW110.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# ASL ARITHMETIC SHIFT LEFT

## (1) Input format

```
ASL S : D -> R
```

where:

S: (Source) is a data storage register or constant to be shifted.

D: (Destination) is a shift-bit-count storage register or a constant.

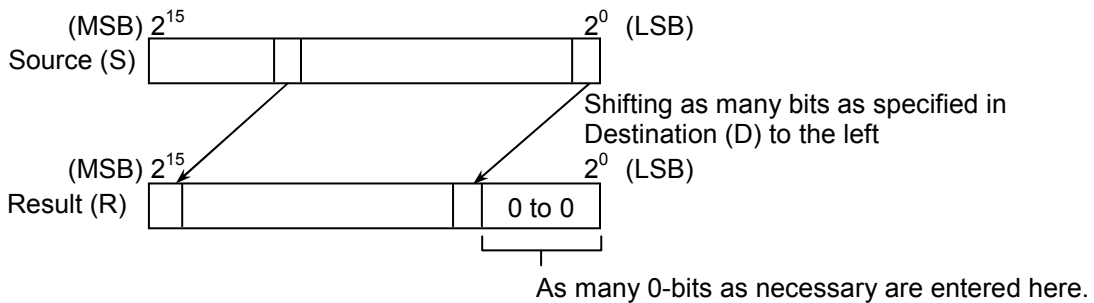
R: (Result) is an operation result (shifted value) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

### ● Shifting word data left

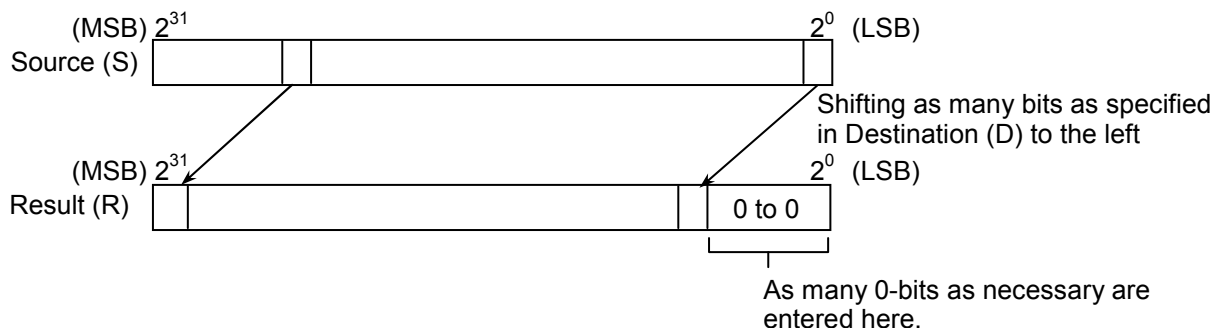
The ASL instruction shifts a 16-bit data value specified in Source (S) as many bits as specified in Destination (D) to the left and stores the resulting value in Result (R).



- As the shift-bit-count, only the low-order 4 bits of a data value specified in Destination (D) are effective.
- The values that may be specified in Destination (D) are in the range 0 to 15.

### ● Shifting long-word data left

The ASL instruction shifts a 32-bit data value specified in Source (S) as many bits as specified in Destination (D) to the left and stores the resulting value in Result (R).



- As the shift-bit-count, only the low-order 5 bits of a data value specified in Destination (D) are effective.
- The values that may be specified in Destination (D) are in the range 0 to 31.

(3) Data types

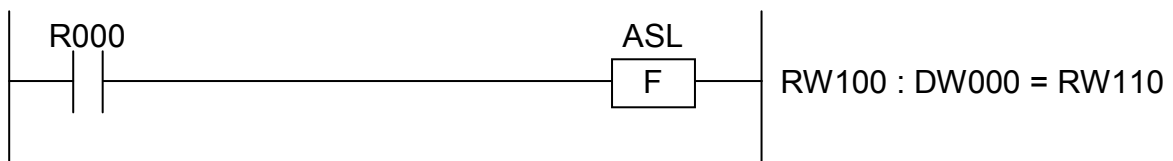
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	-	-	√
D	√	√	-	-	-	-	√
R	√	-	√	-	-	-	√

√: May be specified.

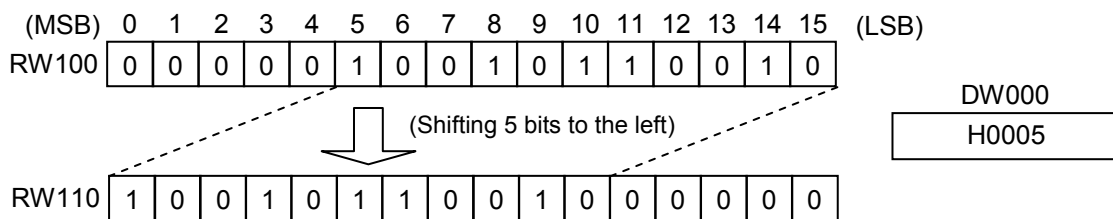
-: May not be specified.

The types of S and R must be the same (i.e., either word or long-word). If the two are of different types, an input error will result. The type of D must always be word.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the ASL instruction shifts the content of RW100 as many bits as specified in DW000 to the left and stores the resulting value in RW110.



(5) Error handling

● Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	↑↓

where:

V: Set to 1 if the sign bit changes at least once during the shift operation; otherwise, set to 0.

All the other flags then V remain unchanged.

- If an overflow occurs in the operation (the V-flag is set), one of the following full-scale values will be stored in Result (R):

	Word	Long-word
When (S) > 0:	H7FFF	H7FFFFFFF
When (S) < 0:	H8000	H80000000



# ROR ROTATE RIGHT

## (1) Input format

ROR S : D -> R
----------------

where:

S: (Source) is a data storage register or constant to be rotated.

D: (Destination) is a rotating-bit-count storage register or a constant.

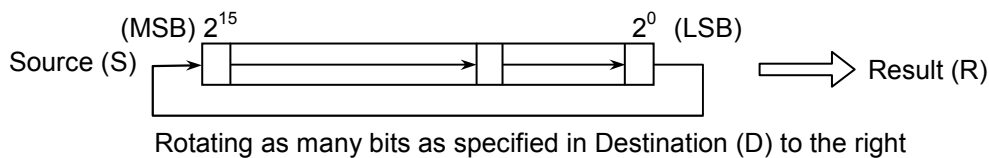
R: (Result) is an operation result (rotated value) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

### ● Rotating word data right

The ROR instruction rotates a 16-bit data value specified in Source (S) as many bits as specified in Destination (D) to the right and stores the resulting value in Result (R).



- As the rotating-bit-count, only the low-order 4 bits of a data value specified in Destination (D) are effective.
- The values that may be specified in Destination (D) are in the range 0 to 15.

### ● Rotating long-word data right

The ROR instruction rotates a 32-bit data value specified in Source (S) as many bits as specified in Destination (D) to the right and stores the resulting value in Result (R).



- As the rotating-bit-count, only the low-order 5 bits of a data value specified in Destination (D) are effective.
- The values that may be specified in Destination (D) are in the range 0 to 31.

(3) Data types

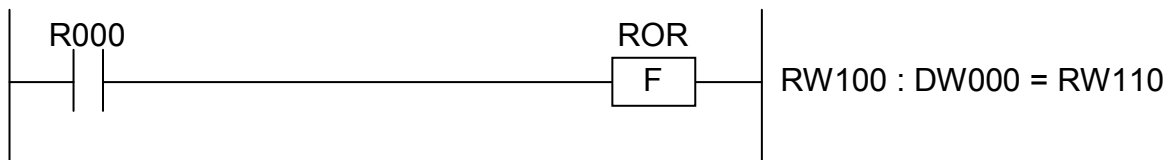
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	-	-	√
D	√	√	-	-	-	-	√
R	√	-	√	-	-	-	√

√: May be specified.

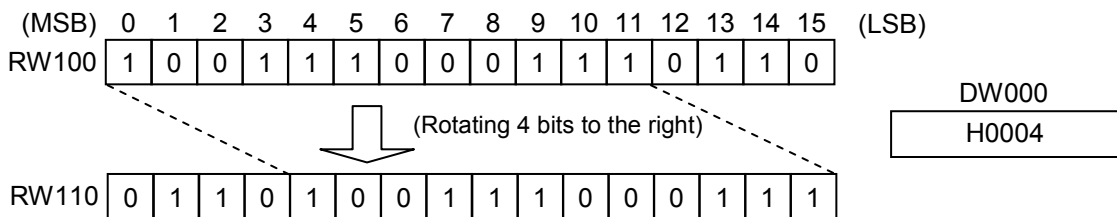
-: May not be specified.

The types of S and R must be the same (i.e., either word or long-word). If the two are of different types, an input error will result. The type of D must always be word.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the ROR instruction rotates the content of RW100 as many bits as specified in DW000 to the right and stores the resulting value in RW110.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# ROL ROTATE LEFT

## (1) Input format

ROL S : D -> R
----------------

where:

S: (Source) is a data storage register or constant to be rotated.

D: (Destination) is a rotating-bit-count storage register or a constant.

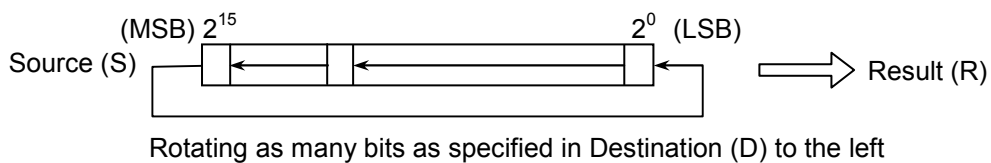
R: (Result) is an operation result (rotated value) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

### ● Rotating word data left

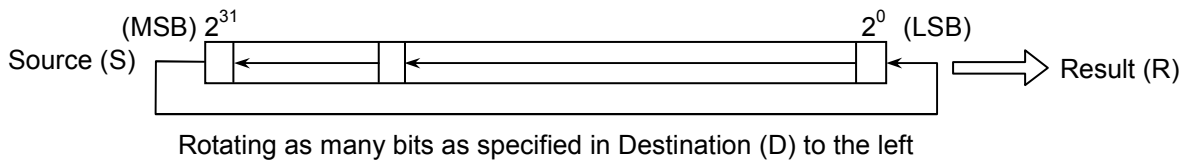
The ROL instruction rotates a 16-bit data value specified in Source (S) as many bits as specified in Destination (D) to the left and stores the resulting value in Result (R).



- As the rotating-bit-count, only the low-order 4 bits of a data value specified in Destination (D) are effective.
- The values that may be specified in Destination (D) are in the range 0 to 15.

### ● Rotating long-word data left

The ROL instruction rotates a 32-bit data value specified in Source (S) as many bits as specified in Destination (D) to the left and stores the resulting value in Result (R).



- As the rotating-bit-count, only the low-order 5 bits of a data value specified in Destination (D) are effective.
- The values that may be specified in Destination (D) are in the range 0 to 31.

(3) Data types

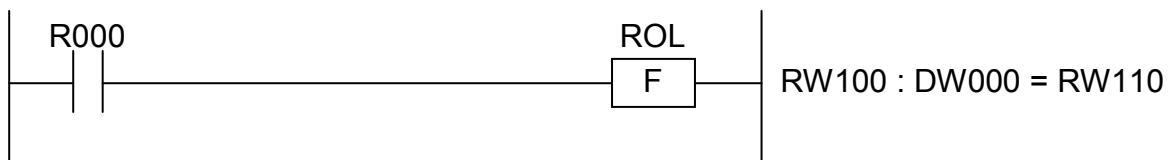
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	-	-	√
D	√	√	-	-	-	-	√
R	√	-	√	-	-	-	√

√: May be specified.

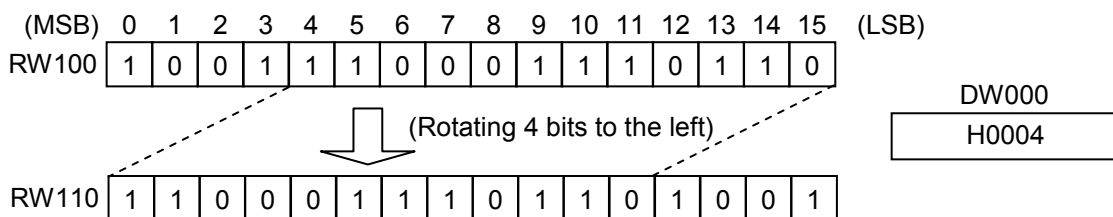
-: May not be specified.

The types of S and R must be the same (i.e., either word or long-word). If the two are of different types, an input error will result. The type of D must always be word.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the ROL instruction rotates the content of RW100 as many bits as specified in DW000 to the left and stores the resulting value in RW110.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

## (1) Input format

LIM S : D1 : D2 -> R

where:

S: (Source) is an input-value storage register or a constant.

D1: (Destination 1) is an upper-limit-value storage register or a constant.

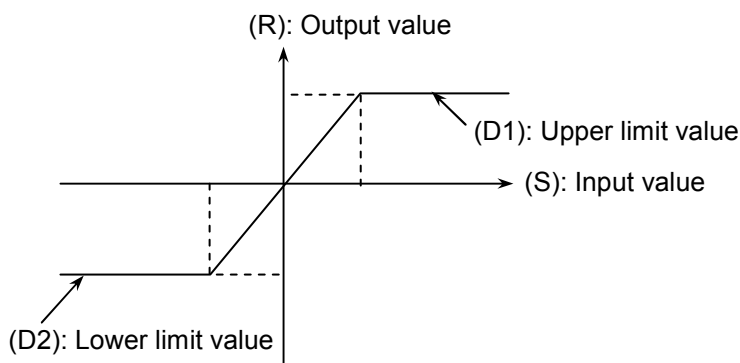
D2: (Destination 2) is a lower-limit-value storage register or a constant.

R: (Result) is an operation result (limit-controlled output value) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

The LIM instruction checks if an input value specified in Source (S) is within the upper and lower limits specified in Destination 1 (D1) and Destination 2 (D2), and stores in Result (R) an output value that is controlled within those limits.



### ● Limit control over word data

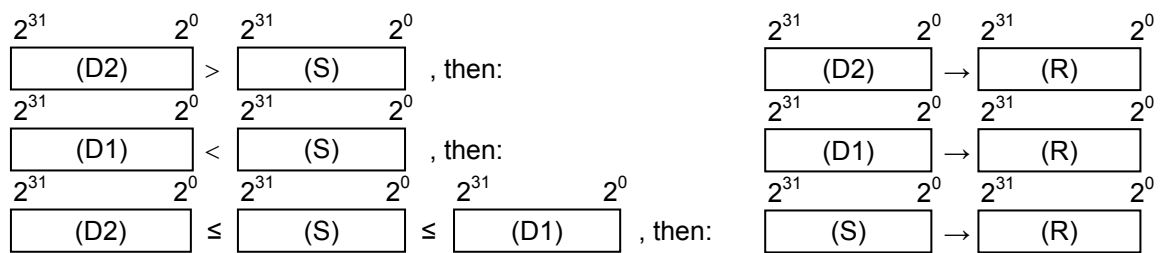
- The LIM instruction exerts limit control over 16-bit data values in the following way:

$$\begin{array}{l}
 \begin{array}{c} 2^{15} \\ \boxed{(D2)} \\ 2^0 \end{array} > \begin{array}{c} 2^{15} \\ \boxed{(S)} \\ 2^0 \end{array}, \text{ then: } \begin{array}{c} 2^{15} \\ \boxed{(D2)} \\ 2^0 \end{array} \rightarrow \begin{array}{c} 2^{15} \\ \boxed{(R)} \\ 2^0 \end{array} \\
 \begin{array}{c} 2^{15} \\ \boxed{(D1)} \\ 2^0 \end{array} < \begin{array}{c} 2^{15} \\ \boxed{(S)} \\ 2^0 \end{array}, \text{ then: } \begin{array}{c} 2^{15} \\ \boxed{(D1)} \\ 2^0 \end{array} \rightarrow \begin{array}{c} 2^{15} \\ \boxed{(R)} \\ 2^0 \end{array} \\
 \begin{array}{c} 2^{15} \\ \boxed{(D2)} \\ 2^0 \end{array} \leq \begin{array}{c} 2^{15} \\ \boxed{(S)} \\ 2^0 \end{array} \leq \begin{array}{c} 2^{15} \\ \boxed{(D1)} \\ 2^0 \end{array}, \text{ then: } \begin{array}{c} 2^{15} \\ \boxed{(S)} \\ 2^0 \end{array} \rightarrow \begin{array}{c} 2^{15} \\ \boxed{(R)} \\ 2^0 \end{array}
 \end{array}$$

- The values that may be specified in Source (S), Destination 1 (D1), and Destination 2 (D2) are in the range -32768 to 32767.

● Limit control over long-word data

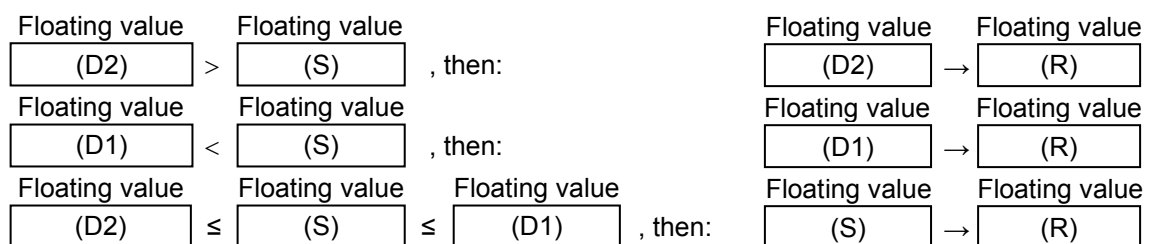
- The LIM instruction exerts limit control over 32-bit data values in the following way:



- The values that may be specified in Source (S), Destination 1 (D1), and Destination 2 (D2) are in the range -2147483648 to 2147483647.

● Limit control over floating data

- The LIM instruction exerts limit control over floating data values in the following way:



- The values that may be specified in Source (S), Destination 1 (D1), and Destination 2 (D2) are in the range:  
 $0, \pm 2^{-126}$  to  $\pm 2^{128}$

(3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D1	√	√	√	√	√	√	√
D2	√	√	√	√	√	√	√
R	√	–	√	–	√	–	√

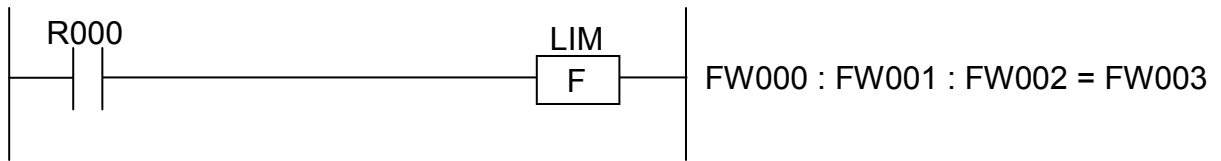
√: May be specified.

–: May not be specified.

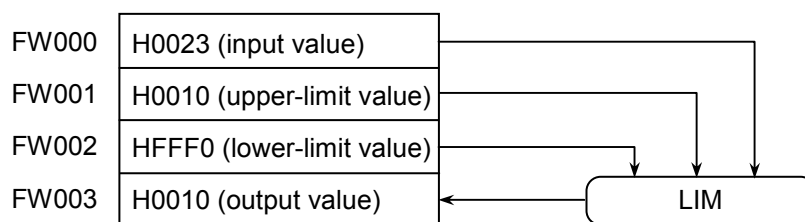
The types of S, D1, D2, and R must be the same (i.e., either word, long-word, or floating). If the four are of different types, an input error will result.

## LIM LIMITER

### (4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the LIM instruction checks if the content of FW000 is within the limits specified in FW001 and FW002, and stores in FW003 an output value that is controlled within those limits.



### (5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	-

where:

E: Set to 1 if Destination 1 (D1) is smaller than Destination 2 (D2); otherwise, set to 0.  
All the other flags then E remain unchanged.

- If the E-flag is set, this instruction does not make a check against (D1) and (D2).

**This Page Intentionally Left Blank**



# BND DEAD BAND

## (1) Input format

```
BND S : D1 : D2 -> R
```

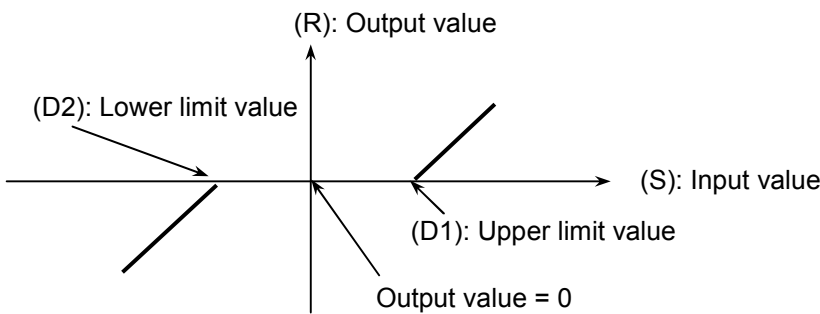
where:

- S: (Source) is a dead-band input-value storage register or a constant.
- D1: (Destination 1) is a dead-band upper-limit-value storage register or a constant.
- D2: (Destination 2) is a dead-band lower-limit-value storage register or a constant.
- R: (Result) is an operation result (dead-band-controlled output value) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

The BND instruction checks if an input value specified in Source (S) is within the upper and lower limits of dead band specified in Destination 1 (D1) and Destination 2 (D2), and stores in Result (R) an output value that is controlled within those limits -- that is, if the input value is within the limits (dead band), a value of zero (0) is stored in Result (R).



### ● Dead-band control over word data

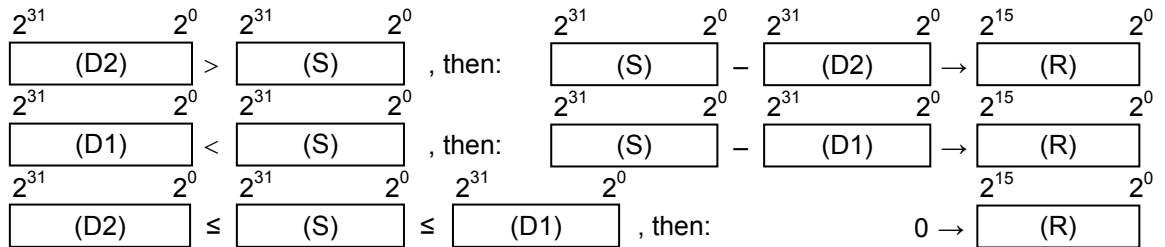
- The BND instruction exerts dead-band control over 16-bit data values in the following way:

$$\begin{array}{l}
 \begin{array}{c} 2^{15} \quad 2^0 \\ \boxed{(D2)} \end{array} > \begin{array}{c} 2^{15} \quad 2^0 \\ \boxed{(S)} \end{array}, \text{ then:} \\
 \begin{array}{c} 2^{15} \quad 2^0 \\ \boxed{(D1)} \end{array} < \begin{array}{c} 2^{15} \quad 2^0 \\ \boxed{(S)} \end{array}, \text{ then:} \\
 \begin{array}{c} 2^{15} \quad 2^0 \\ \boxed{(D2)} \end{array} \leq \begin{array}{c} 2^{15} \quad 2^0 \\ \boxed{(S)} \end{array} \leq \begin{array}{c} 2^{15} \quad 2^0 \\ \boxed{(D1)} \end{array}, \text{ then:}
 \end{array}
 \qquad
 \begin{array}{c}
 \begin{array}{c} 2^{15} \quad 2^0 \\ \boxed{(S)} \end{array} - \begin{array}{c} 2^{15} \quad 2^0 \\ \boxed{(D2)} \end{array} \rightarrow \begin{array}{c} 2^{15} \quad 2^0 \\ \boxed{(R)} \end{array} \\
 \begin{array}{c} 2^{15} \quad 2^0 \\ \boxed{(S)} \end{array} - \begin{array}{c} 2^{15} \quad 2^0 \\ \boxed{(D1)} \end{array} \rightarrow \begin{array}{c} 2^{15} \quad 2^0 \\ \boxed{(R)} \end{array} \\
 0 \rightarrow \begin{array}{c} 2^{15} \quad 2^0 \\ \boxed{(R)} \end{array}
 \end{array}$$

- The values that may be specified in Source (S), Destination 1 (D1), and Destination 2 (D2) are in the range -32768 to 32767.

● Dead-band control over long-word data

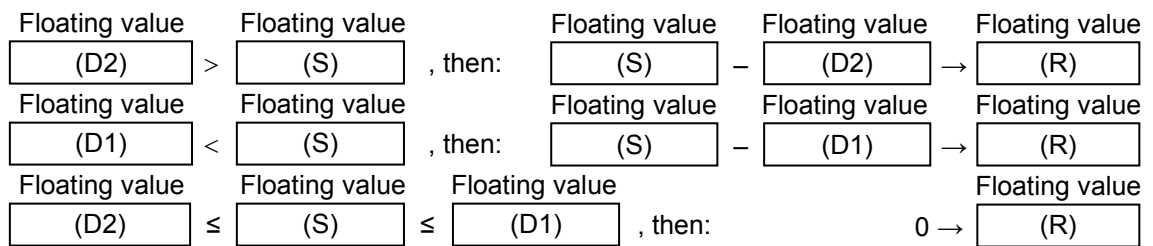
- The BND instruction exerts dead-band control over 32-bit data values in the following way:



- The values that may be specified in Source (S), Destination 1 (D1), and Destination 2 (D2) are in the range -2147483648 to 2147483647.

● Dead-band control over floating data

- The BND instruction exerts dead-band control over floating data values in the following way:



- The values that may be specified in Source (S), Destination 1 (D1), and Destination 2 (D2) are in the range:  
 $0, \pm 2^{-126}$  to  $\pm 2^{128}$

(3) Data types

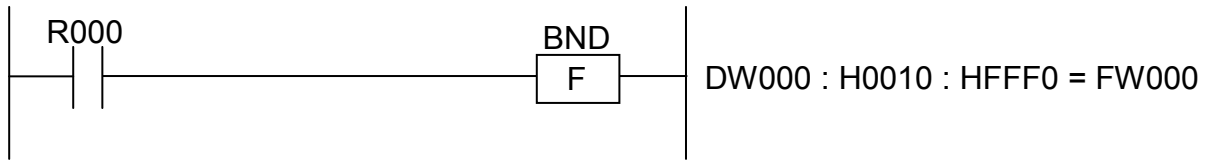
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D1	√	√	√	√	√	√	√
D2	√	√	√	√	√	√	√
R	√	–	√	–	√	–	√

√: May be specified.

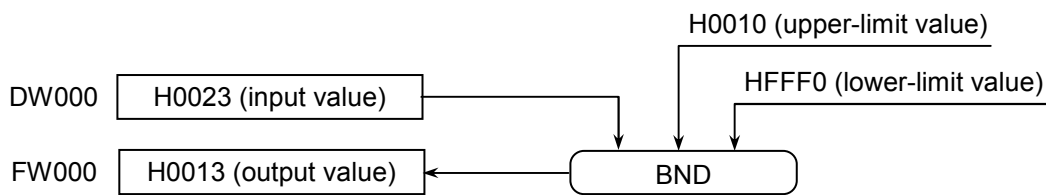
–: May not be specified.

The types of S, D1, D2, and R must be the same (i.e., either word, long-word, or floating). If the four are of different types, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the BND instruction checks if the content of DW000 is within the limits specified by the constants H0010 and HFFF0, and stores in FW000 an output value that is controlled in reference to the dead band.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	↕

where:

V: When the type of given data is word:

- Set to 0 if Result (R) is in the range -32768 to 32767; otherwise, set to 1.

When it is long-word:

- Set to 0 if Result (R) is in the range -2147483648 to 2147483647; otherwise, set to 1.

When it is floating:

- Not affected by the result of the operation performed; it remains unchanged.

E: When the type of given data is word or long-word:

- Set to 1 if Destination 1 (D1) is smaller than Destination 2 (D2);
- Otherwise, set to 0.

When it is floating:

- Set to 1 if Result (R) is a non-zero value and out of the range shown below; otherwise, set to 0.

$$\pm 2^{-126} \text{ to } \pm 2^{128}$$

All the other flags then V and E remain unchanged.

- If  $(D1) < (D2)$ , the error flag (E-flag) is set, with the overflow flag (V-flag) reset. Result (R) remains unchanged.
- If an overflow occurs in the operation, one of the following full-scale values will be stored in Result (R):

	In case of a positive overflow:	In case of a negative overflow:
Word	H7FFF	H8000
Long-word	H7FFFFFFF	H80000000
Floating	+3.402823E38	-3.402823E38

If a floating value causes an overflow, the V-flag is not set. (The V-flag is set only if a word or long-word value causes an overflow.)

- If a floating value causes an underflow, a value of zero (0) with correct sign will be stored in Result (R), the operation result flags remaining unchanged.

# ZON DEAD ZONE

## (1) Input format

ZON S : D1 : D2 -> R

where:

S: (Source) is an input-value storage register or a constant for zone control.

D1: (Destination 1) is a positive-bias-value storage register or a constant.

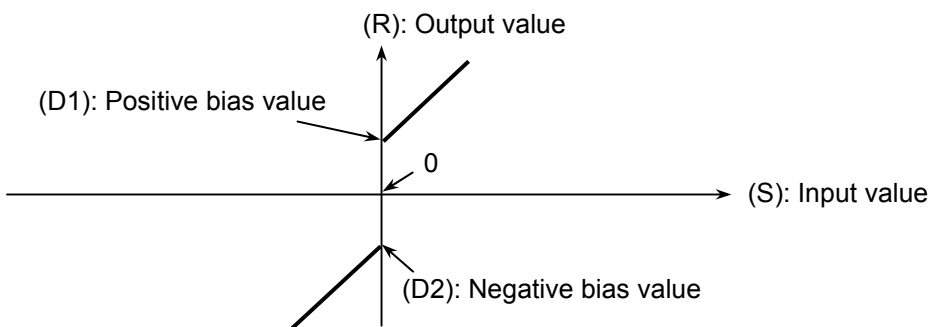
D2: (Destination 2) is a negative-bias-value storage register or a constant.

R: (Result) is an operation result (zone-controlled output value) storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

The ZON instruction adds a positive or negative bias value specified in Destination 1 (D1) or Destination (D2) to an input value specified in Source (S) and stores the resulting value in Result (R).



### ● Zone control over word data

- The ZON instruction exerts zone control over 16-bit data values in the following way:

$$\begin{matrix} 2^{15} & 2^0 \\ \boxed{\text{(S)}} \end{matrix} > 0, \text{ then: } \begin{matrix} 2^{15} & 2^0 \\ \boxed{\text{(S)}} \end{matrix} + \begin{matrix} 2^{15} & 2^0 \\ \boxed{\text{(D1)}} \end{matrix} \rightarrow \begin{matrix} 2^{15} & 2^0 \\ \boxed{\text{(R)}} \end{matrix}$$

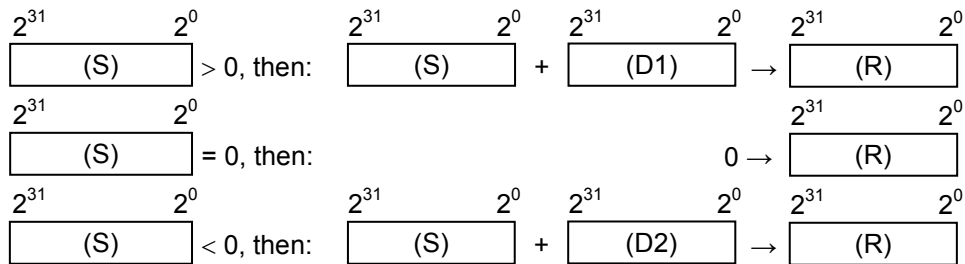
$$\begin{matrix} 2^{15} & 2^0 \\ \boxed{\text{(S)}} \end{matrix} = 0, \text{ then: } 0 \rightarrow \begin{matrix} 2^{15} & 2^0 \\ \boxed{\text{(R)}} \end{matrix}$$

$$\begin{matrix} 2^{15} & 2^0 \\ \boxed{\text{(S)}} \end{matrix} < 0, \text{ then: } \begin{matrix} 2^{15} & 2^0 \\ \boxed{\text{(S)}} \end{matrix} + \begin{matrix} 2^{15} & 2^0 \\ \boxed{\text{(D2)}} \end{matrix} \rightarrow \begin{matrix} 2^{15} & 2^0 \\ \boxed{\text{(R)}} \end{matrix}$$

- The values that may be specified in Source (S), Destination 1 (D1), and Destination 2 (D2) are in the range -32768 to 32767.

● Zone control over long-word data

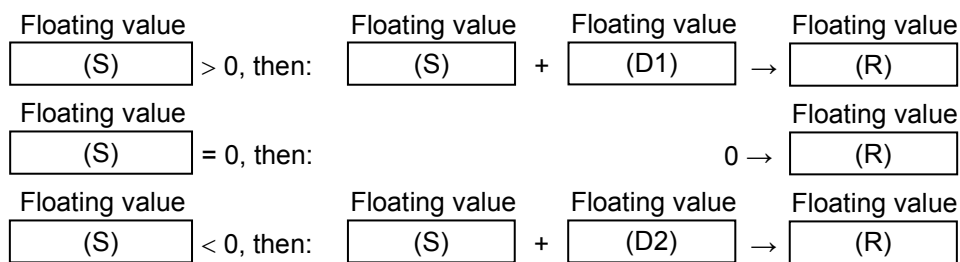
- The ZON instruction exerts zone control over 32-bit data values in the following way:



- The values that may be specified in Source (S), Destination 1 (D1), and Destination 2 (D2) are in the range -2147483648 to 2147483647.

● Zone control over floating data

- The ZON instruction exerts zone control over floating data values in the following way:



- The values that may be specified in Source (S), Destination 1 (D1), and Destination 2 (D2) are in the range:  
 $0, \pm 2^{-126}$  to  $\pm 2^{128}$

(3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D1	√	√	√	√	√	√	√
D2	√	√	√	√	√	√	√
R	√	–	√	–	√	–	√

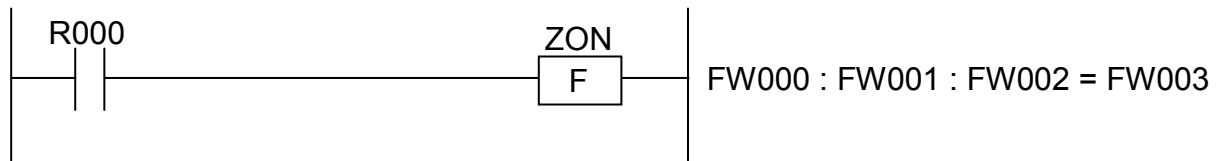
√: May be specified.

–: May not be specified.

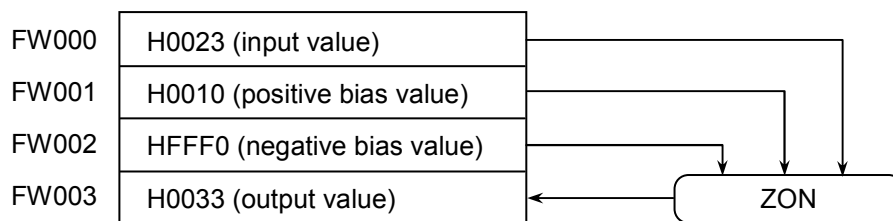
The types of S, D1, D2, and R must be the same (i.e., either word, long-word, or floating). If the four are of different types, an input error will result.

## ZON DEAD ZONE

### (4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the ZON instruction adds the content of FW001 or FW002 to the content of FW000 and stores in FW003 an output value that is controlled in reference to the dead zone.



### (5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	↕

where:

V: When the type of given data is word:

- Set to 0 if Result (R) is in the range -32768 to 32767; otherwise, set to 1.

When it is long-word:

- Set to 0 if Result (R) is in the range -2147483648 to 2147483647; otherwise, set to 1.

When it is floating:

- Not affected by the result of the operation performed; it remains unchanged.

E: When the type of given data is word or long-word:

- Set to 1 if Destination 1 (D1) is smaller than Destination 2 (D2);
- Otherwise, set to 0.

When it is floating:

- Set to 1 if Result (R) is a non-zero value and out of the range shown below; otherwise, set to 0.

$$\pm 2^{-126} \text{ to } \pm 2^{128}$$

All the other flags then V and E remain unchanged.

- If  $(D1) < (D2)$ , the error flag (E-flag) is set, with the overflow flag (V-flag) reset. Result (R) remains unchanged.
- If an overflow occurs in the operation, one of the following full-scale values will be stored in Result (R):

	In case of a positive overflow:	In case of a negative overflow:
Word	H7FFF	H8000
Long-word	H7FFFFFFF	H80000000
Floating	+3.402823E38	-3.402823E38

If a floating value causes an overflow, the V-flag is not set. (The V-flag is set only if a word or long-word value causes an overflow.)

- If a floating value causes an underflow, a value of zero (0) with correct sign will be stored in Result (R), the operation result flags remaining unchanged.



# SQR SQUARE ROOT

## (1) Input format



where:

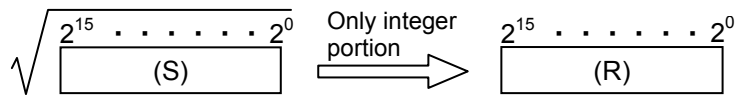
S: (Source) is a data storage register or a constant from which to compute a square root.  
 R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

### ● Square root of word data

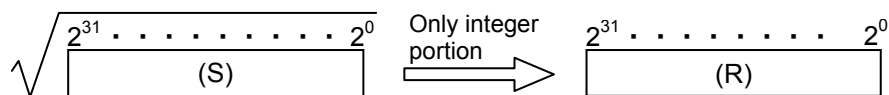
- The SQR instruction computes the square root of a 16-bit data value specified in Source (S) and stores only the integer portion of the result in Result (R).



- If Source (S) is smaller than zero (0), the instruction stores a value of zero (0) in Result (R).
- The values that may be specified in Source (S) are within the range -32768 to 32767.

### ● Square root of long-word data

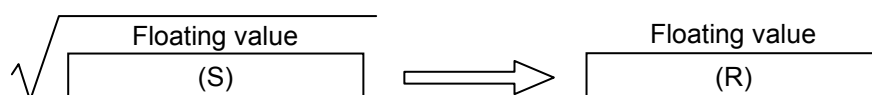
- The SQR instruction computes the square root of a 32-bit data value specified in Source (S) and stores only the integer portion of the result in Result (R).



- If Source (S) is smaller than zero (0), the instruction stores a value of zero (0) in Result (R).
- The values that may be specified in Source (S) are within the range -2147483648 to 2147483647.

### ● Square root of floating data

- The SQR instruction computes the square root of a floating data value specified in Source (S) and stores the result in Result (R).



- If Source (S) is smaller than zero (0), the instruction stores a value of zero (0) in Result (R).
- The values that may be specified in Source (S) are within the range:  
 $0, \pm 2^{-126}$  to  $\pm 2^{128}$

(3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
R	√	–	√	–	√	–	√

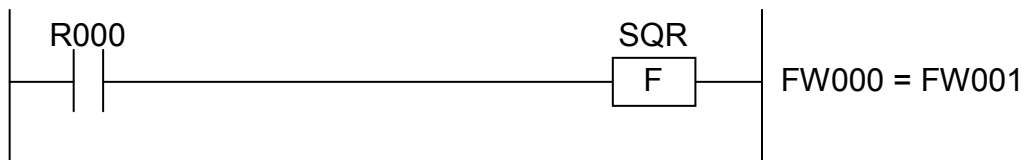
√: May be specified.

–: May not be specified.

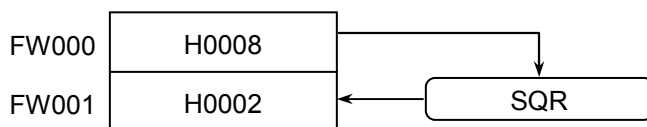
The types of S and R must be the same (i.e., either word, long-word, or floating). If the two are of different types, an input error will result.

(4) Example program

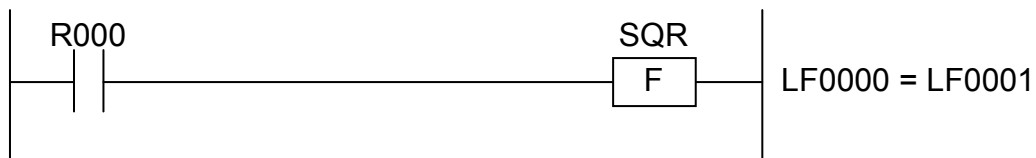
- Computing the square root of word data



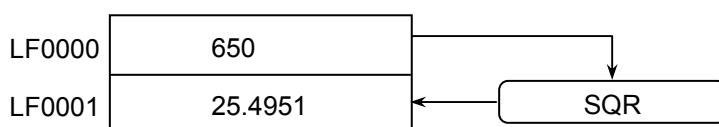
In this example, if the contact R000 (input condition) is closed (ON), the SQR instruction computes the square root of the content of FW000 and stores the result (integer portion only) in FW001.



- Computing the square root of floating data



In this example, if the contact R000 (input condition) is closed (ON), the SQR instruction computes the square root of the content of LF0000 and stores the result in LF0001.



## SQR SQUARE ROOT

---

### (5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	-

where:

E: When the type of given data is word or long-word:

- Not affected by the result of the operation performed; it remains unchanged.

When it is floating:

- Set to 1 if Result (R) is a non-zero value and out of the range shown below; otherwise, set to 0.

$$\pm 2^{-126} \text{ to } \pm 2^{128}$$

All the other flags then E remain unchanged.

- If the E-flag is set, this instruction performs no further processing. (Result (R) remains unchanged.)

**This Page Intentionally Left Blank**

# SIN SINE

---

## (1) Input format

SIN S -> R

where:

S: (Source) is an angle-data storage register or a constant from which to compute the sine of the angle.

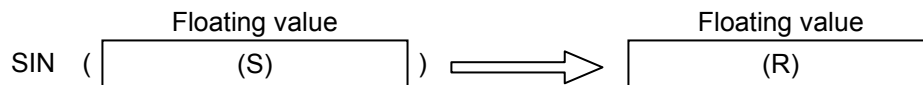
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

The SIN instruction computes the sine of an angle specified in Source (S) and stores the result in Result (R).

The angle specified in Source (S) must be expressed in radians (i.e.,  $\text{angle} \times \pi/180$ ).



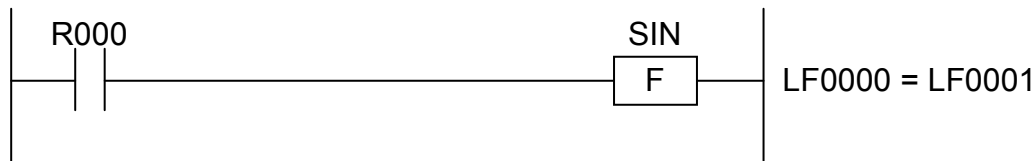
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	-	-	-	-	√	√	√
R	-	-	-	-	√	-	√

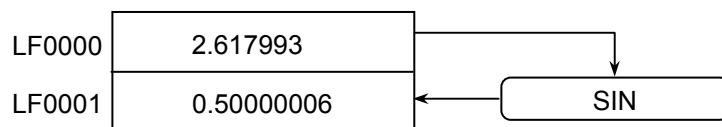
√: May be specified.

-: May not be specified.

## (4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the SIN instruction computes the sine of the content of LF0000 and stores the result in LF0001.



## (5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	-

where:

E: Set to 1 if Result (R) is a non-zero value and out of the range shown below; otherwise, set to 0.

$$\pm 2^{-126} \text{ to } \pm 2^{128}$$

All the other flags then E remain unchanged.

- If the E-flag is set, this instruction performs no further processing. (Result (R) remains unchanged.)

# COS COSINE

## (1) Input format

COS S -> R

where:

S: (Source) is an angle-data storage register or a constant from which to compute the cosine of the angle.

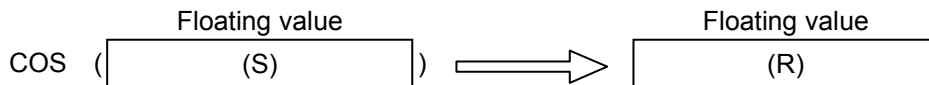
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

The COS instruction computes the cosine of an angle specified in Source (S) and stores the result in Result (R).

The angle specified in Source (S) must be expressed in radians (i.e.,  $\text{angle} \times \pi/180$ ).



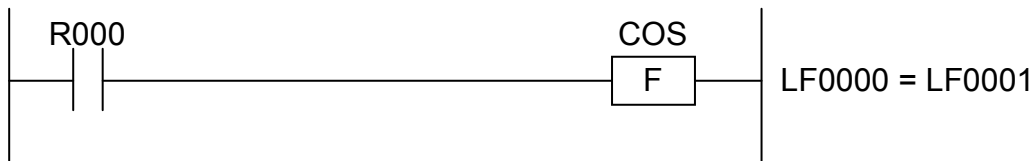
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	-	-	-	-	√	√	√
R	-	-	-	-	√	-	√

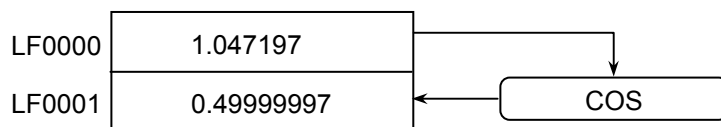
√: May be specified.

-: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the COS instruction computes the cosine of the content of LF0000 and stores the result in LF0001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	-

where:

E: Set to 1 if Result (R) is a non-zero value and out of the range shown below; otherwise, set to 0.

$$\pm 2^{-126} \text{ to } \pm 2^{128}$$

All the other flags then E remain unchanged.

- If the E-flag is set, this instruction performs no further processing. (Result (R) remains unchanged.)



# TAN TANGENT

## (1) Input format

TAN S -> R
------------

where:

S: (Source) is an angle-data storage register or a constant from which to compute the tangent of the angle.

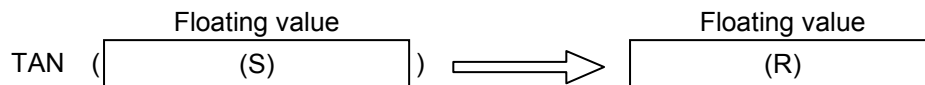
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

The TAN instruction computes the tangent of an angle specified in Source (S) and stores the result in Result (R).

The angle specified in Source (S) must be expressed in radians (i.e.,  $\text{angle} \times \pi/180$ ).



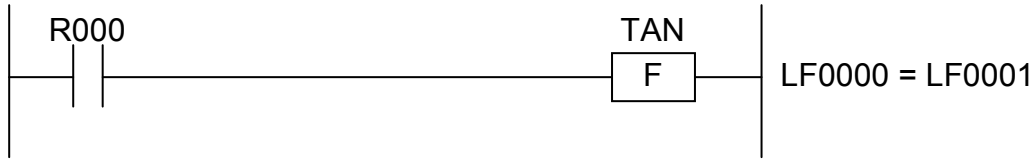
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	-	-	-	-	√	√	√
R	-	-	-	-	√	-	√

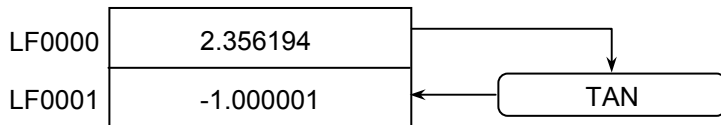
√: May be specified.

-: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the TAN instruction computes the tangent of the content of LF0000 and stores the result in LF0001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	-

where:

E: Set to 1 if Result (R) is a non-zero value and out of the range shown below; otherwise, set to 0.

$$\pm 2^{-126} \text{ to } \pm 2^{128}$$

All the other flags then E remain unchanged.

- If the E-flag is set, this instruction performs no further processing. (Result (R) remains unchanged.)

# ASIN ARC SINE (SIN<sup>-1</sup>)

## (1) Input format

ASIN S -> R

where:

S: (Source) is an angle-data storage register or a constant from which to compute an arc sine.

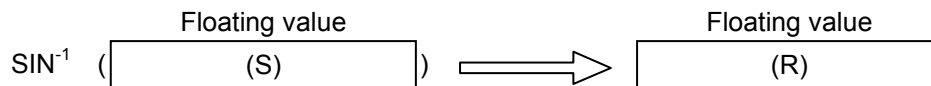
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

The ASIN instruction computes an angle from a sine value specified in Source (S) and stores the result in Result (R).

The sine values that may be specified in Source (S) are within the range -1.0 to 1.0.



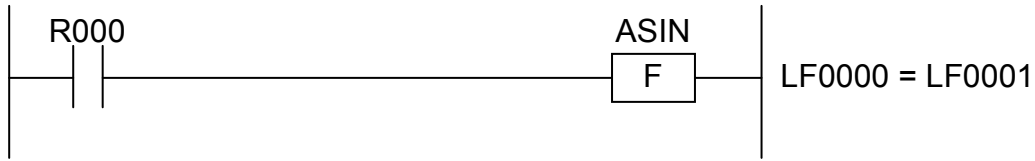
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	-	-	-	-	√	√	√
R	-	-	-	-	√	-	√

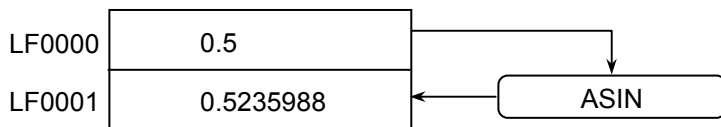
√: May be specified.

-: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the ASIN instruction computes an arc sine from the content of LF0000 and stores the result in LF0001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	-

where:

E: Set to 1 if the value specified in Source (S) is out of the range -1.0 to 1.0; otherwise, set to 0. In addition, it is also set to 1 if Result (R) is a non-zero value and out of the range shown below; otherwise, it is set to 0.

$$\pm 2^{-126} \text{ to } \pm 2^{128}$$

All the other flags then E remain unchanged.

- If the E-flag is set, this instruction performs no further processing. (Result (R) remains unchanged.)

# ACOS ARC COSINE (COS<sup>-1</sup>)

## (1) Input format

ACOS S -> R

where:

S: (Source) is an angle-data storage register or a constant from which to compute an arc cosine.

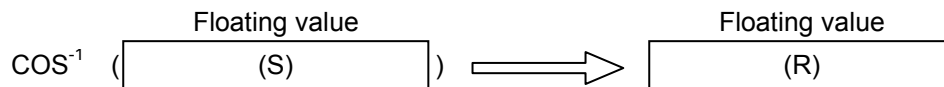
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

The ACOS instruction computes an angle from a cosine value specified in Source (S) and stores the result in Result (R).

The cosine values that may be specified in Source (S) are within the range -1.0 to 1.0.



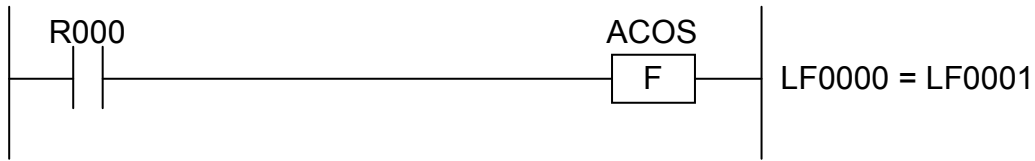
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	-	-	-	-	√	√	√
R	-	-	-	-	√	-	√

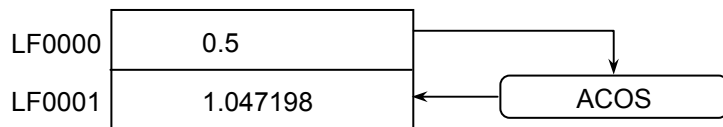
√: May be specified.

-: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the ACOS instruction computes an arc cosine from the content of LF0000 and stores the result in LF0001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	-

where:

E: Set to 1 if the value specified in Source (S) is out of the range -1.0 to 1.0; otherwise, set to 0. In addition, it is also set to 1 if Result (R) is a non-zero value and out of the range shown below; otherwise, it is set to 0.

$$\pm 2^{-126} \text{ to } \pm 2^{128}$$

All the other flags then E remain unchanged.

- If the E-flag is set, this instruction performs no further processing. (Result (R) remains unchanged.)

# ATAN ARC TANGENT ( $TAN^{-1}$ )

## (1) Input format

ATAN S -> R

where:

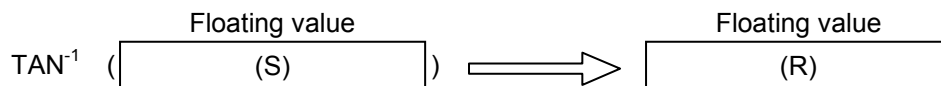
S: (Source) is an angle-data storage register or a constant from which to compute an arc tangent.

R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

The ATAN instruction computes an angle from a tangent value specified in Source (S) and stores the result in Result (R).



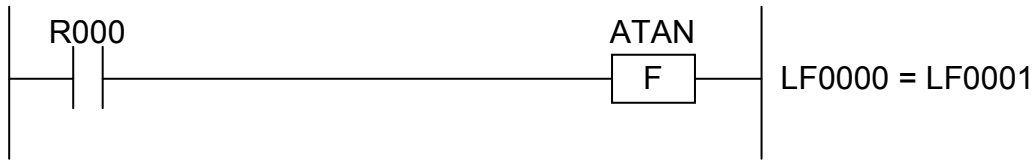
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	-	-	-	-	√	√	√
R	-	-	-	-	√	-	√

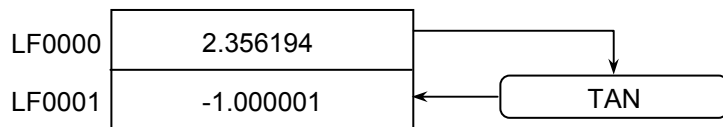
√: May be specified.

-: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the ATAN instruction computes an arc tangent from the content of LF0000 and stores the result in LF0001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	-

where:

E: Set to 1 if Result (R) is a non-zero value and out of the range shown below; otherwise, it is set to 0.

$$\pm 2^{-126} \text{ to } \pm 2^{128}$$

All the other flags then E remain unchanged.

- If the E-flag is set, this instruction performs no further processing. (Result (R) remains unchanged.)



# EXP EXPONENTIAL

## (1) Input format

EXP S -> R

where:

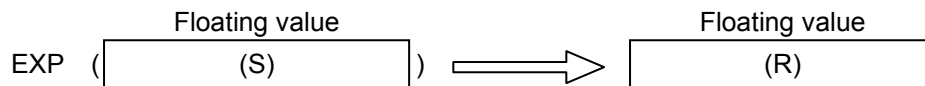
S: (Source) is a power-data storage register or a constant to which to raise e.

R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

The EXP instruction raises e (=2.71828...) to a power specified in Source (S) and stores the result in Result (R).



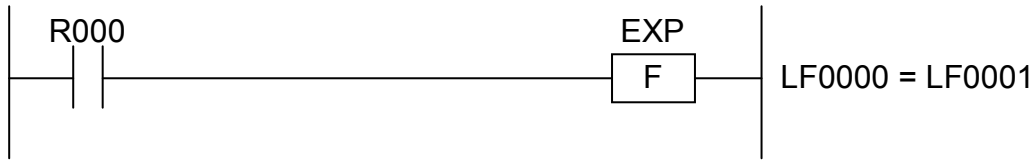
## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	-	-	-	-	√	√	√
R	-	-	-	-	√	-	√

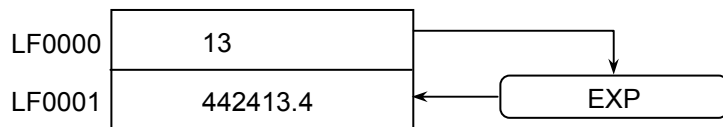
√: May be specified.

-: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the EXP instruction raises e to the content of LF0000 (power) and stores the result in LF0001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	-

where:

E: Set to 1 if Result (R) is a non-zero value and out of the range shown below; otherwise, it is set to 0.

$$\pm 2^{-126} \text{ to } \pm 2^{128}$$

All the other flags then E remain unchanged.

- If the E-flag is set, this instruction performs no further processing. (Result (R) remains unchanged.)

# LOG NATURAL LOGARITHM

---

## (1) Input format

LOG S -> R

where:

S: (Source) is a data storage register or constant from which to compute a natural logarithm.

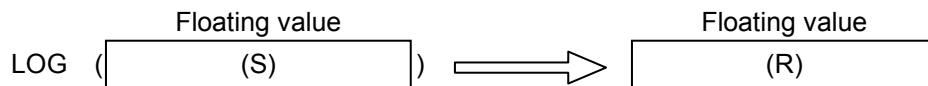
R: (Result) is an operation result storage register.

Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbol “->” may be omitted.

## (2) Function

The LOG instruction computes the logarithm of a data value specified in Source (S) to the base e (e = 2.71828...) and stores the result in Result (R).

The values that may be specified in Source (S) are limited to positive integers.



## (3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	-	-	-	-	√	√	√
R	-	-	-	-	√	-	√

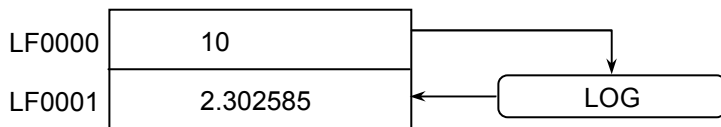
√: May be specified.

-: May not be specified.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the LOG instruction computes the natural logarithm of the content of LF0000 and stores the result in LF0001.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	↕	-	-	-	-

where:

E: Set to 1 if Source (S) is a negative value; and set to 0 if it is a positive value. In addition, it is also set to 1 if Result (R) is a non-zero value and out of the range shown below; otherwise, it is set to 0.

$$\pm 2^{-126} \text{ to } \pm 2^{128}$$

All the other flags then E remain unchanged.

- If the E-flag is set, this instruction performs no further processing. (Result (R) remains unchanged.)

# MAX MAXIMUM VALUE

## (1) Input format

MAX S : D -> R

where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register or a constant.

R: (Result) is an operation result (maximum value) storage register.

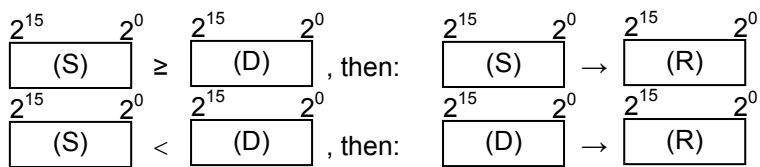
Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

The MAX instruction compares the data values specified in Source (S) and Destination (D) and stores the larger value in Result (R).

### ● Obtaining maximum values of type word

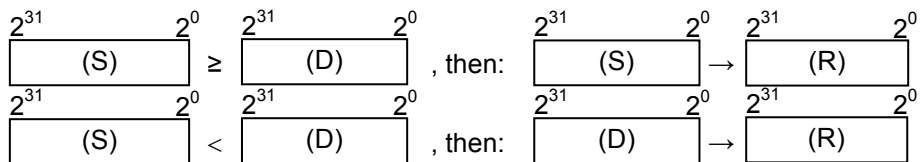
- The MAX instruction compares two given 16-bit data values in the following way and stores the larger value in Result (R).



- The values that may be specified in Source (S) and Destination (D) are within the range -32768 to 32767.

### ● Obtaining maximum values of type long-word

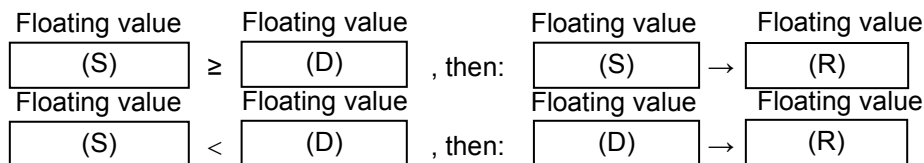
- The MAX instruction compares two given 32-bit data values in the following way and stores the larger value in Result (R).



- The values that may be specified in Source (S) and Destination (D) are within the range -2147483648 to 2147483647.

### ● Obtaining maximum values of type floating

- The MAX instruction compares two given floating data values in the following way and stores the larger value in Result (R).



- The values that may be specified in Source (S) and Destination (D) are within the range:  $0, \pm 2^{-126}$  to  $\pm 2^{128}$

(3) Data types

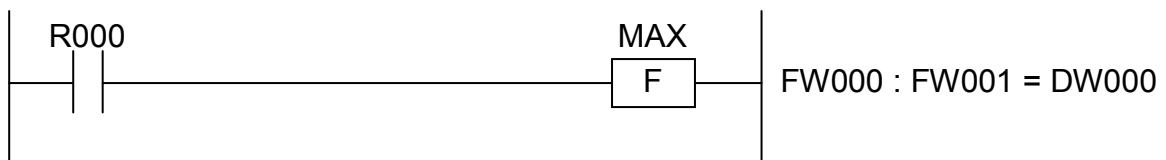
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D	√	√	√	√	√	√	√
R	√	-	√	-	√	-	√

√: May be specified.

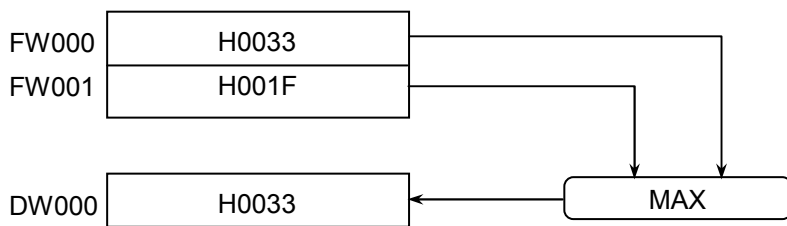
-: May not be specified.

The types of S, D, and R must be the same (i.e., either word, long-word, or floating). If the three are of different types, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the MAX instruction compares the contents of FW000 and FW001 and stores the larger value in DW000.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# MIN MINIMUM VALUE

## (1) Input format



where:

S: (Source) is a source storage register or a constant.

D: (Destination) is a destination storage register or a constant.

R: (Result) is an operation result (minimum value) storage register.

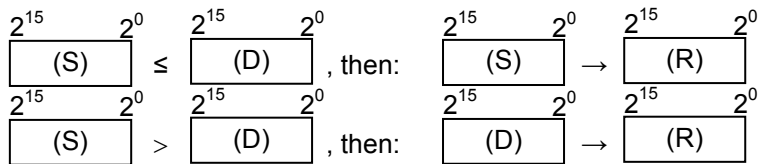
Note: Spaces must be inserted between the function name and the first parameter and between the parameters. The symbols “:” and “->” may be omitted.

## (2) Function

The MIN instruction compares the data values specified in Source (S) and Destination (D) and stores the smaller value in Result (R).

### ● Obtaining minimum values of type word

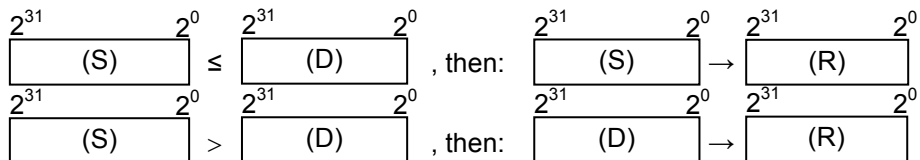
- The MIN instruction compares two given 16-bit data values in the following way and stores the smaller value in Result (R).



- The values that may be specified in Source (S) and Destination (D) are within the range -32768 to 32767.

### ● Obtaining minimum values of type long-word

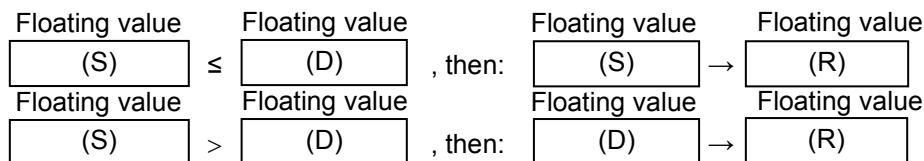
- The MIN instruction compares two given 32-bit data values in the following way and stores the smaller value in Result (R).



- The values that may be specified in Source (S) and Destination (D) are within the range -2147483648 to 2147483647.

### ● Obtaining minimum values of type floating

- The MIN instruction compares two given floating data values in the following way and stores the smaller value in Result (R).



- The values that may be specified in Source (S) and Destination (D) are within the range:  $0, \pm 2^{-126}$  to  $\pm 2^{128}$

(3) Data types

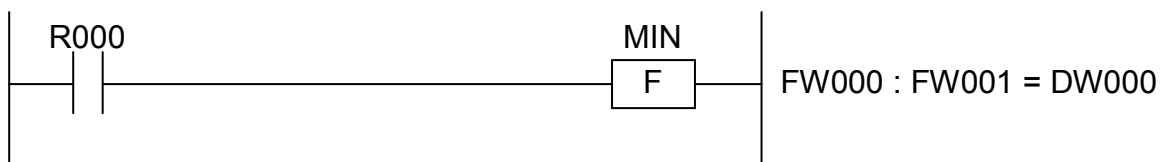
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	√	√	√	√	√
D	√	√	√	√	√	√	√
R	√	-	√	-	√	-	√

√: May be specified.

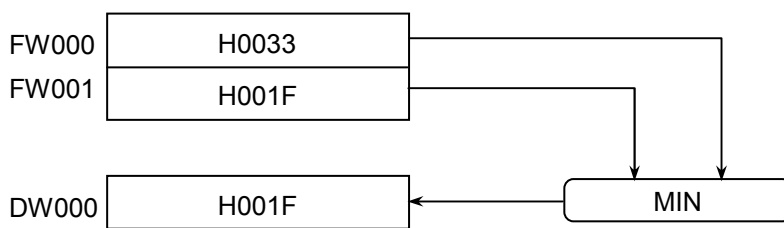
-: May not be specified.

The types of S, D, and R must be the same (i.e., either word, long-word, or floating). If the three are of different types, an input error will result.

(4) Example program



In this example, if the contact R000 (input condition) is closed (ON), the MIN instruction compares the contents of FW000 and FW001 and stores the smaller value in DW000.



(5) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.



# CLR CLEAR

---

## (1) Input format

XCLR
YCLR
GCLR
RCLR
KCLR
TCLR
UCLR
CCLR
VCLR
ECLR
FCLR

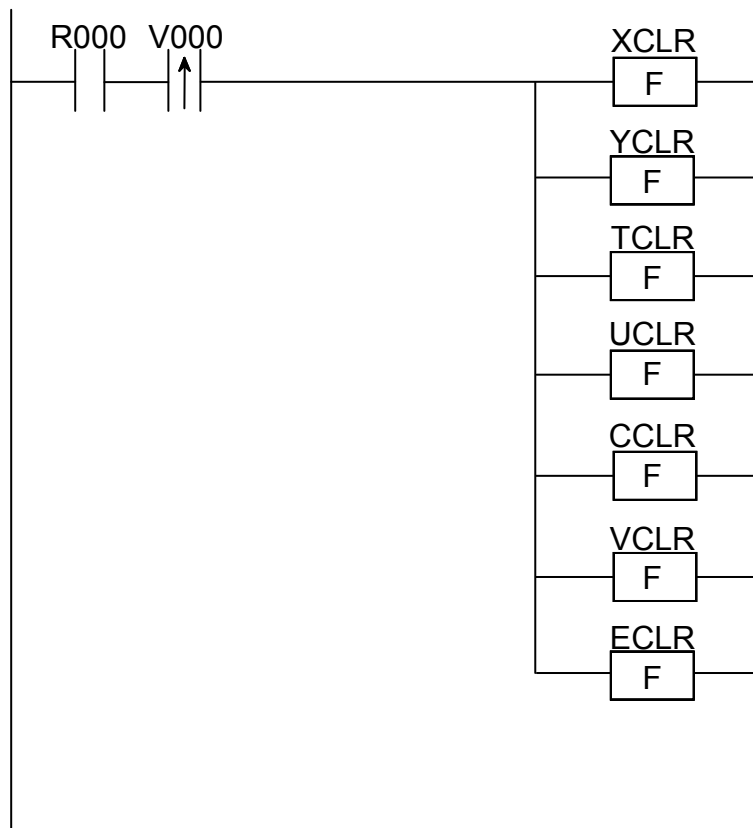
Note: All the above variations of the CLR instruction require no parameters.

## (2) Function

Any of the following CLR variations clears a predetermined I/O area:

- ① XCLR: Clear the X-area (external input).
- ② YCLR: Clear the Y-area (external output).
- ③ GCLR: Clear the G-area (global link registers).
- ④ RCLR: Clear the R-area (internal registers).
- ⑤ KCLR: Clear the K-area (keep relays).
- ⑥ TCLR: Clear the T-area (ON-delay timers and counts).
- ⑦ UCLR: Clear the U-area (one-shot timers and counts).
- ⑧ CCLR: Clear the C-area (up-down counters and counts).
- ⑨ VCLR: Clear the V-area (edge contacts).
- ⑩ ECLR: Clear the E-area (event registers).
- ⑪ FCLR: Clear the operation result flags (X, E, P, N, Z, and V).

## (3) Example program



In this example, if the contact R000 (input condition) makes a transition from OFF to ON state, the XCLR, YCLR, TCLR, UCLR, CCLR, VCLR, and ECLR instructions clear the X-, Y-, T-, U-, C-, V-, and E-areas only once.

## (4) Error handling

All variations of the CLR instruction always end their execution normally.

# JT JUMP IF TRUE

## (1) Input format

JT LAB

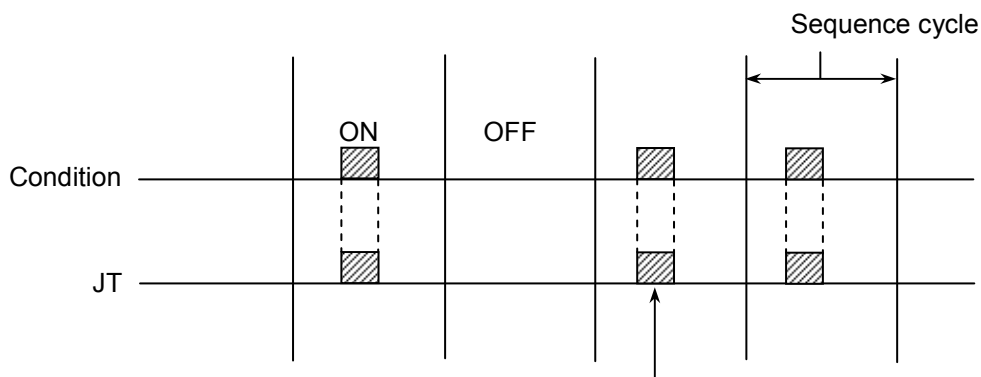
where:

LAB: Is the label name given to the destination of a jump to be made.

Note: At least one space must be inserted between the function name and parameter.

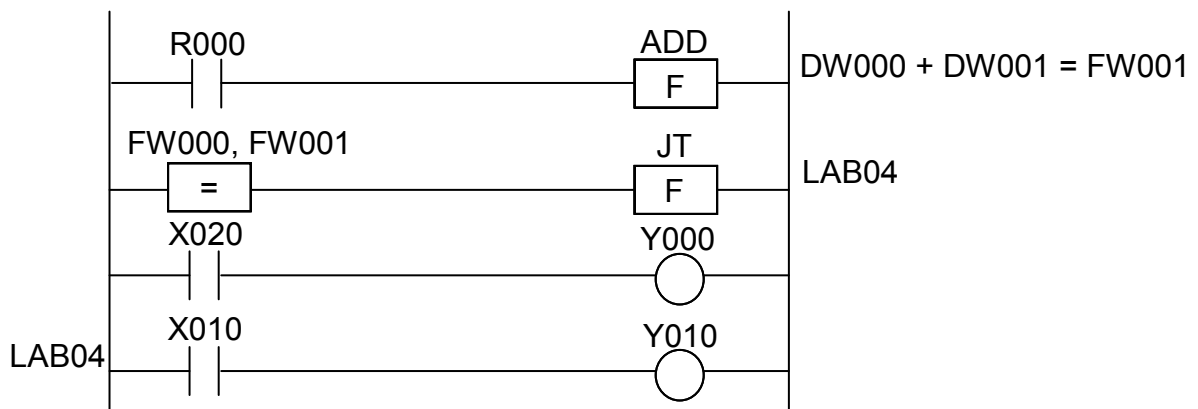
## (2) Function

The JT instruction jumps to a specified label if a given condition is true; otherwise, it proceeds to the next step in normal sequence.



A jump is made every time the condition is true (ON).

## (3) Example program



In this example, if the contact R000 (input condition) is closed (ON), the ADD instruction adds the contents of DW000 and DW001 together and stores the result in FW001. Then, if the contents of FW000 and FW001 are equal, the JT instruction jumps to the label LAB04 and, if the contact X010 therein is closed (ON), the coil Y010 becomes ON. On the other hand, if the contents of FW000 and FW001 are not equal, the JT instruction proceeds to the next step without jumping to the label LAB04 and, if the contact X020 is closed (ON), the coil Y000 becomes ON. Then, the step with LAB04 and the subsequent steps, if any, are executed.

(4) Error handling

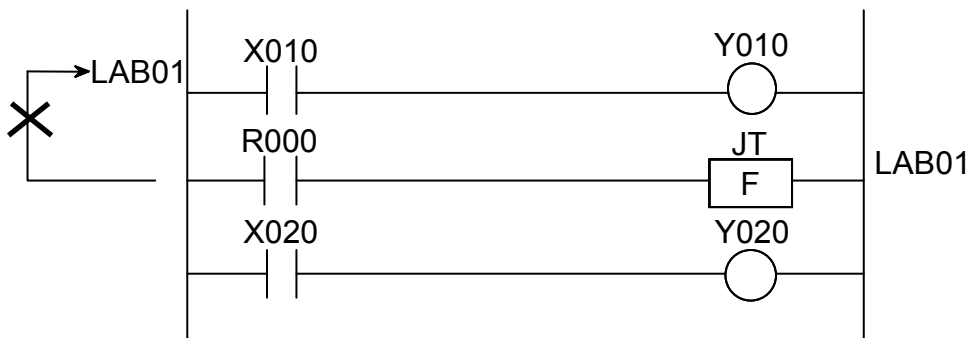
- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

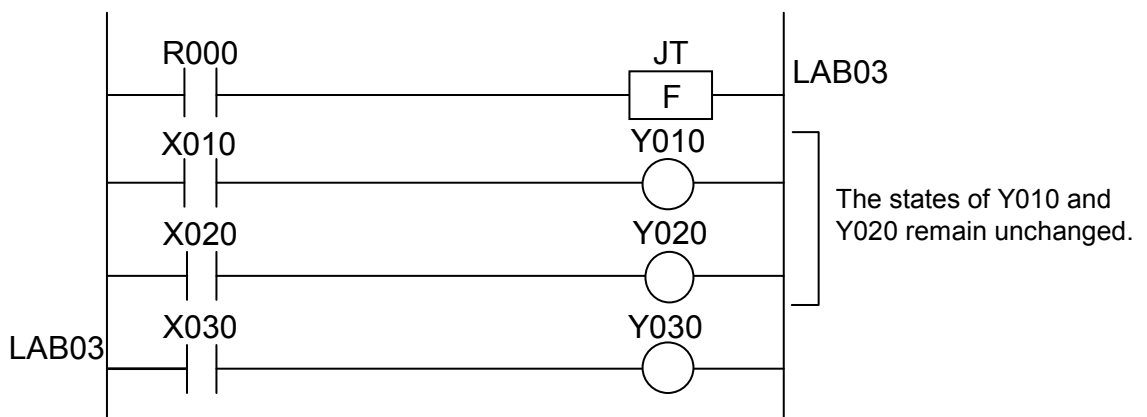
All the above flags remain unchanged.

Notes:

- No jump instruction can jump to any label that appears before the step in which the jump instruction is being executed in normal sequence. (This restriction is imposed to prevent any endless loop in the ladder program.)



- The coil(s) that are skipped by a jump instruction in normal sequence stay in the same states as before the execution of the jump instruction.



# JMP UNCONDITIONAL JUMP

## (1) Input format

```
JMP LAB
```

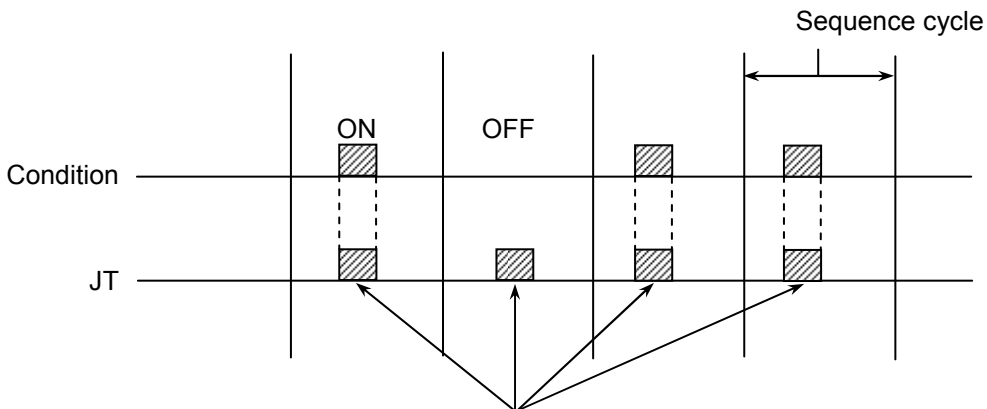
where:

LAB: Is the label name given to the destination of a jump to be made.

Note: At least one space must be inserted between the function name and parameter.

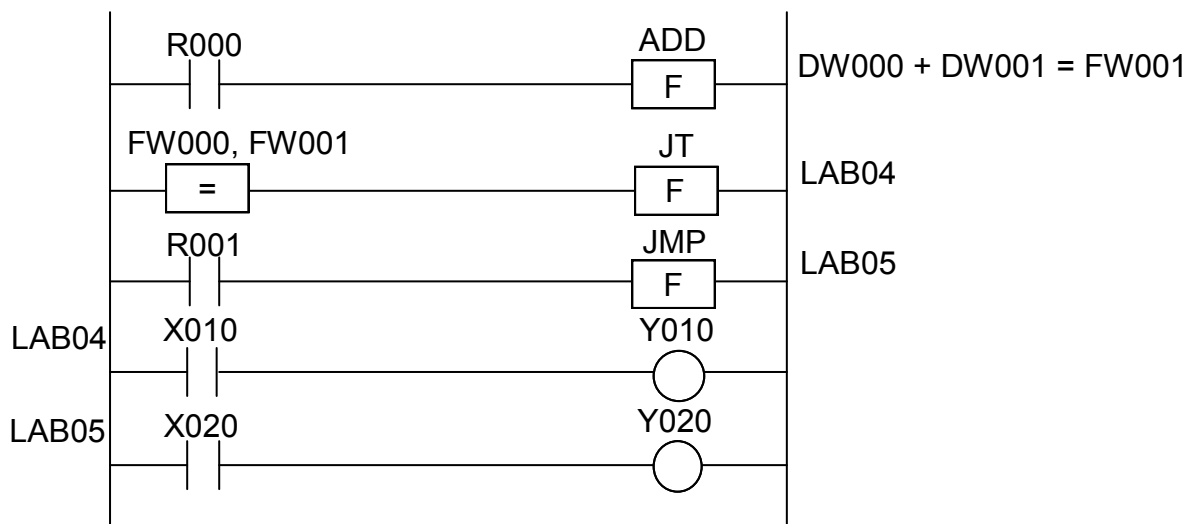
## (2) Function

The JMP instruction jumps to a specified label unconditionally.



A jump is always made regardless of the condition.

## (3) Example program



In this example, if the contact R000 (input condition) is closed (ON), the ADD instruction adds the contents of DW000 and DW001 together and stores the result in FW001. Then, if the contents of FW000 and FW001 are equal, the JT instruction jumps to the label LAB04 and, if the contact X010 therein is closed (ON), the coil Y010 becomes ON. Then, if the contact X020 is closed (ON), the coil Y020 becomes ON. On the other hand, if the contents of FW000 and FW001 are not equal, the JMP instruction unconditionally jumps to the label LAB05 regardless of the ON/OFF status of R001. Then, if the contact X020 is closed (ON), the coil Y020 becomes ON.

(4) Error handling

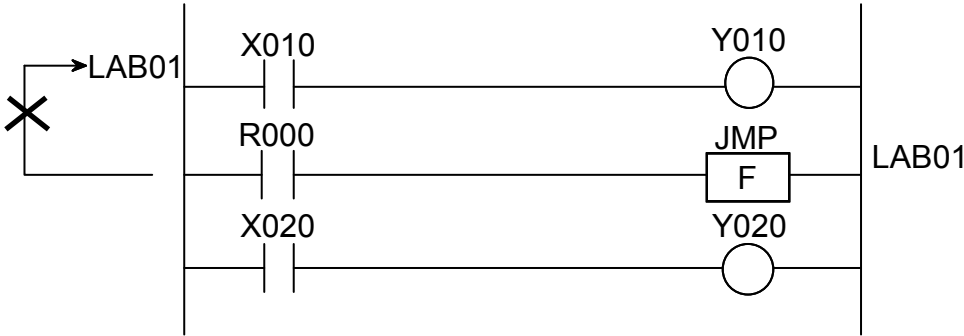
- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

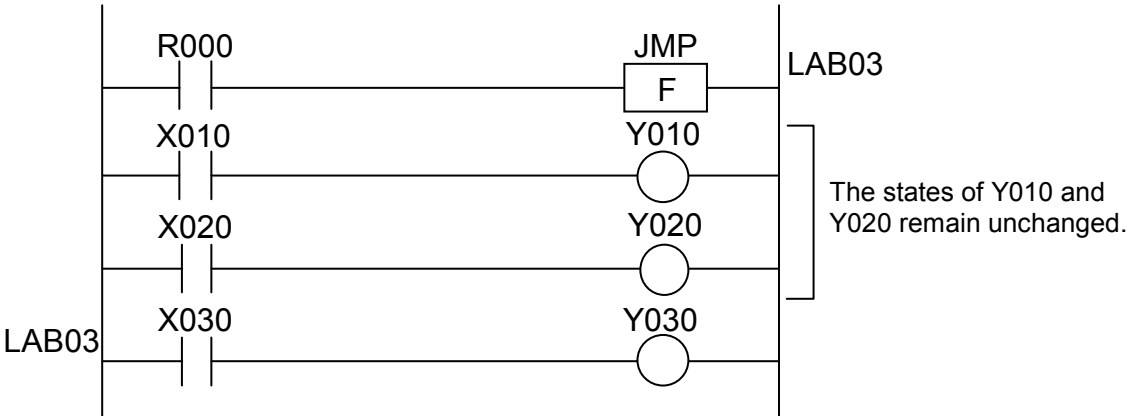
All the above flags remain unchanged.

Notes:

- No jump instruction can jump to any label that appears before the step in which the jump instruction is being executed in normal sequence. (This restriction is imposed to prevent any endless loop in the ladder program.)



- The coil(s) that are skipped by a jump instruction in normal sequence stay in the same states as before the execution of the jump instruction.



# JSE CONDITIONAL JUMP TO SEND

## (1) Input format

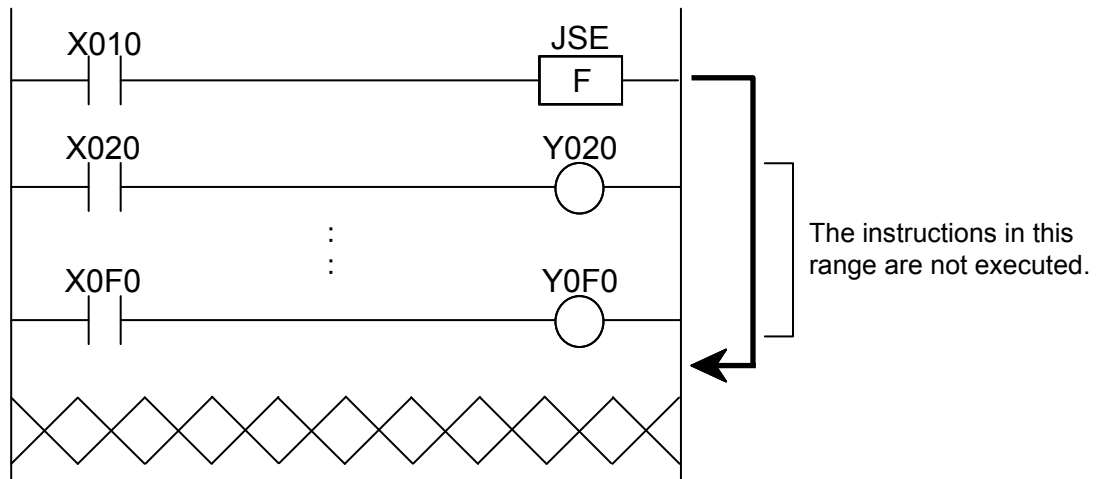


## (2) Function

The JSE instruction jumps to the end of the currently running N-coil program, or SEND (\*), if a given condition is true (ON).

(\*) The symbol SEND is an abbreviation of SequenceEND and denotes the end of an N-coil program.

## (3) Example program



In this example, if the contact X010 (input condition) is closed (ON), the JSE instruction jumps to the SEND.

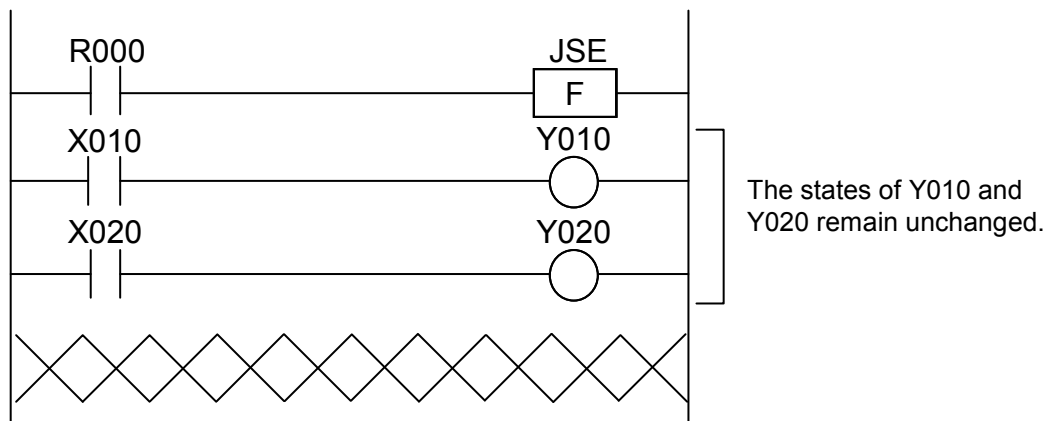
(4) Error handling

- Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

Note: The coil(s) that are skipped by a jump instruction in normal sequence stay in the same states as before the execution of the jump instruction.





### 2.7 Ethernet Communication Instructions

#### 2.7.1 Functional overview

To perform TCP and UDP communications in ladder programs, use system extension arithmetic functions for Ethernet communication.

The ladder chart system makes the following interface available as the system extension arithmetic functions for Ethernet communication.

Instruction	Function
TOP	Opens a TCP connection (client).
TPOP	Opens a TCP connection (server).
TCLO	Closes a TCP connection.
TRCV	TCP reception
TSND	TCP transmission
UOP	Opens UDP.
UCLO	Closes UDP.
URCV	UDP reception
USND	UDP transmission

The table below shows the specifications of communications performed by the system extension arithmetic functions.

Item	Specification	Remarks
The maximum number of sockets usable at a time	CMU: 16	This number is the total number of sockets that can be used at a time for TCP/UDP transmissions and/or receptions.
	ET.NET (main module): 16	
	ET.NET (submodule): 16	
	OPTET (any of modules 0 thru 3; sockets shared by them)	
Transmission/reception data size	For TCP communications: 0 to 4096 bytes	
	For UDP communications: 0 to 1472 bytes	
Port number	1 to 65535	Users are recommended to use port numbers in the range 10000 to 59999. The port numbers 60000 onwards are reserved for the system.

The use of Ethernet communication arithmetic functions requires the hardware module(s) of a specific version(s) listed below, depending on how the LPU unit is configured.

Configuration	Required module
CMU only	CMU (LQP520): 03-01 or later
ET.NET only	LPU (LQP510): 02-02 or later
	ET.NET (LQE720): 01-00 or later
OPTET only	LPU (LQP510): 02-02 or later
	OPTET (LQE710): 01-00 or later
Both CMU and ET.NET	LPU (LQP510): 02-02 or later
	CMU (LQP520): 04-00 or later
	ET.NET (LQE720): 01-00 or later
Both CMU and OPTET	LPU (LQP510): 02-02 or later
	CMU (LQP520): 06-00 or later
	OPTET (LQE710): 01-00 or later

These Ver.-Rev. numbers are those of the individual modules' microprograms which are displayed in the "Module List" by the S10V base system.

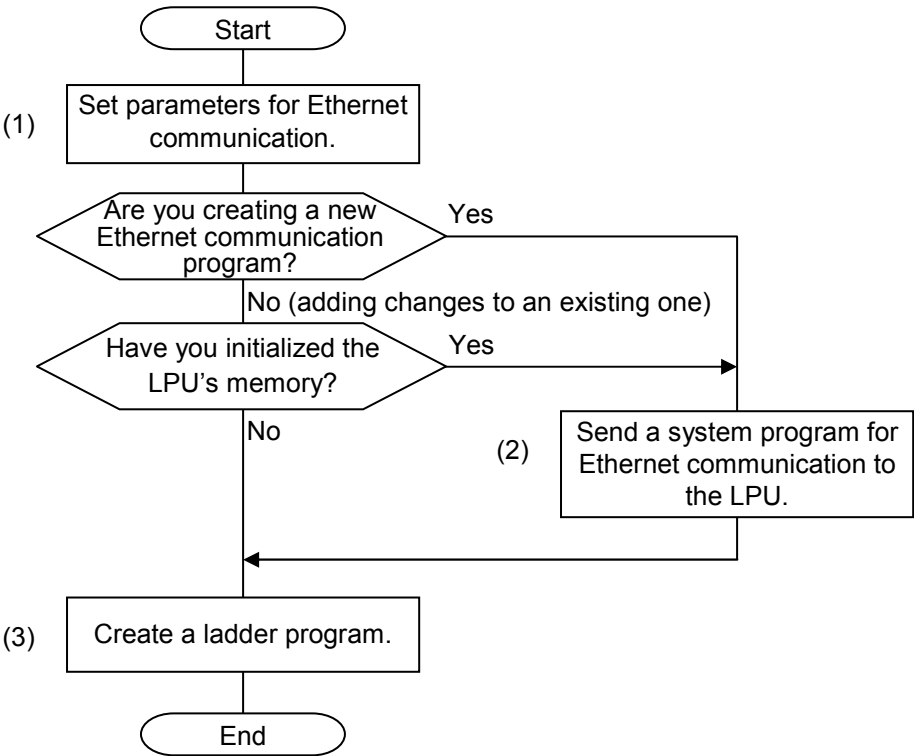
## 2 ARITHMETIC FUNCTIONS

Each time a system extension arithmetic function for Ethernet communication is executed, its execution result is flagged in one of the system registers S9C0 through S9EF and S690 through S6AF according to the management number used, which is predefined in one-to-one correspondence with an available socket. When the execution of such an arithmetic function is terminated normally or abnormally, the result is flagged by setting the system register associated with the management number to 0 or 1, respectively.

Register type		Management number	Remarks
Word	Bit		
SW9C0	S9C0	1	Provided for CMU Ethernet communications.
	S9C1	2	
	⋮	⋮	
	S9CE	15	
	S9CF	16	
SW9D0	S9D0	17	Provided for ET.NET (main module) Ethernet communications.
	S9D1	18	
	⋮	⋮	
	S9DE	31	
	S9DF	32	
SW9E0	S9E0	33	Provided for ET.NET (submodule) Ethernet communications.
	S9E1	34	
	⋮	⋮	
	S9EE	47	
	S9EF	48	
SW690	S690	49	Provided for OPTET Ethernet communications.
	S691	50	
	⋮	⋮	
	S69E	63	
	S69F	64	
SW6A0	S6A0	65	Provided for OPTET Ethernet communications.
	S6A1	66	
	⋮	⋮	
	S6AE	79	
	S6AF	80	

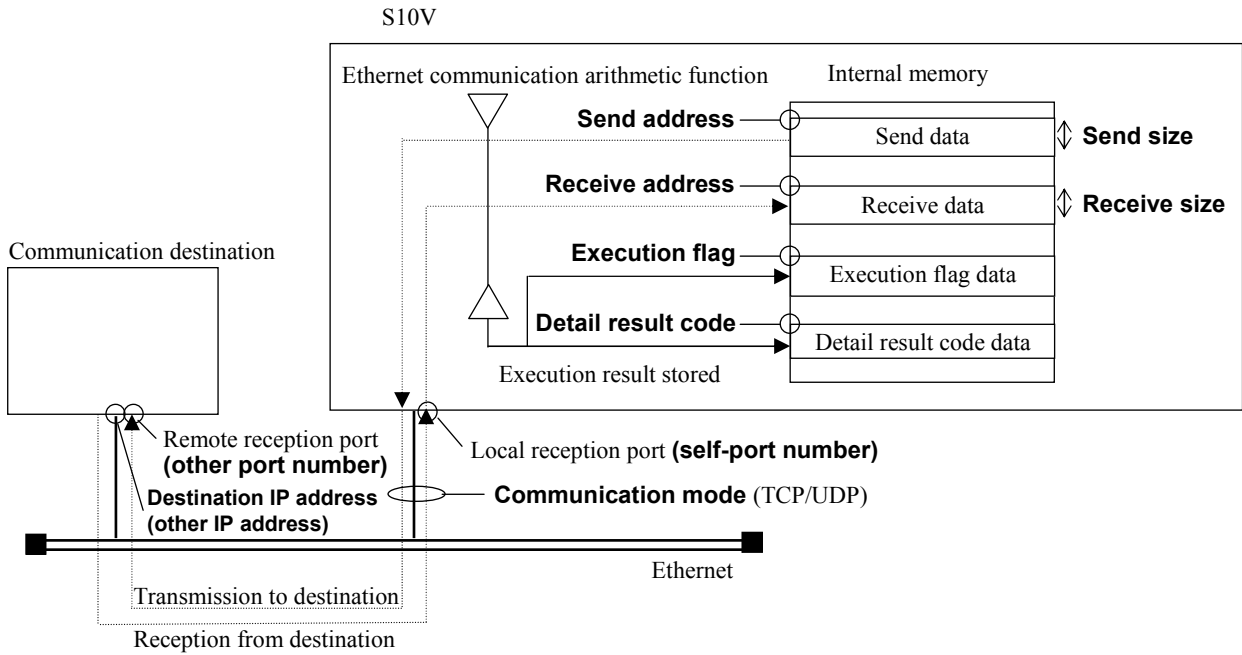
2.7.2 Usage

System extension arithmetic functions for Ethernet communication perform processing according to the parameters supplied by the user in the [Set Ethernet Communication] window of the ladder chart system. Therefore, users have to set all necessary parameters in the [Set Ethernet Communication] window before executing their ladder program. The procedure is as flowcharted below.



## 2 ARITHMETIC FUNCTIONS

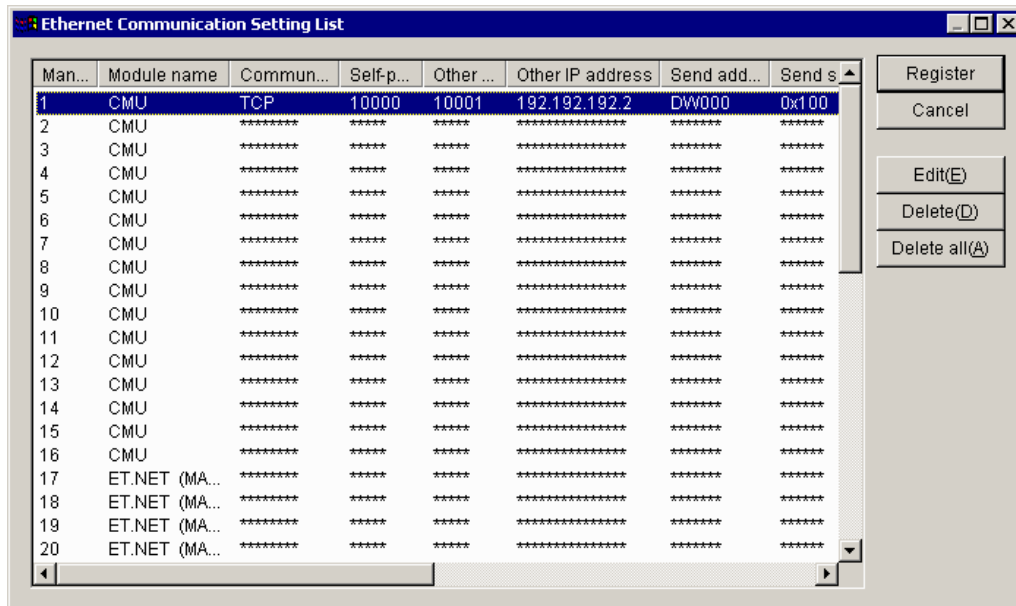
When you set parameters in the window, use the figure given below as a reference. The items shown in boldface are those set in the [Set Ethernet Communication] window. For details on the settings in the [Set Ethernet Communication] window, see the description under the heading “(1) Setting Ethernet communication parameters” below.



## (1) Setting Ethernet communication parameters

To set parameters in the [Set Ethernet Communication] window, select [Utility] – [Set Ethernet Communication] – [Set Parameter] from the ladder chart system's menu. Then, the following [Ethernet Communication Setting List] window appears on screen.

<[Ethernet Communication Setting List] window>



In this window, select the desired parameter information line and click the **Edit** button. Alternatively, double-click the desired parameter information line. Then, the [Set Ethernet Communication] window for the selected line will appear.

For details on the settings made in the window, refer to the “SOFTWARE MANUAL OPERATION S10V LADDER CHART For Windows® (Manual number SVE-3-131).”

## 2 ARITHMETIC FUNCTIONS

---

<[Set Ethernet Communication] window>

Management No. : 1

Module name : CMU

Communication mode(C) : TCP

Connection information

Self-port No.(M) : 10000

Other port No.(O) : 10001

Other IP address(I) : 192 . 192 . 192 . 2

Send/Receive area

Send address(S) : DW000 ~ DW07F

Send size(D) : / 100 Byte

Receive address(R) : FW000 ~ FW0FF

Receive size(Z) : / 200 Byte

Receive timeout(T) : 20 (\*100ms)

Result storing area

Execution flag(P) : R000

Details result code(E) : LWL0000

Socket disconnection mode(K) : Waiting for non-sent data sending

OK

Cancel

Each of the parameters shown above are described below.

<Item>

Management No.:

Displays management numbers specified on [Ethernet Communication Setting List] window.

Module name:

Displays the module for communication specified on [Ethernet Communication Setting List] window.

The module name is fixed according to the management number and the module shown below will be displayed.

When choosing an OPTET module before use, whose management number is in the range 49 to 80, specify its module number.

Management No.	Module name
1 to 16	CMU
17 to 32	ET.NET (main)
33 to 48	ET.NET (sub)
49 to 80	OPTET (Module 0 to 3)

Communication mode:

While in the combo box, select “TCP” or “UDP”. It is “TCP” by default.

Self-port No.:

Specify a port number for communication in decimal. (The specification range is from 1 to 65535.) It is blank by default. (Using a number between 10000 and 59999 is recommended. The system reserves numbers for 60000 and above.)

Other port No.:

Specify the port number of the destination in decimal. (The specification range is between 1 and 65535.) It is blank by default. (Using a number between 10000 and 59999 is recommended. The system reserves a number for 60000 and above.)

Other IP address:

Specify the IP address of the destination. It is blank by default. To broadcast data by UDP transmission, specify the node address as 255, as in 255.255.255.255.

Send address:

Specify the starting address of sent data in word form (registers for longword and floating only are in longword and floating forms) of PI/O. The system does not allow you to specify a bit-type register, specify an area unassigned as a PI/O, or span two or more registers. It is blank by default. The send address and send size are used to calculate the final address of sent data and display it.



## 2 ARITHMETIC FUNCTIONS

---

### Send size:

Specify a send size for data in a hexadecimal number. It is blank by default. The unit is the byte. For each communication type, the system allows you to specify either of the following sizes:

TCP: 0x0 to 0x1000 (0 to 4096)

UDP: 0x0 to 0x5C0 (0 to 1472)

### Receive address:

Specify the starting address of the area for storing received data in word form (registers for longword and floating only are in longword and floating forms) of PI/O. The system does not allow you to specify a bit-type register, specify an area unassigned as PI/O, or span two or more registers. The receive address and the receive size are used to calculate the final address of received data and display it.

### Receive size:

Specify a receive size for data in a hexadecimal number. It is blank by default and the units are bytes. For each communication type, the system allows you to specify either of the following sizes:

TCP: 0x0 to 0x1000 (0 to 4096)

UDP: 0x0 to 0x5C0 (0 to 1472)

### Receive timeout:

Set a wait time for received data to arrive in case data cannot be received when a reception instruction is issued. Specify a range between 0 and 100 (0 and 10 seconds) in increments of 100 ms. (0 means no timeout.) It is set to 10 (1 second) by default. Set a timeout setting. If a reception instruction causes a reception timeout, the reception instruction will cause an error with no reception data (EWOULDBLOCK).

### Execution flag:

Specify with a bit-type register that specifies whether an applied instruction for Ethernet communication is being processed. It is blank by default.

### Details result code:

Specify with a long-type register an area for storing a detail result code for the execution result of an applied instruction for Ethernet communication. It is blank by default.

Socket disconnection mode:

Can only be specified when the communication mode is “TCP”. Select “Waiting for non-sent data sending” or “Non-sent data destruction” from the combo box. It is “Waiting for non-sent data sending” by default. Here are the options and their meanings:

Waiting for non-sent data sending: If data has not yet been sent, the system will wait until the data flows. Any unread data will be discarded.

Non-sent data destruction: If data has not yet been sent, the system will disconnect the channel and relieve the socket without waiting for the data to flow. In that case, the TCP of the destination host will receive an RST. Since the disconnection takes place differently from the way it usually occurs, be careful as to how the system functions (the method of reporting when an RST is received by the UP) when the destination host receives an RST. Any unread received data will be discarded.

The following list shows the registers, which can be specified on the [Set Ethernet Communication] window.

<Setting Registers>

(1/2)

No.	Item	Symbol	Send address	Receive address	Execution flag	Detail result code
1	External input	X	√	√	√	√
2	External output	Y	√	√	√	√
3	Internal register	R	√	√	√	√
4	Keep relay	K	√	√	√	√
5	On-delay timer	T	√	√	√	√
6	One-shot timer	U	√	√	√	√
7	Up/down counter	C	√	√	√	√
8	Global link register	G	√	√	√	√
9	Nesting coil	N	√	√	√	√
10	Process register	P	√	√	√	√
11	Event register	E	√	√	√	√
12	Edge contact	V	√	√	√	√
13	Z-register	Z	√	√	√	√
14	System register	S	√	√	√	√
15	Data register	DW	√	√	—	√

√: Enable to be specified

—: Disable to be specified

## 2 ARITHMETIC FUNCTIONS

(2/2)

No.	Item	Symbol	Send address	Receive address	Execution flag	Detail result code
16	Work register	FW	√	√	–	√
17	Internal register	M	√	√	√	√
18	Internal register (Longword)	BD	–	–	–	–
19	For high speed RI/O input	I	√	√	–	√
20	For high speed RI/O output	O	√	√	–	√
21	Register for which HI-FLOW and Ladder share data.	J	√	√	√	√
22		Q	√	√	√	√
23	Work register	LB	√	√	√	√
24	Work register for word only	LW	√	√	–	√
25	Work register for longword only	LL	√	√	–	√
26	Work register for single-precision floating point only	LF	√	√	–	√
27	Work register for word only (Power failure retention)	LX	√	√	–	√
28	Work register for longword only (Power failure retention)	LM	√	√	–	√
29	Work register for single-precision floating point only (Power failure retention)	LG	√	√	–	√
30	Work register for ladder converter only	LR	√	√	√	√
31	Work register for ladder converter only (Edge contact)	LV	√	√	√	√

√: Enable to be specified

–: Disable to be specified

The table below is a list of all detail result codes returned by the system extension arithmetic functions for Ethernet communication.

<Detail result codes>

(1/2)

Value	Meaning	Required remedial action
0	Normal end (for TOP, TPOP, TCLO, UOP, UCLO)	—
0 to 4096	Normal end (send/receive data size; for TRCV, TSND, URCV, USND)	—
0x80000005 (EIO)	A serious error is detected in the adapter (device).	Consult the description of a remedial action given as part of the error log information. (*1)
0x8000000D (EACCES)	A broadcast address is specified as the destination IP address.	The Ethernet communication settings contain an error. Review the settings.
0x80000016 (EINVAL)	A disconnected socket is specified, or the receive buffer length is a negative value.	The Ethernet communication settings contain an error. Review the settings.
0x800000DA (EMSGSIZE)	A given send data length is out of the permitted range.	The Ethernet communication settings contain an error. Review the settings.
0x800000E2 (EADDRINUSE)	The port number is already in use by another socket.	Review the port number used.
0x800000E3 (EADDRNOTAVAIL)	A specified port number and IP address contain an error.	The Ethernet communication settings contain an error. Review the settings.
0x800000E4 (ENETDOWN)	The device is not initialized yet or is stopped.	Consult the description of a remedial action given as part of the error log information. (*1)
0x800000E5 (ENETUNREACH)	No routing information is present for a given destination IP address.	Review the routing information settings for the CMU or ET.NET module. (*2)
0x800000E7 (ECONNABORTED)	The connection is terminated abruptly.	<ul style="list-style-type: none"> <li>• Check the cable wiring.</li> <li>• Review the program running in the connection destination's host.</li> </ul>
0x800000E8 (ECONNRESET)	Connection is reset by the TCP of the connection destination's host.	Review the program running in the connection destination's host.
0x800000E9 (ENOBUFS)	Memory securing has failed.	Consult the description of a remedial action given as part of the error log information. (*1)
0x800000EB (ENOTCONN)	An attempt is made to send data to a socket with which a connection is not established yet.	Execution of TOP or TPOP has failed. Review the program.
0x800000EC (ESHUTDOWN)	The socket is released by some other task.	Check if the same management number as used in the ladder program is used in a HI-FLOW program.
0x800000EE (ETIMEDOUT)	A connection request is timed out.	<ul style="list-style-type: none"> <li>• Check the cable wiring.</li> <li>• Review the program running in the connection destination's host.</li> </ul>
0x800000EF (ECONNREFUSED)	The connection destination's socket is missing (a server task is not bound yet).	Review the program running in the connection destination's host.
0x800000F6 (EWOULDBLOCK)	No data is received. Data cannot be transmitted because TCP's transmission window is full.	Review the program.
0x800000F9 (ENSOCK)	The maximum number of sockets that can be open at a time is exceeded.	Review the program so that no more than 16 sockets will be open at a time for each module.
0x80000516 (EBADF)	The socket is not opened yet.	None of TOP, TPOP, and UOP is executed, or although one is initiated, its execution has failed. Review the program.

(\*1) For information on how to obtain the error log information, refer to the "USER'S MANUAL BASIC MODULE (Manual number SVE-1-100)."

(\*2) Routing information settings are made with the setting tool for each module.

## 2 ARITHMETIC FUNCTIONS

(2/2)

Value	Meaning	Remedial action required
0xFFFFFFFFB	The Ethernet module is down.	By resetting the LPU module, restart the ET.NET module. Then, if the same error recurs, the ET.NET module is probably damaged. Replace the module.
0xFFFFFFFFC	An Ethernet module is probably not installed yet.	Check that a CMU or ET.NET module is installed.
0xFFFFFFFFD	Initiation of a task has failed.	Check that the CMU module's version supports applied instructions for Ethernet communication.
0xFFFFFFFFE	Communication identifier error detected (the management number is already in use).	Check if another ladder program and a HI-FLOW program use the Ethernet communication settings managed under the same management number.
0xFFFFFFFFF	Inconsistency detected with the mode in use (inconsistency between parameter-specified transmission method and communication mode).	Check that the transmission method specified as part of the Ethernet communication settings is the same as the communication mode used in the ladder and the HI-FLOW program.

- Error type

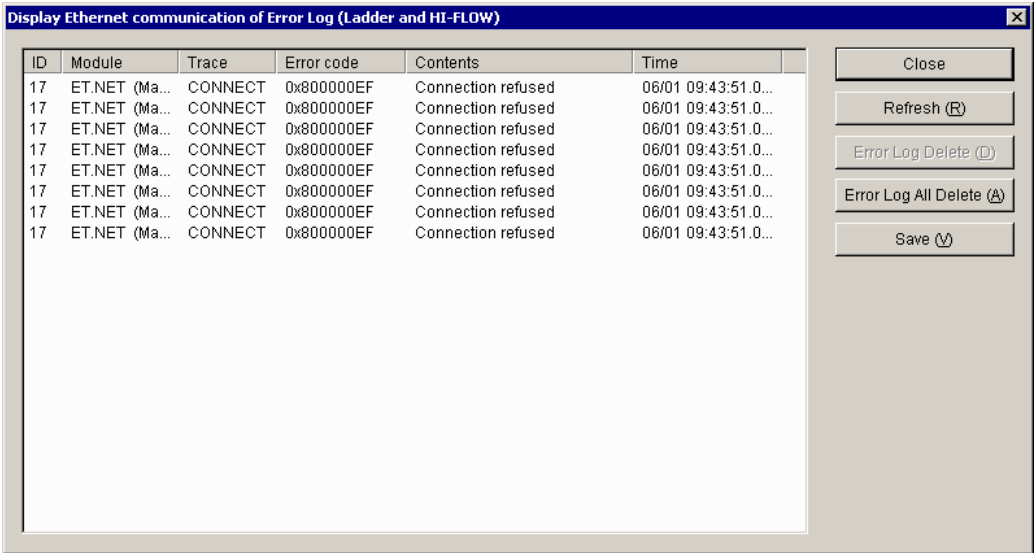
0x8XXXXXXXX: CPMS socket macro error (this code is created by adding the actual CPMS socket macro error code and the value 0x80000000 together).

0xFXXXXXXXX: System program or task error.

The information provided below is concerning the error tracing that occurs during Ethernet communication.

A total of eight cases of error trace information are collected for each management number. These cases of error trace information can be viewed in the base system's [Display Ethernet communication of Error Log (Ladder and HI-FLOW)] window. For information on how to identify errors from the error trace information displayed in the [Display Ethernet communication of Error Log (Ladder and HI-FLOW)] window, refer to the "USER'S MANUAL BASIC MODULE. (Manual number SVE-1-100)."

<[Display Ethernet communication of Error Log (Ladder and HI-FLOW)] window>



## 2 ARITHMETIC FUNCTIONS

---

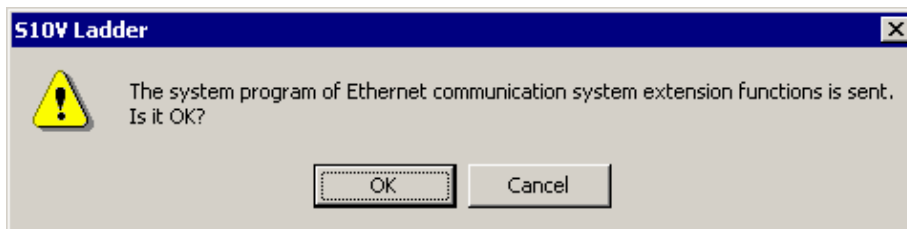
### (2) Sending an Ethernet communication system program to the LPU module

Send to the LPU module a system program that provides Ethernet communication facilities (system extension arithmetic functions). The system program providing Ethernet communication facilities which is available from the ladder chart system must be transmitted to the LPU module before use if you want to use them.

The procedure is as described below, which can be used only during online operation of the system.

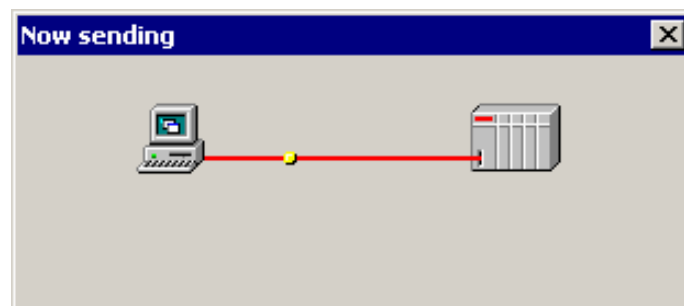
To send the system program to the LPU module, choose [Set Ethernet Communication] – [Send System Program] from the [Utility] menu. Then, the following System Program Send Confirmation window appears:

<System Program Send Confirmation window>



If you want to send it to the LPU, click the  button; otherwise, click the  button. When the  button is clicked, the following [Now sending] window appears:

<[Now sending] window>



At the end of the transmission, the [Now sending] window will disappear from the screen.

<Notice>

- A ladder program can be created even if you do not send the system program (Equal to mount Ethernet communication system extension functions). Ethernet communication system extension functions are used in the ladder program, provided, however, that checking whether Ethernet communication system extension functions are mounted during ladder program transit. If not, the error message dialog box will be displayed and ladder program will not be sent.

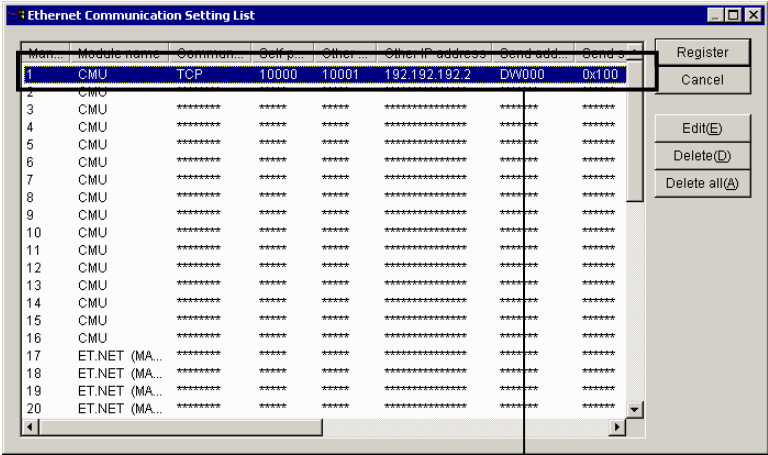
Error message: XXXX of a system extension function is not mounted. (XXXX is the function name of Ethernet communication system extension functions.)

- When initializing the LPU module memory, the system program sent to LPU module memory will be cleared. Therefore, initializing the LPU module memory is required to resend the system program.

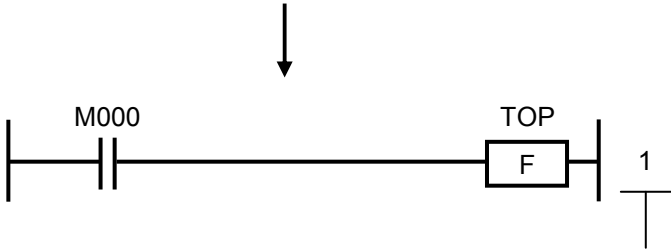
(3) Creating a ladder program

When you create a ladder program, choose the desired management number from among those that have been set in the [Set Ethernet Communication] window, and specify it as the parameter to an appropriate Ethernet communication system extension function. The system extension function is then able to perform processing according to the window-set information piece identified by that management number.

<Example>



The information piece identified by management number 1 is one that has been set in the [Set Ethernet Communication] window.



Specify management number 1 here.



## 2 ARITHMETIC FUNCTIONS

### 2.7.3 Details on the instructions

Information in this section is concerning all available Ethernet communication system extension function instructions and is organized as follows.

(1) Input format

Under this heading is shown the input format of each instruction.

(2) Function

Under this heading is provided a description of each instruction's function. The system registers mentioned in these descriptions are the system registers, S9C0 through S9FF, that are used to store execution results of the system extension function instructions.

(3) Data types

Under this heading are listed the types of data that can be specified as parameters to each instruction.

Example:

	This portion shows whether such registers as DW000 and such constants as H0001 may be used with the instruction or not.		This portion shows whether such registers as LLL0000 and such constants as H04231556 may be used with the instruction or not.		This portion shows whether such registers as LF0000 and such constants as 1.12E-002 may be used with the instruction or not.		
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	–	–	–	–	–

√: May be specified.

–: May not be specified.

If a register may be specified, this portion shows whether an index may be specified or not.

According to the above sample table, users can specify, as Source (S), addresses of such data as word, long-word, and floating, including index specifications, and can specify constants of those types.

Note: Bit I/O areas, such as R000 and Y1FF, are handled as word data in arithmetic functions. In these cases, only the LSB is valid and all the other bits are zero (0) in reading and invalid in writing. For details, see Section 2.3.2, "Handling of bit registers."

(4) Example program

Under this heading is shown a simple ladder program using each instruction and its operation.

(5) Error handling

Under this heading is described what processing will be done if an error occurs. The operation result flag(s) reflecting the error are also shown under this heading.

# TOP OPEN A TCP CONNECTION (CLIENT)

## (1) Input format

TOP S
-------

where:

S: (Source) is a communication identifier, which is one of the following module-specific management numbers:

Management No.	Module name
1 to 16	CMU
17 to 32	ET.NET (main module)
33 to 48	ET.NET (submodule)
49 to 80	OPTET (Module 0 to 3)

A maximum of 16 TCP and/or UDP communication processes can be executed simultaneously for each module other than the OPTET. Where OPTET modules are used for communication, a maximum total of 32 TCP and/or UDP communication processes can be executed simultaneously for the four OPTET modules.

## (2) Function

- Opening a TCP connection (client)

The TOP (TCP connection open [client]) instruction opens a socket and establishes a connection with a destination by using its other port number and other IP address, which have been set in the [Set Ethernet Communication] window. This instruction ends its execution even if the initiated process is not completed. The result of the process is reported by storing appropriate values in a given system register, [Execution flag], and [Detail result code], the latter two of which are parameters that have been set in the [Set Ethernet Communication] window.

Execution flag: Set to 1 if the initiated process is in progress; otherwise (i.e., it is completed), set to 0. When the initiated process is completed, its result is reported by setting the system register and [Detail result code] to appropriate values. To obtain the information, be sure to monitor the [Execution flag] constantly until the process is complete.

Detail result code: Set to an appropriate value at the end of the initiated process, the value that indicates the result of execution of the process. When the [Execution flag] is set to 0, get information from this [Detail result code].

## (3) Data types

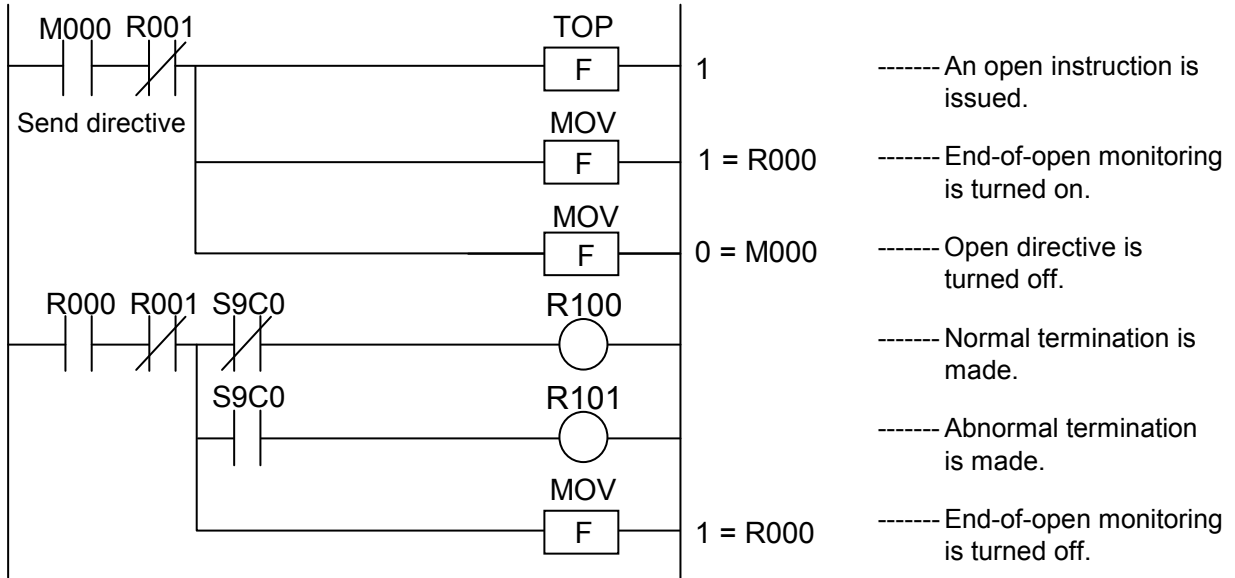
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	-	-	-	-	-

√: May be specified.

-: May not be specified.

(4) Example program

Opening using the management number 1 and execution flag R001:



(5) Error handling

If opening a TCP connection (client) succeeds, the system register and [Detail result code] are both set to 0. If it fails, the system register is set to 1 and the [Detail result code] is set to the error number (non-zero value).

Whether the initiated process has succeeded or not can be determined from the set value of the system register.

If the process has failed, get error cause information from the [Detail result code].

● Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# TPOP OPEN A TCP CONNECTION (SERVER)

## (1) Input format

TPOP S
--------

where:

S: (Source) is a communication identifier, which is one of the following module-specific management numbers:

Management No.	Module name
1 to 16	CMU
17 to 32	ET.NET (main module)
33 to 48	ET.NET (submodule)
49 to 80	OPTET (Module 0 to 3)

A maximum of 16 TCP and/or UDP communication processes can be executed simultaneously for each module other than the OPTET. Where OPTET modules are used for communication, a maximum total of 32 TCP and/or UDP communication processes can be executed simultaneously for the four OPTET modules.

## (2) Function

### ● Opening a TCP connection (server)

The TPOP (TCP connection open [server]) instruction opens a socket, accepts a connection request from a client by using the server's self-port number, which has been set in the [Set Ethernet Communication] window, and establishes a connection between the server and that client. This instruction ends its execution even if the initiated process is not completed. The result of the process is reported by storing appropriate values in a given system register, [Execution flag], and [Detail result code], the latter two of which are parameters that have been set in the [Set Ethernet Communication] window.

**Execution flag:** Set to 1 if the initiated process is in progress; otherwise (i.e., it is completed), set to 0. When the initiated process is completed, its result is reported by setting the system register and [Detail result code] to appropriate values. To obtain the information, be sure to monitor the [Execution flag] constantly until the process is complete.

**Detail result code:** Set to an appropriate value at the end of the initiated process, the value that indicates the result of execution of the process. When the [Execution flag] is set to 0, get information from this [Detail result code].

## (3) Data types

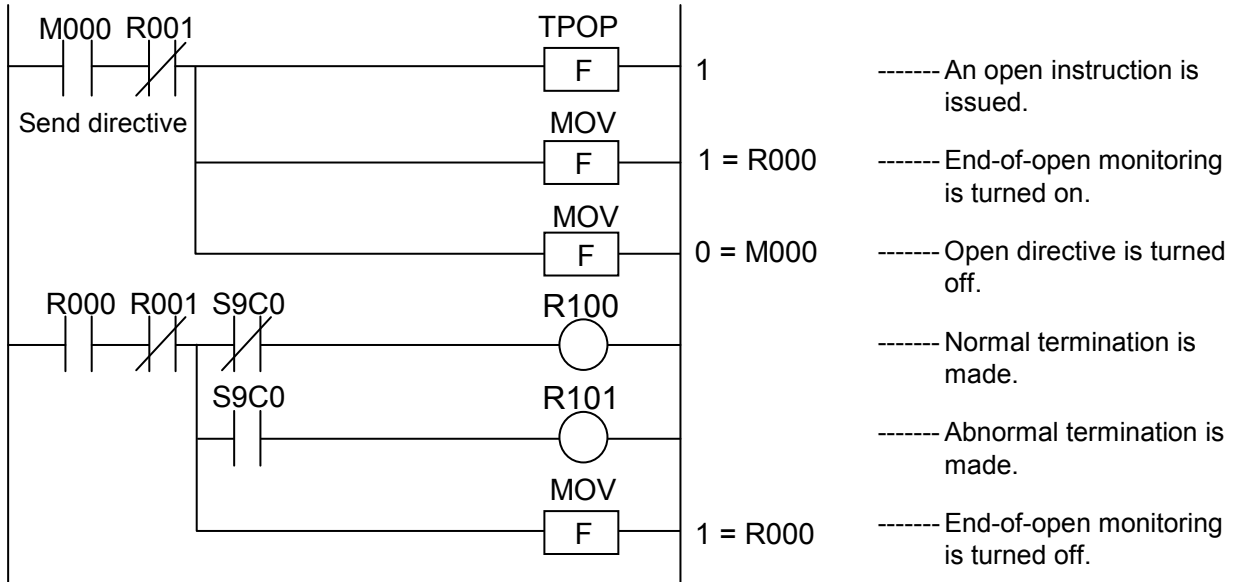
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	–	–	–	–	–

√: May be specified.

–: May not be specified.

(4) Example program

Opening using the management number 1 and execution flag R001:



(5) Error handling

If opening a TCP connection (server) succeeds, the system register and [Detail result code] are both set to 0. If it fails, the system register is set to 1 and the [Detail result code] is set to the error number (non-zero value).

Whether the initiated process has succeeded or not can be determined from the set value of the system register.

If the process has failed, get error cause information from the [Detail result code].

● Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

# TCLO CLOSE A TCP CONNECTION

## (1) Input format

TCLO S
--------

where:

S: (Source) is a communication identifier, which is one of the following module-specific management numbers:

Management No.	Module name
1 to 16	CMU
17 to 32	ET.NET (main module)
33 to 48	ET.NET (submodule)
49 to 80	OPTET (Module 0 to 3)

A maximum of 16 TCP and/or UDP communication processes can be executed simultaneously for each module other than the OPTET. Where OPTET modules are used for communication, a maximum total of 32 TCP and/or UDP communication processes can be executed simultaneously for the four OPTET modules.

## (2) Function

### ● Closing a TCP connection

The TCLO (TCP connection close) instruction logically disconnects a communication path by the method ([Socket disconnection mode]) that has been specified in the [Set Ethernet Communication] window and releases the socket. This instruction ends its execution even if the initiated process is not completed. The result of the process is reported by storing appropriate values in a given system register, [Execution flag], and [Detail result code], the latter two of which are parameters that have been set in the [Set Ethernet Communication] window.

**Execution flag:** Set to 1 if the initiated process is in progress; otherwise (i.e., it is completed), set to 0. When the initiated process is completed, its result is reported by setting the system register and [Detail result code] to appropriate values. To obtain the information, be sure to monitor the [Execution flag] constantly until the process is complete.

**Detail result code:** Set to an appropriate value at the end of the initiated process, the value that indicates the result of execution of the process. When the [Execution flag] is set to 0, get information from this [Detail result code].

## (3) Data types

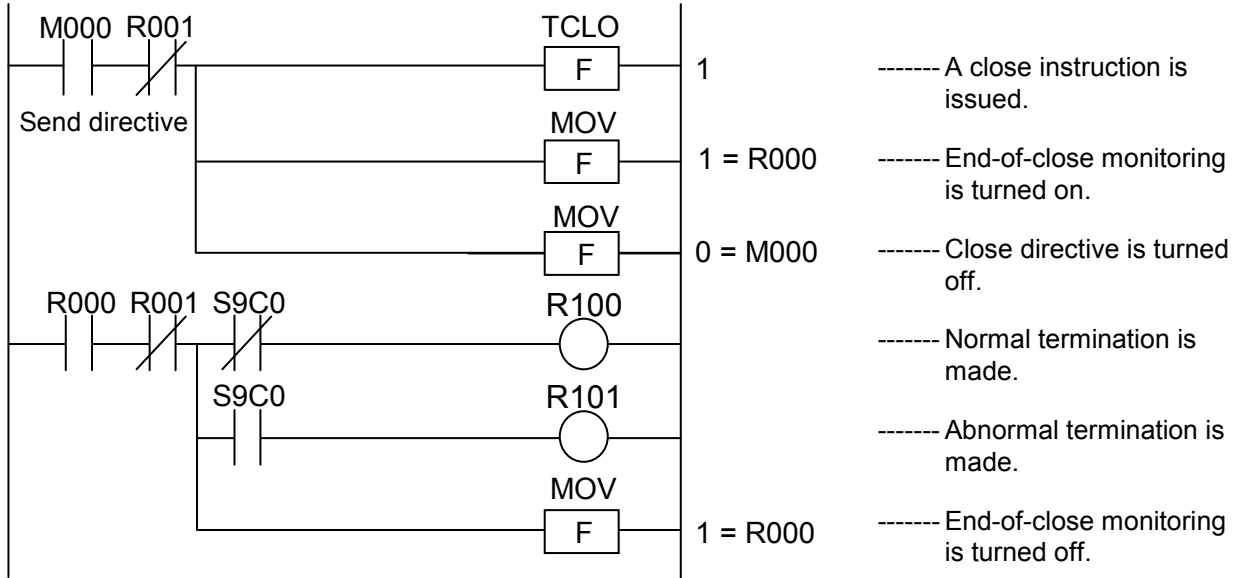
\	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	-	-	-	-	-

√: May be specified.

-: May not be specified.

(4) Example program

Closing using the management number 1 and execution flag R001:



(5) Error handling

If closing a TCP connection succeeds, the system register and [Detail result code] are both set to 0. If it fails, the system register is set to 1 and the [Detail result code] is set to the error number (non-zero value).

Whether the initiated process has succeeded or not can be determined from the set value of the system register.

If the process has failed, get error cause information from the [Detail result code].

● Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.



## (1) Input format

TRCV S
--------

where:

S: (Source) is a communication identifier, which is one of the following module-specific management numbers:

Management No.	Module name
1 to 16	CMU
17 to 32	ET.NET (main module)
33 to 48	ET.NET (submodule)
49 to 80	OPTET (Module 0 to 3)

A maximum of 16 TCP and/or UDP communication processes can be executed simultaneously for each module other than the OPTET. Where OPTET modules are used for communication, a maximum total of 32 TCP and/or UDP communication processes can be executed simultaneously for the four OPTET modules.

## (2) Function

### ● Reception by TCP

The TRCV (TCP receive) instruction receives as much message data as specified by the [Receive size] from a given socket and stores the received data in the area specified by the [Receive address], where the [Receive size] and [Receive address] are parameters that have been set in the [Set Ethernet Communication] window. This instruction ends its execution even if the initiated process is not completed. The result of the process is reported by storing appropriate values in a given system register, [Execution flag], and [Detail result code], the latter two of which are parameters that have been set in the [Set Ethernet Communication] window.

If there is no data to be received at the time of its issuance, this instruction monitors the reception process for the time period specified by the [Receive timeout], which has been set in the [Set Ethernet Communication] window. If the timeout has elapsed with no data received, the reception process is terminated with the [Detail result code] set to the value "EWOULDBLOCK". In this case, if you still want to receive data, re-issue this instruction.

**Execution flag:** Set to 1 if the initiated process is in progress; otherwise (i.e., it is completed), set to 0. When the initiated process is completed, its result is reported by setting the system register and [Detail result code] to appropriate values. To obtain the information, be sure to monitor the [Execution flag] constantly until the process is complete.

**Detail result code:** Set to a positive value if data is received. This positive value indicates the size of the received data. If the size of the received data is not equal to the [Receive size] value, one of the following takes place:

If [Receive size] > size of received data: All the received data is read in.

If [Receive size] < size of received data:

As much data as specified by the [Receive size] is read in from the received data.

The rest of the received data is retained and can be read in by issuing the TRCV instruction again.

If the reception process fails, the [Detail result code] is set to a negative value, which is the error code.

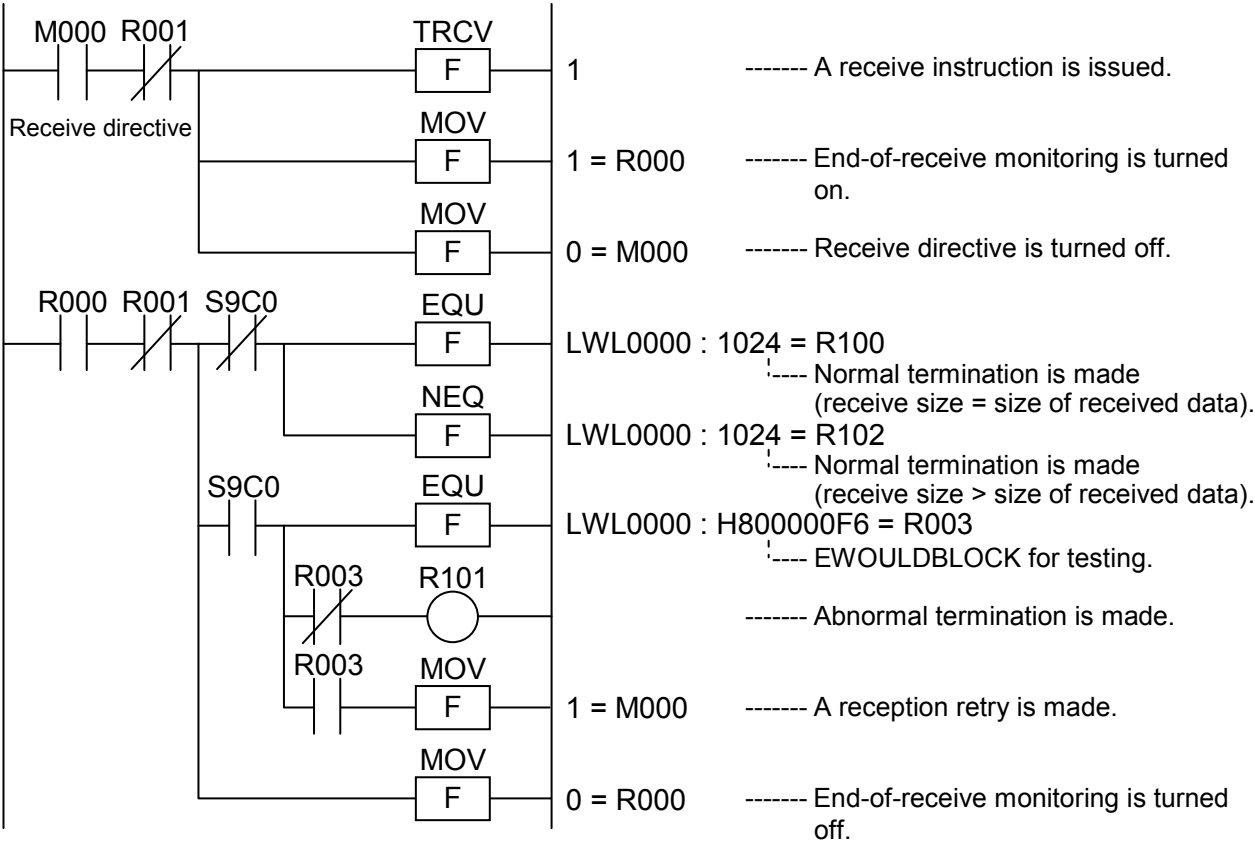
(3) Data types

	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	-	-	-	-	-

√: May be specified.  
 -: May not be specified.

(4) Example program

Receiving using the management number 1, execution flag R001, detail result code LWL0000, and a receive size of 1024 bytes, and performing a reception retry if no data is received:



### (5) Error handling

If the initiated reception process succeeds, the system register is set to 0 and the [Detail result code] is set to a value indicating the size of the received data. If it fails, the system register is set to 1 and the [Detail result code] is set to the error number (negative value). Whether the initiated process has succeeded or not can be determined from the set value of the system register.

If the process has failed, get error cause information from the [Detail result code]. If the [Detail result code] is EWOULDBLOCK, you can re-issue the TRCV instruction to receive data.

#### ● Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

**This Page Intentionally Left Blank**

## (1) Input format

TSND S
--------

where:

S: (Source) is a communication identifier, which is one of the following module-specific management numbers:

Management No.	Module name
1 to 16	CMU
17 to 32	ET.NET (main module)
33 to 48	ET.NET (submodule)
49 to 80	OPTET (Module 0 to 3)

A maximum of 16 TCP and/or UDP communication processes can be executed simultaneously for each module other than the OPTET. Where OPTET modules are used for communication, a maximum total of 32 TCP and/or UDP communication processes can be executed simultaneously for the four OPTET modules.

## (2) Function

### ● Transmission by TCP

The TSND (TCP send) instruction transmits as much send data as specified by the [Send size] from the area specified by the [Send address] to a given socket, where the [Send size] and [Send address] are parameters that have been set in the [Set Ethernet Communication] window. This instruction ends its execution even if the initiated process is not completed. The result of the process is reported by storing appropriate values in a given system register, [Execution flag], and [Detail result code], the latter two of which are parameters that have been set in the [Set Ethernet Communication] window.

**Execution flag:** Set to 1 if the initiated process is in progress; otherwise (i.e., it is completed), set to 0. When the initiated process is completed, its result is reported by setting the system register and [Detail result code] to appropriate values. To obtain the information, be sure to monitor the [Execution flag] constantly until the process is complete.

**Detail result code:** Set to an appropriate value at the end of the initiated process, the value that indicates the result of execution of the process. When the [Execution flag] is set to 0, get information from this [Detail result code].

## (3) Data types

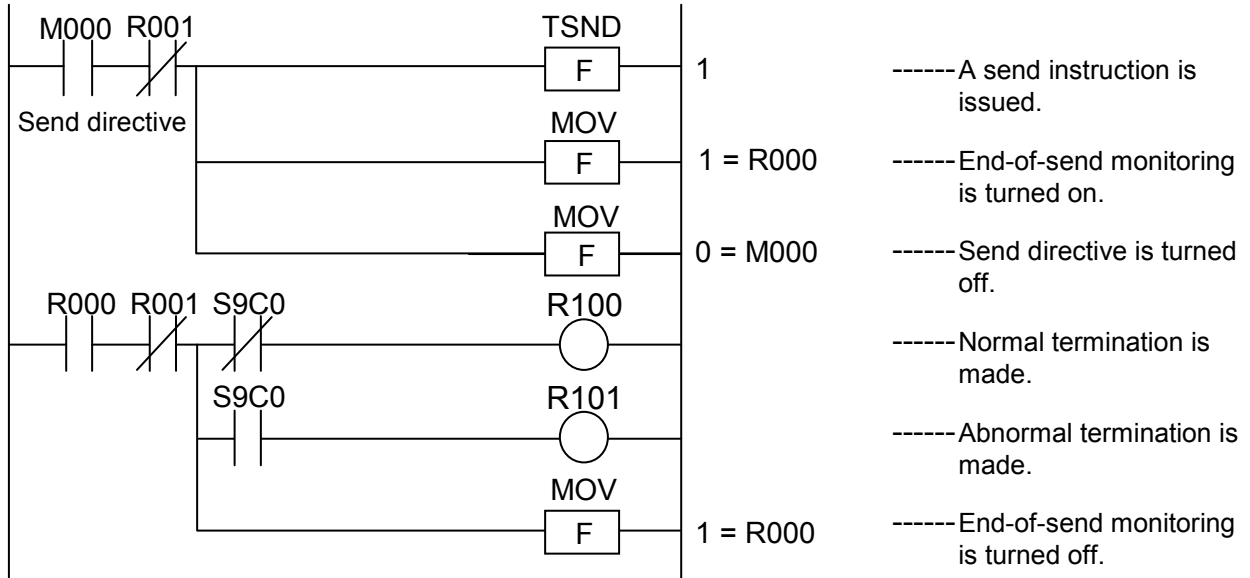
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	–	–	–	–	–

√: May be specified.

–: May not be specified.

(4) Example program

Transmitting using the management number 1 and execution flag R001:



(5) Error handling

If the initiated transmission process succeeds, the system register and [Detail result code] are both set to 0. If it fails, the system register is set to 1 and the [Detail result code] is set to the error number (non-zero value).

Whether the initiated process has succeeded or not can be determined from the set value of the system register.

If the process has failed, get error cause information from the [Detail result code].

● Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

## (1) Input format

UOP S

where:

S: (Source) is a communication identifier, which is one of the following module-specific management numbers:

Management No.	Module name
1 to 16	CMU
17 to 32	ET.NET (main module)
33 to 48	ET.NET (submodule)
49 to 80	OPTET (Module 0 to 3)

A maximum of 16 TCP and/or UDP communication processes can be executed simultaneously for each module other than the OPTET. Where OPTET modules are used for communication, a maximum total of 32 TCP and/or UDP communication processes can be executed simultaneously for the four OPTET modules.

## (2) Function

### ● Opening UDP

The UOP (UDP open) instruction opens a socket and assigns address information to that socket by using the self-port number that has been set as a parameter in the [Set Ethernet Communication] window. This instruction ends its execution even if the initiated process is not completed. The result of the process is reported by storing appropriate values in a given system register, [Execution flag], and [Detail result code], the latter two of which are parameters that have been set in the [Set Ethernet Communication] window.

**Execution flag:** Set to 1 if the initiated process is in progress; otherwise (i.e., it is completed), set to 0. When the initiated process is completed, its result is reported by setting the system register and [Detail result code] to appropriate values. To obtain the information, be sure to monitor the [Execution flag] constantly until the process is complete.

**Detail result code:** Set to an appropriate value at the end of the initiated process, the value that indicates the result of execution of the process. When the [Execution flag] is set to 0, get information from this [Detail result code].

## (3) Data types

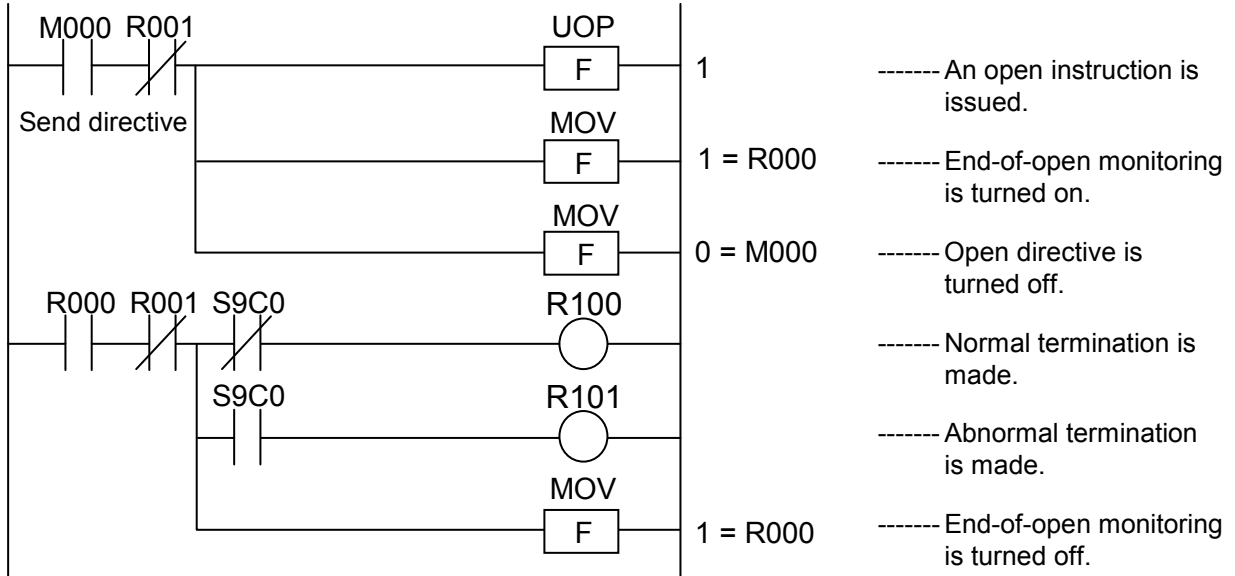
\	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	–	–	–	–	–

√: May be specified.

–: May not be specified.

(4) Example program

Opening using the management number 1 and execution flag R001:



(5) Error handling

If opening UDP succeeds, the system register and [Detail result code] are both set to 0. If it fails, the system register is set to 1 and the [Detail result code] is set to the error number (non-zero value).

Whether the initiated process has succeeded or not can be determined from the set value of the system register.

If the process has failed, get error cause information from the [Detail result code].

● Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.



## (1) Input format

UCLO S

where:

S: (Source) is a communication identifier, which is one of the following module-specific management numbers:

Management No.	Module name
1 to 16	CMU
17 to 32	ET.NET (main module)
33 to 48	ET.NET (submodule)
49 to 80	OPTET (Module 0 to 3)

A maximum of 16 TCP and/or UDP communication processes can be executed simultaneously for each module other than the OPTET. Where OPTET modules are used for communication, a maximum total of 32 TCP and/or UDP communication processes can be executed simultaneously for the four OPTET modules.

## (2) Function

### ● Closing UDP

The UCLO (UDP close) instruction logically disconnects a communication path and releases the socket. This instruction ends its execution even if the initiated process is not completed. The result of the process is reported by storing appropriate values in a given system register, [Execution flag], and [Detail result code], the latter two of which are parameters that have been set in the [Set Ethernet Communication] window.

**Execution flag:** Set to 1 if the initiated process is in progress; otherwise (i.e., it is completed), set to 0. When the initiated process is completed, its result is reported by setting the system register and [Detail result code] to appropriate values. To obtain the information, be sure to monitor the [Execution flag] constantly until the process is complete.

**Detail result code:** Set to an appropriate value at the end of the initiated process, the value that indicates the result of execution of the process. When the [Execution flag] is set to 0, get information from this [Detail result code].

## (3) Data types

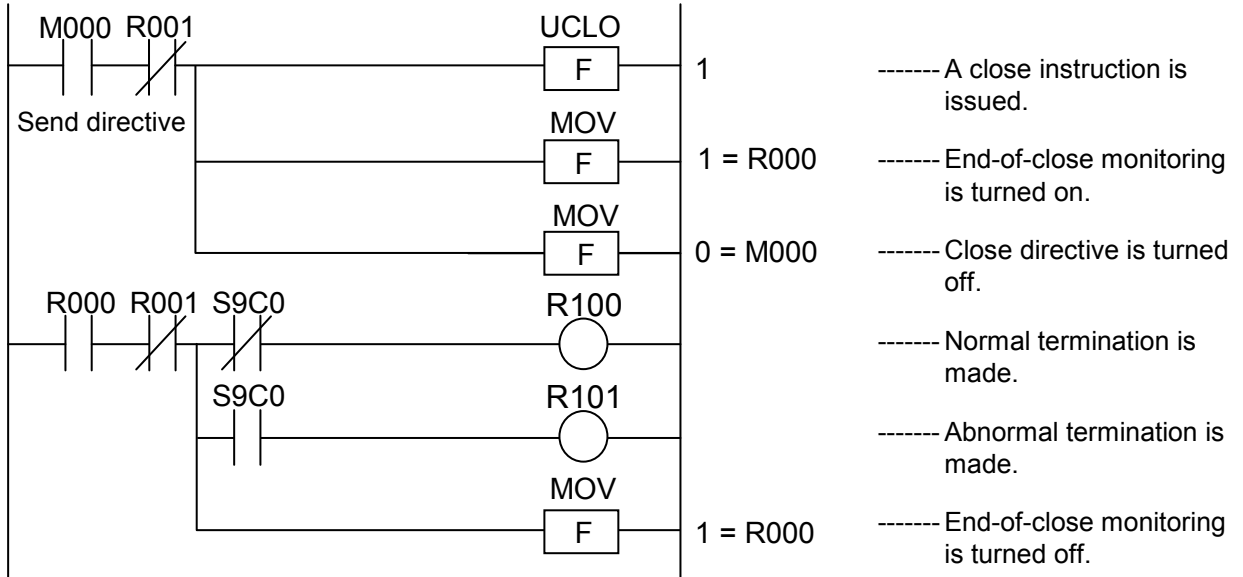
\	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	-	-	-	-	-

√: May be specified.

-: May not be specified.

(4) Example program

Closing using the management number 1 and execution flag R001:



(5) Error handling

If closing UDP succeeds, the system register and [Detail result code] are both set to 0. If it fails, the system register is set to 1 and the [Detail result code] is set to the error number (non-zero value).

Whether the initiated process has succeeded or not can be determined from the set value of the system register.

If the process has failed, get error cause information from the [Detail result code].

● Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

## (1) Input format

URCV S
--------

where:

S: (Source) is a communication identifier, which is one of the following module-specific management numbers:

Management No.	Module name
1 to 16	CMU
17 to 32	ET.NET (main module)
33 to 48	ET.NET (submodule)
49 to 80	OPTET (Module 0 to 3)

A maximum of 16 TCP and/or UDP communication processes can be executed simultaneously for each module other than the OPTET. Where OPTET modules are used for communication, a maximum total of 32 TCP and/or UDP communication processes can be executed simultaneously for the four OPTET modules.

## (2) Function

### ● Reception by UDP

The URCV (UDP receive) instruction receives as much message data as specified by the [Receive size] from a given socket and stores the received data in the receive buffer area specified by the [Receive address], where the [Receive size] and [Receive address] are parameters that have been set in the [Set Ethernet Communication] window. This instruction ends its execution even if the initiated process is not completed. The result of the process is reported by storing appropriate values in a given system register, [Execution flag], and [Detail result code], the latter two of which are parameters that have been set in the [Set Ethernet Communication] window.

If there is no data to be received at the time of its issuance, this instruction monitors the reception process for the time period specified by the [Receive timeout], which has been set in the [Set Ethernet Communication] window. If the timeout has elapsed with no data received, the reception process is terminated with the [Detail result code] set to the value "EWOULDBLOCK". In this case, if you still want to receive data, re-issue this instruction.

**Execution flag:** Set to 1 if the initiated process is in progress; otherwise (i.e., it is completed), set to 0. When the initiated process is completed, its result is reported by setting the system register and [Detail result code] to appropriate values. To obtain the information, be sure to monitor the [Execution flag] constantly until the process is complete.

**Detail result code:** Set to a positive value if data is received. This positive value indicates the size of the received data. If the size of the received data is not equal to the [Receive size] value, one of the following takes place:  
 If [Receive size] > size of received data: All the received data is read in.  
 If [Receive size] < size of received data:

As much data as specified by the [Receive size] is read in from the received data. The rest of the received data is retained and can be read in by issuing the URCV instruction again.

If the reception process fails, the [Detail result code] is set to a negative value, which is the error code.

(3) Data types

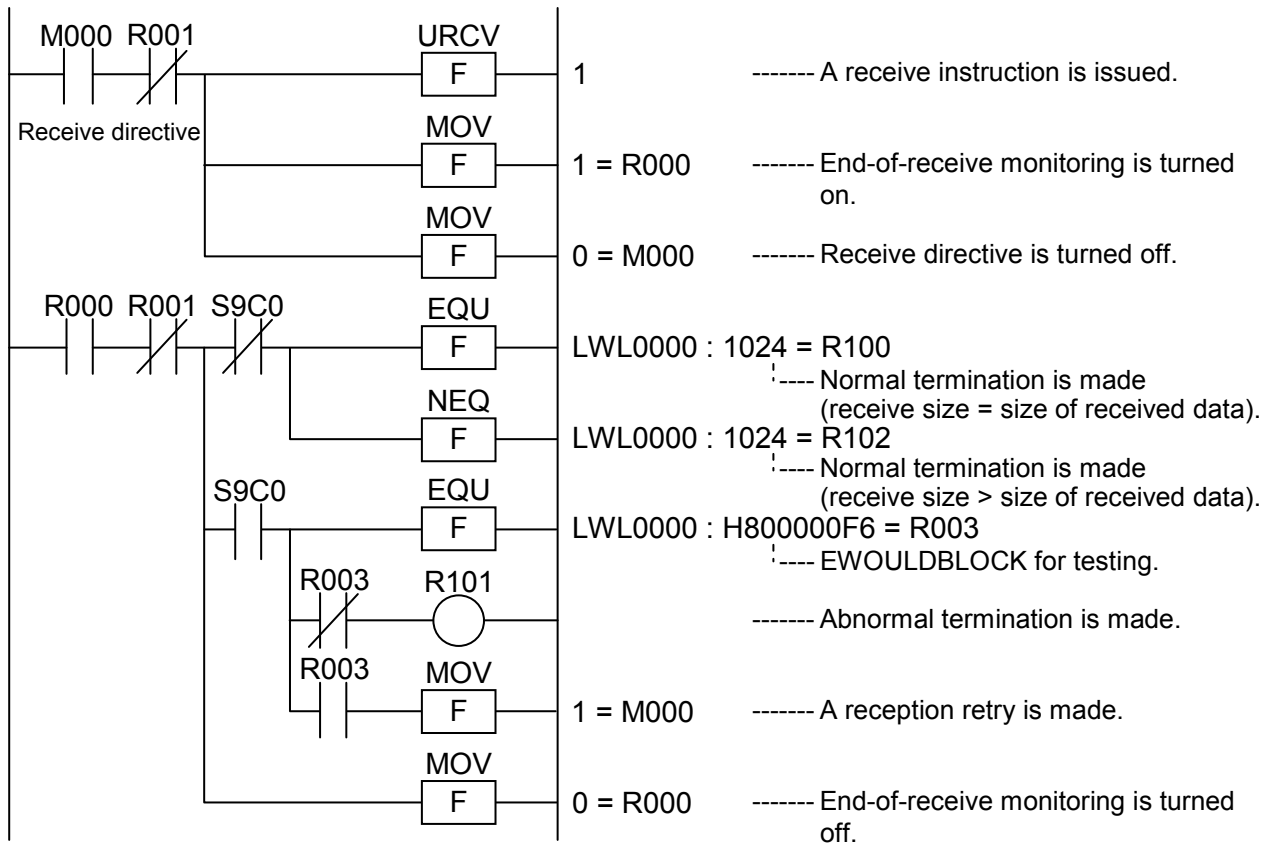
	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	-	-	-	-	-

√: May be specified.

-: May not be specified.

(4) Example program

Receiving using the management number 1, execution flag R001, and detail result code LWL0000:



### (5) Error handling

If the initiated reception process succeeds, the system register is set to 0 and the [Detail result code] is set to a value indicating the size of the received data. If it fails, the system register is set to 1 and the [Detail result code] is set to the error number (negative value). Whether the initiated process has succeeded or not can be determined from the set value of the system register.

If the process has failed, get error cause information from the [Detail result code]. If the [Detail result code] is EWOULDBLOCK, you can re-issue the URCV instruction to receive data.

#### ● Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

**This Page Intentionally Left Blank**

## (1) Input format

USND S
--------

where:

S: (Source) is a communication identifier, which is one of the following module-specific management numbers:

Management No.	Module name
1 to 16	CMU
17 to 32	ET.NET (main module)
33 to 48	ET.NET (submodule)
49 to 80	OPTET (Module 0 to 3)

A maximum of 16 TCP and/or UDP communication processes can be executed simultaneously for each module other than the OPTET. Where OPTET modules are used for communication, a maximum total of 32 TCP and/or UDP communication processes can be executed simultaneously for the four OPTET modules.

## (2) Function

- Transmission by UDP

The USND (UDP send) instruction transmits as much send data as specified by the [Send size] from the send buffer area specified by the [Send address] to a given socket, where the [Send size] and [Send address] are parameters that have been set in the [Set Ethernet Communication] window. This instruction ends its execution even if the initiated process is not completed. The result of the process is reported by storing appropriate values in a given system register, [Execution flag], and [Detail result code], the latter two of which are parameters that have been set in the [Set Ethernet Communication] window.

**Execution flag:** Set to 1 if the initiated process is in progress; otherwise (i.e., it is completed), set to 0. When the initiated process is completed, its result is reported by setting the system register and [Detail result code] to appropriate values. To obtain the information, be sure to monitor the [Execution flag] constantly until the process is complete.

**Detail result code:** Set to an appropriate value at the end of the initiated process, the value that indicates the result of execution of the process. When the [Execution flag] is set to 0, get information from this [Detail result code].

## (3) Data types

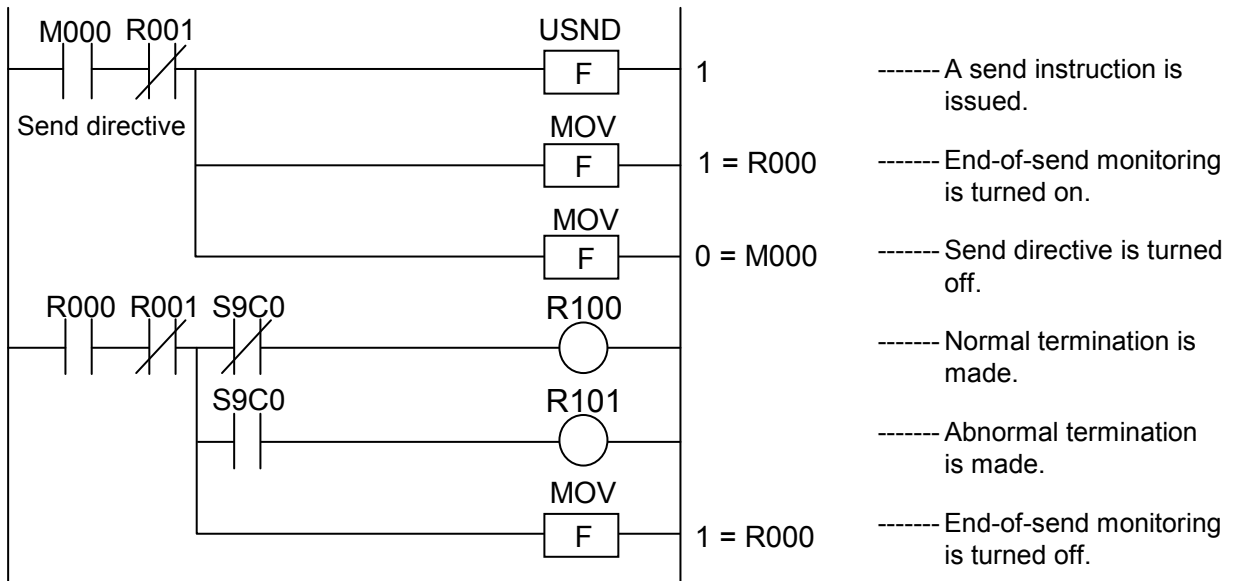
\	Word		Long-word		Floating		Index specification
	Register	Constant	Register	Constant	Register	Constant	
S	√	√	–	–	–	–	–

√: May be specified.

–: May not be specified.

(4) Example program

Transmitting using the management number 1 and execution flag R001:



(5) Error handling

If the initiated transmission process succeeds, the system register and [Detail result code] are both set to 0. If it fails, the system register is set to 1 and the [Detail result code] is set to the error number (non-zero value). Whether the initiated process has succeeded or not can be determined from the set value of the system register.

If the process has failed, get error cause information from the [Detail result code].

● Operation result flags

X	E	P	N	Z	V
-	-	-	-	-	-

All the above flags remain unchanged.

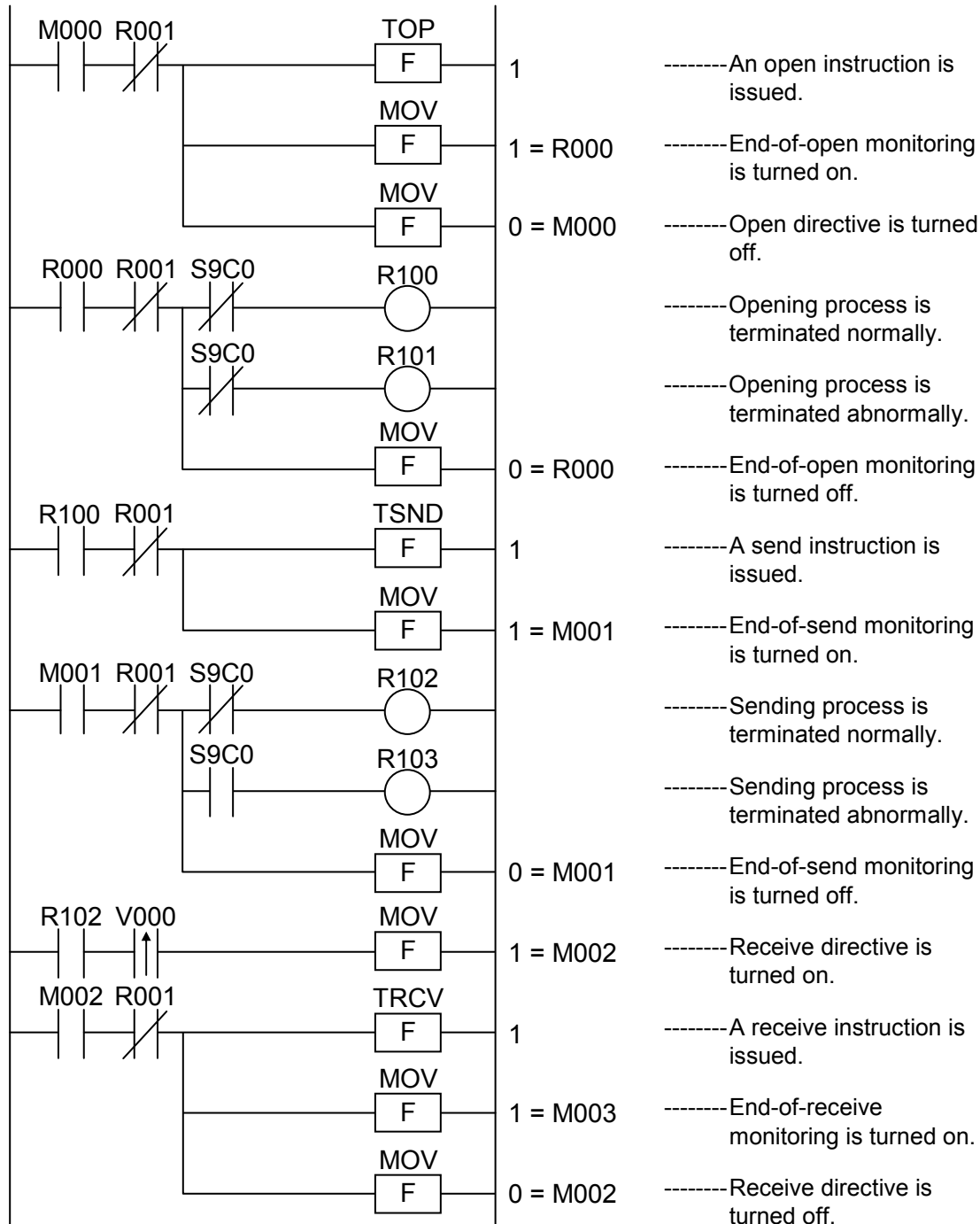


## 2 ARITHMETIC FUNCTIONS

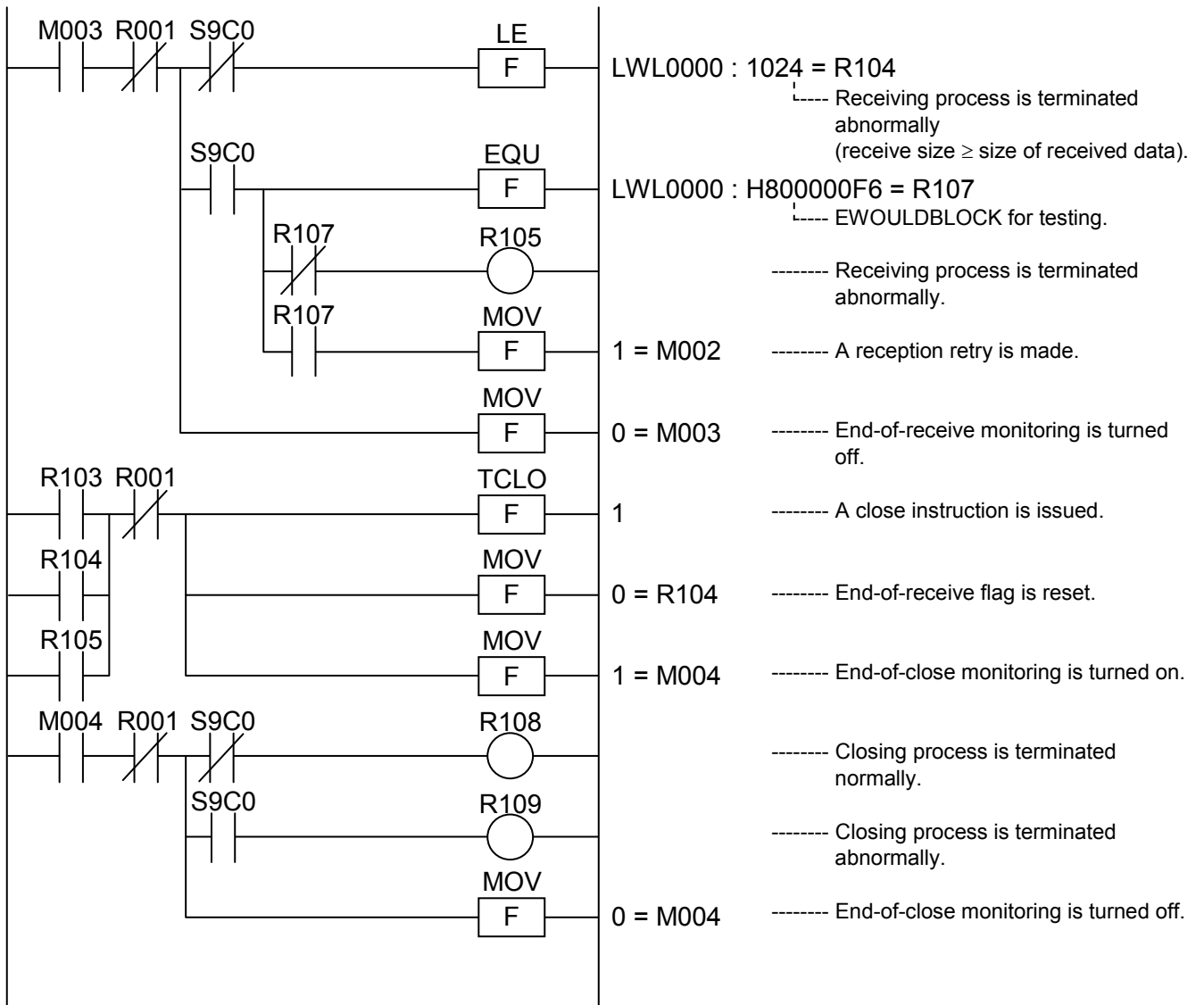
### 2.7.4 Sample programs

This section shows two sample programs each of which opens a socket, sends and receives data, and closes the socket, all by using Ethernet communication system extension functions. It is assumed that the sample programs are provided with the following parameter settings: the management number 1, execution flag R001, and detail result code LWL0000.

#### (1) TCP client

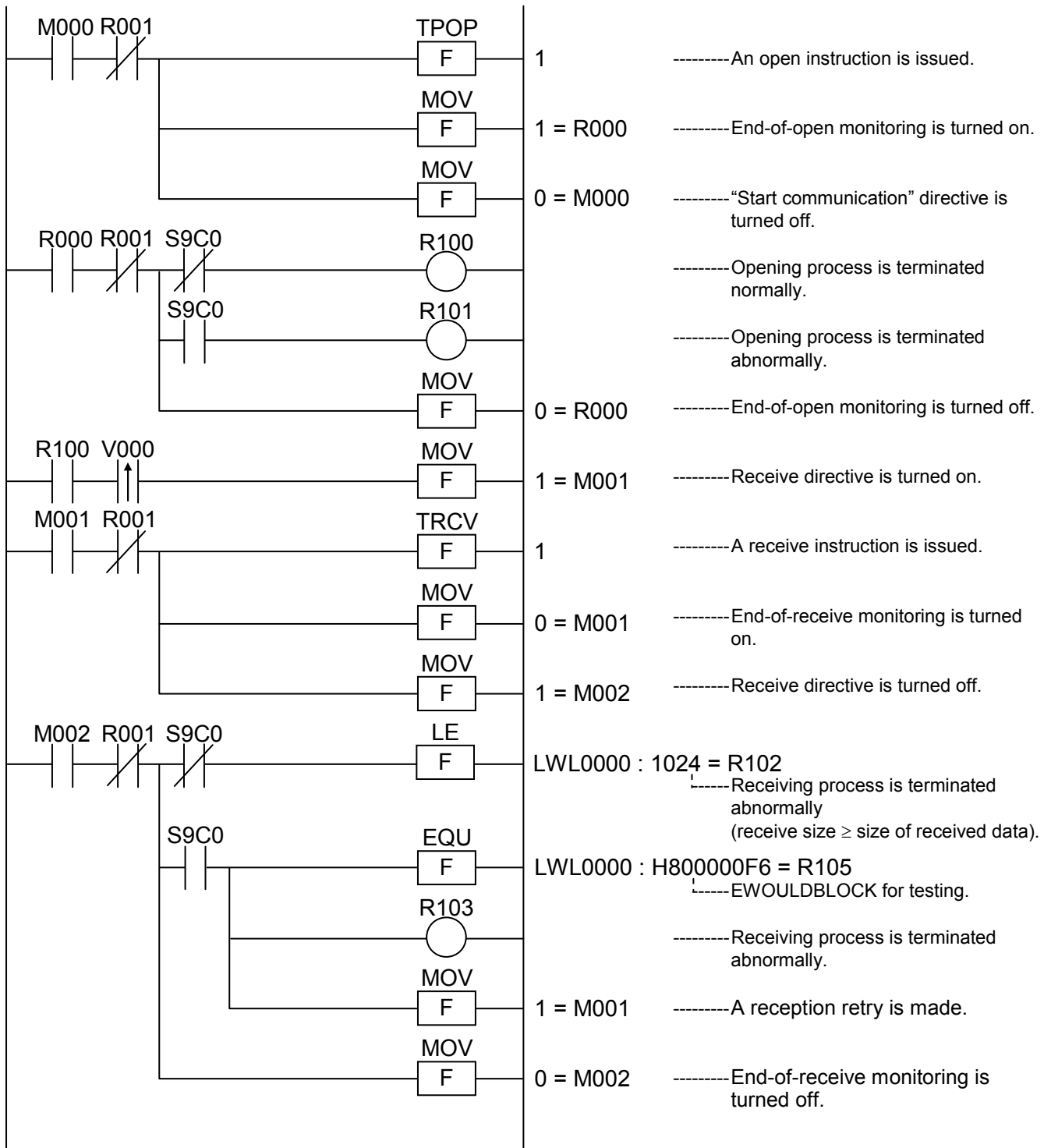


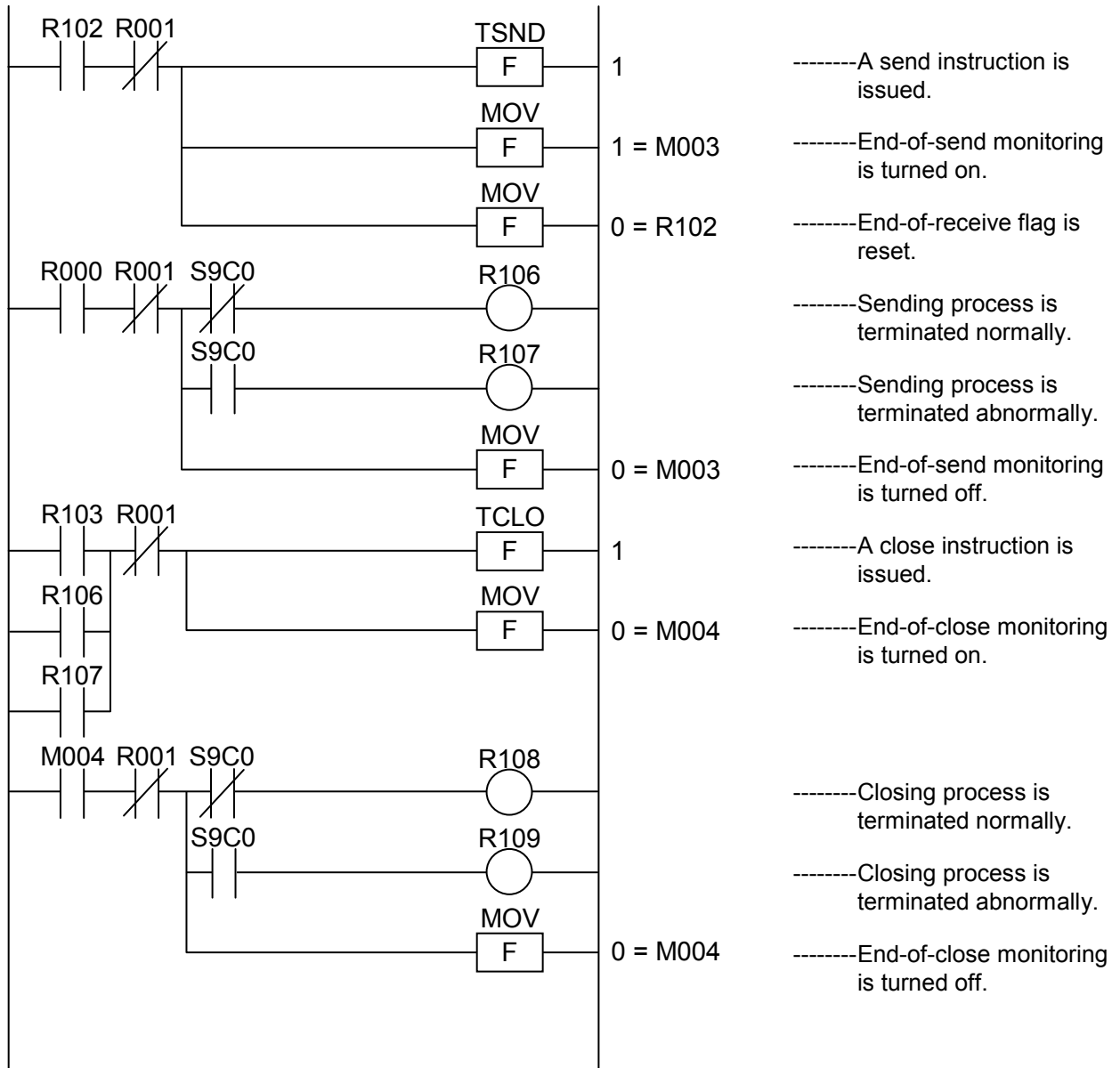
## 2 ARITHMETIC FUNCTIONS



## 2 ARITHMETIC FUNCTIONS

### (2) TCP server





**This Page Intentionally Left Blank**

# SUPPLEMENTS

**SUPPLEMENT A CHECKING OUT THE AVERAGE SCAN TIME**

There are two methods available for finding out the average scan time used in ladder programs: 1) by using the sequence cycle monitoring function of the S10V ladder chart system (model S-7895-02) and 2) by adding a special circuit to the ladder program.

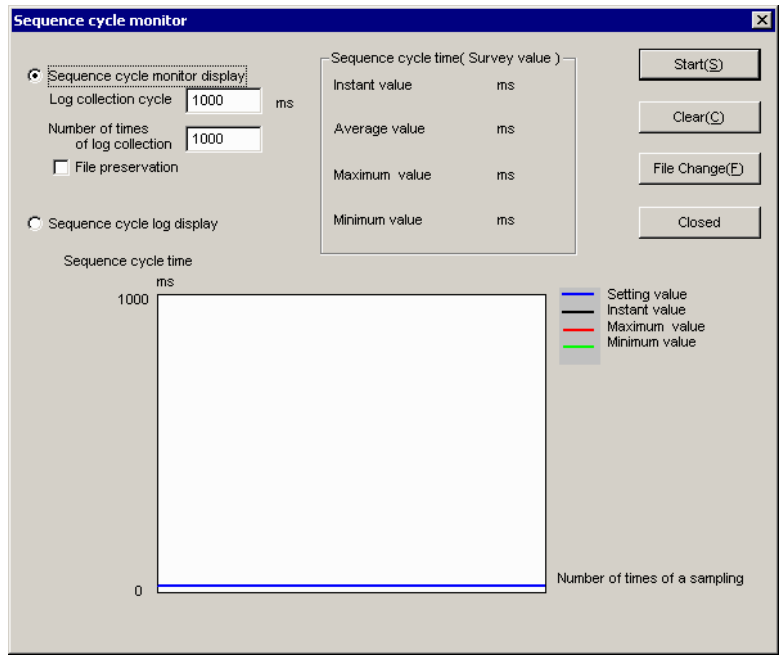
**A.1 Check-out using the S10V ladder chart system**

The S10V ladder chart system’s sequence cycle monitoring function can be used to check out the instant value, maximum value, minimum value, and average value of scan times. The procedure is as described below.

To start the monitoring function, choose [Utility] – [Monitor control status] – [Sequence cycle monitor] from the S10V ladder chart system’s menu.

For information on how to operate the sequence cycle monitor, refer to the “SOFTWARE MANUAL OPERATION S10V LADDER CHART For Windows® (Manual number SVE-3-131).”

<[Sequence cycle monitor] window>



A.2 Check-out using a ladder program

It becomes possible to check out the average scan time in ladder programs if you add the following circuit to the ladder program. The average scan time obtained is the average of scan times (milliseconds) measured at 8-second intervals during the execution of the ladder program and is stored in FWBFF.

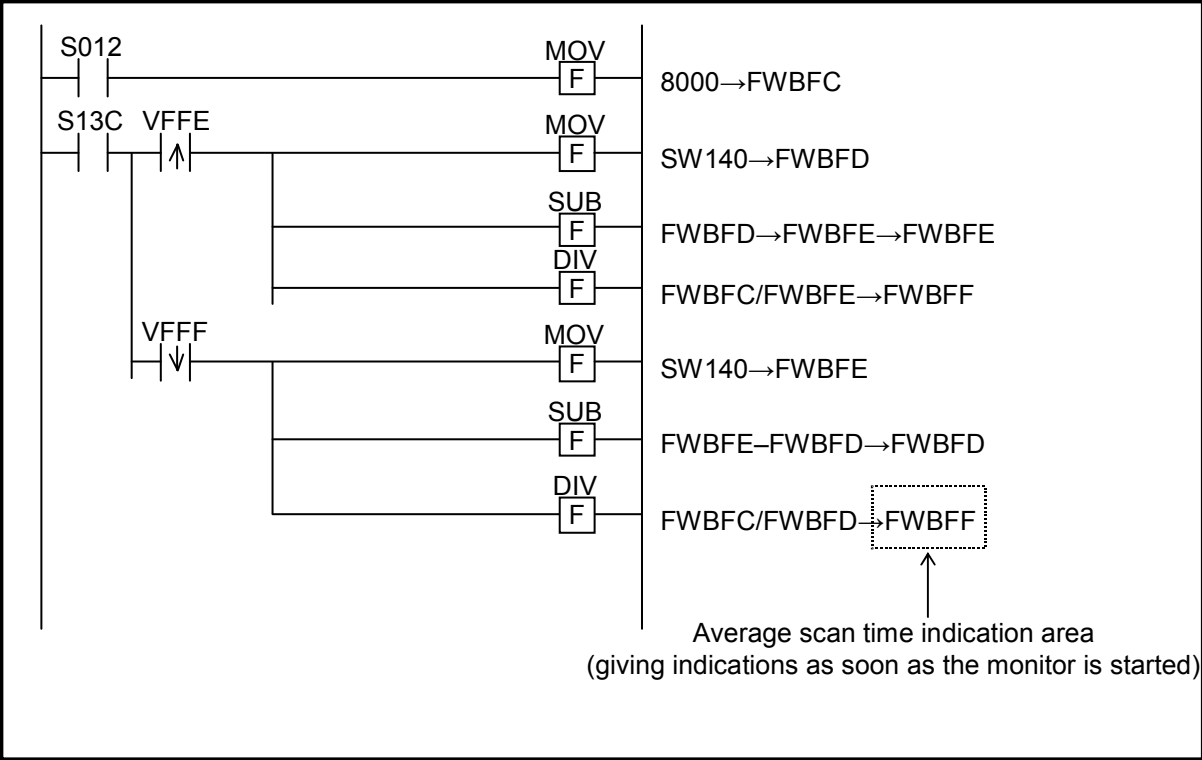


Figure A-1 Scan Time-Indicating Circuit



**SUPPLEMENT B SPECIAL PRECAUTIONS**

**B.1 Precautions in converting ladder programs**

You can convert downward-sloping-rung ladder programs used in S10/2 $\alpha$  Series and S10mini Series systems into normal-rung ladder programs for use in S10V systems by using the converter of the S10V ladder chart system (model S-7895-02). This section presents special precautions to keep in mind when you use the converter. For information on the operating procedure and functions of the converter, refer to the “SOFTWARE MANUAL OPERATION S10V LADDER CHART For Windows® (Manual number SVE-3-131).”

- The following arithmetic functions used in the S10/2 $\alpha$  Series and S10mini Series in S10V were cancelled and changed. The converter function automatically converts the arithmetic functions of the S10/2 $\alpha$  Series and S10mini Series, with some exceptions, to the S10V arithmetic functions. It’s not necessary to mind the cancellation and change of the arithmetic functions.

Arithmetic functions for S10/2 $\alpha$ Series and S10mini Series systems	Arithmetic functions for S10V system	Reason for cancellation and change
Source indirect transfer (MSI) Destination indirect transfer (MDI)	Batch transfer (MOM)	Index specifications (indirect addressing) are now supported by MOM.
Data set (DST)	Transfer (MOV)	Constants are now supported by MOV.
Square root (ROT)	Square root (SQR)	The function mnemonic has been changed.
FIFO writing (PSH)	FIFO writing (PSHO)	Because S10/2 $\alpha$ and S10mini-compatible FIFO was supported. (However, applicable for Ver.-Rev. 01-16 and later of the Ladder Chart System)
FIFO reading (POP)	FIFO reading (POPO)	Because S10/2 $\alpha$ and S10mini-compatible FIFO was supported. (However, applicable for Ver.-Rev. 01-16 and later of the Ladder Chart System)

- The S10V system’s performance is increased, compared with S10/2 $\alpha$  Series and S10mini Series systems, so S10V ladder programs requiring less processing time may show different external I/O timings than S10/2 $\alpha$  Series and S10mini Series ladder programs. Make sufficient testing before you first use S10V ladder programs with the existing equipment.

- If addresses are specified as immediate values in arithmetic functions, the converter does not perform address translation. In these cases, correct the addresses after conversion into normal-rung ladder programs. Care must be taken especially in cases where expansion memory is used.
- In S10V of LPU module revision L (Ver.-Rev. 02-05) and earlier, register numbers (such as XL000 and FL004) used for the accesses in long-word boundaries need to be specified for the long-word register according to the hardware restrictions. So, the word boundary specification (such as XL010 and FL003) is not available. The converter function does not convert these values. If an error occurs for the long-word register having an odd number at compilation, correct the register number so that the long-word register has an even number. (The following modifications are unnecessary for LPU module revision M (Ver.-Rev. 02-06) and later.)

Examples:

(Before correction) (After correction)

XL010 → XL004  
 FL003 → FL004

The table below shows the numbers specifiable for long-word registers.

The LPUs in the list of the numbers specifiable for long-word registers (module revision M (Ver.-Rev. 02-06 and later)) is excluded.

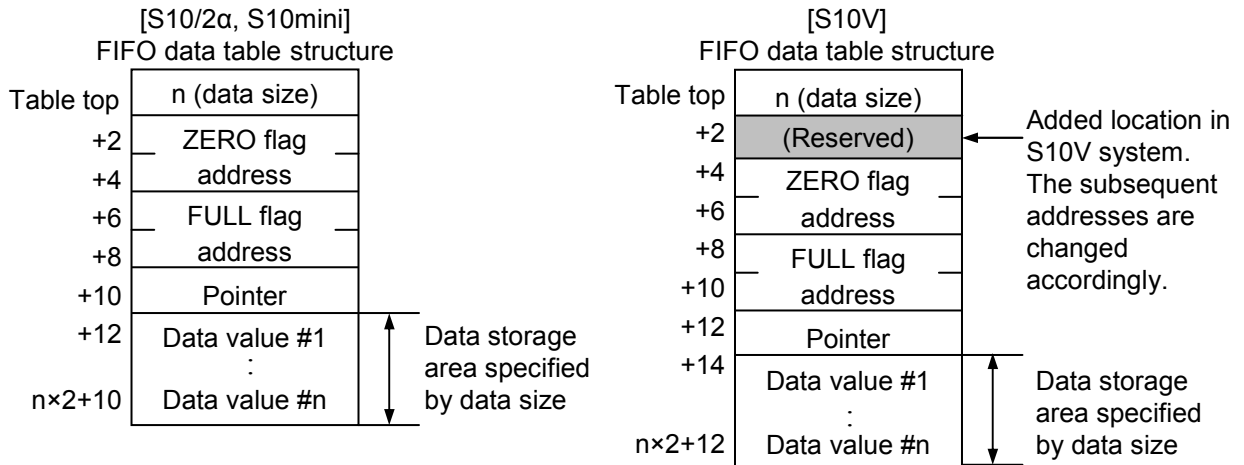
Register type	Register name	Specifiable numbers
I/O register (long-word)	XL, JL, YL, QL, GL, AL, RL, ML, KL, TL, UL, CL, NL, PL, VL, EL, ZL, SL, LBL, LRL, LVL	Numbers that have an even digit in the second least significant digit position, as in XL0 <sup>2</sup> 0 or LBL00 <sup>4</sup> 0 (boxed position)
Work register (long-word)	DL, FL, LWL, LXL	Numbers that have an even digit in the least significant digit position, as in DL00 <sup>6</sup> or LWL000 <sup>8</sup> (boxed position)

**NOTICE**

The LPU module stops due to “Invalid instruction detected.” This occurs if the long-word register having an odd number or a ladder program containing PSHO and POPO, which were created by the Ladder Chart System of Ver.-Rev. 01-16 or later, is sent to a Ladder Chart System having Ver.-Rev. 01-15 or earlier, or to an LPU having module revision L (Ver.-Rev. 02-05) or earlier using batch loading.

SUPPLEMENTS

- The FIFO table structure for FIFO-basis write (PSH) and read (POP) that is used in S10/2α Series and S10mini Series systems has been changed as shown below.



<For LPU module revision M (Ver.-Rev. 02-06) and later>

The converter function of the S10V Ladder Chart System (Ver.-Rev. 01-16 and later) converts arithmetic functions PSH and POP to PSHO and POPO and allows the use of the FIFO data table of the above [S10/2α and S10mini].

<For LPU module revision L (Ver.-Rev. 02-05) and earlier>

The converter function of the S10V Ladder Chart System (Ver.-Rev. 01-16 and later) converts arithmetic functions PSH and POP to PSHO and POPO. However, these conversion instructions cannot be used. So, change the LPU module to the one having module revision M (Ver.-Rev. 02-06) and later. Alternatively, correct the ladder program for PSH and POP, and use the FIFO data table of the above [S10V]