HITACHI
S10α SERIES

SOFTWARE MANUAL
PROGRAMMING

LADDER CHART For Windows®

2α 4α S10mini SERIES

Applicable to :

HITACHI-S10/2α
HITACHI-S10/2α E
HITACHI-S10/2α H
HITACHI-S10/2α Hf
HITACHI-S10/4α
HITACHI-S10/4α H
S10mini model S
S10mini model H
S10mini model F
S10mini model D
S10mini model C

NESP-S25E
NESP-2α E
NESP-2α H
NESP-2α Hf
NESP-S25M
NESP-4α H

HITACHI

HITACHI
S10α SERIES

SOFTWARE MANUAL
PROGRAMMING

LADDER CHART
For Windows®

2α
4α
S10mini
SERIES

Applicable to :

HITACHI-S10/2α	NESP-S25E
HITACHI-S10/2αE	NESP-2αE
HITACHI-S10/2αH	NESP-2αH
HITACHI-S10/2αHf	NESP-2αHf
HITACHI-S10/4α	NESP-S25M
HITACHI-S10/4αH	NESP-4αH
S10mini model S	
S10mini model H	
S10mini model F	
S10mini model D	
S10mini model C	

HITACHI

NOTE

All information in this manual is based on the latest product information available at the time of printing. Hitachi has reviewed the accuracy of this manual, but assumes no responsibility for any omissions or errors which may appear. The design of the product is under constant review and, while every effort is made to keep this manual up to date, the right is reserved to change specifications and equipment at any time without prior notice.

PROHIBITION

These products should not be used for medical, power supply, nuclear, water supply, drainage plants, traffic control, military, space, nor disaster prevention equipment.

Diversion and/or resale of these products without this manual is prohibited.

Reproduction of the contents of this manual in whole or in part, without written permission of Hitachi, is prohibited.

TRADEMARKS

HITACHI-S10/2α, S10/4α and PSEα are registered trademarks of Hitachi, Ltd.
MS-DOS is a registered trademark of Microsoft Corporation.
Ethernet is a trademark of U.S. Xerox Corporation.
PS/2 is a registered trademark of International Business Machine, Inc.

FIRST EDITION, JUNE, 1997, SAE - 3 - 121 (A) (out of print)
SECOND EDITION, NOVEMBER, 2000, SAE - 3 - 121 (C) (out of print)
THIRD EDITION, SEPTEMBER, 2003, SAE - 3 - 121 (D) (out of print)
FOURTH EDITION, OCTOBER, 2007, SAE - 3 - 121 (E)

IC-IC-MK<IC-IC> (FL-MW20, AI8.0)

LIMITED WARRANTY

Hitachi, Ltd., warrants its products to be manufactured in accordance with published specifications and free from defects in materials and/or workmanship.

Hitachi, Ltd., warrants its products against defects in parts and workmanship for one full year from date of purchase.

HITACHI, LTD., MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED EXCEPT AS PROVIDED HEREIN, INCLUDING WITHOUT LIMITATION THEREOF, WARRANTIES AS TO MARKETABILITY FOR A PARTICULAR PURPOSE OF USE, OR AGAINST INFRINGEMENT OF ANY PATENT. IN NO EVENT SHALL HITACHI BE LIABLE FOR ANY DIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES OF ANY NATURE, OR COSTS, CHARGES, LOSSES OR EXPENSES RESULTING FROM ANY DEFECTIVE PRODUCT OR THE USE OF ANY PRODUCT.

SOFTWARE UP–TO DATE POLICY

Hitachi, Ltd., constantly reviews its software so as to incorporate the latest technology. Hitachi reserves the right to make changes to any software to improve reliability, function, or design. Hitachi cannot be held responsible for any errors in its software.

⚠ 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 the 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.

⚠ "RUN/STOP" SWITCH CAUTION

The "RUN/STOP" switch only stops execution of the ladder logic program or HI-FLOW program. Digital and analog outputs are left in the active state when execution stops, unless the optional rungs described in the CPU manual have been added. The "RUN/STOP" switch does not affect the operation of C-language or FA-BASIC language programs. Outputs can still be produced in response to C-language or FA-BASIC programs, or by the action of programmers typing in commands in these languages, while the "RUN/STOP" switch is in the "STOP" position.

DO NOT DEPEND ON THE STOP SWITCH TO STOP MOVING PARTS OR TO PREVENT UNEXPECTED MOTION OR ENERGIZATION. USE HARDWIRED SAFETY DISCONNECT AND LOCK OUT POWER AND CONTROL VOLTAGES BEFORE WORKING ON ELECTRICAL CIRCUITS OR PARTS THAT CAN MOVE.

Programming Terminal
PSEα Specifications

Supply voltage		100-120 VAC $^{+10}_{-15}$ % single-phase 50/60 Hz±4 Hz	
Power requirement	Continuous	130 VA	
	Surge	6,000 VA	
Temperature		Operational	Storage
		50 to 95 ˚F (10 to 35 ℃)	23 to 122 ˚F (-5 to +50 ℃)
Humidity		40-80% RH	10-98% RH
Vibration (Max)		0.5 G, 17 Hz vibration applied for 30 s	
Dust		0.1 mg/m^3 or less	
Dimensions	EL cover closed	400 W×110 H×350 D (mm)	
	EL cover open	400 W×230 H×350 D (mm)	
Weight		Approx. 4.5 kg (10 lb)	

This manual provides information for the following program products:

<Program products>
S-7890-02, Ladder Chart System, 07-07
S-7890-17, 4α Ladder Chart System, 07-06
S-7890-18, 4αH Ladder Chart System, 07-06
S-7890-49, 4α Replace Ladder Chart System, 01-00

The following change has been made to the 4α Replace Ladder Chart System (01-00) and covered in this manual.

Description of added changes	Page
Corrections have been made by newly supporting the 4α Replace Ladder Chart System.	ix, 11, 13, 18, 21, 26

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

Revision record

Revision No.	Revision record (revision details and reason for revision)	Month, Year	Remarks
E	Corrections have been made by newly supporting the 4α Replace Ladder Chart System.	October 2007	4α Replace Ladder Chart System, 01-00 or later

THIS PAGE INTENTIONALLY LEFT BLANK.

PREFACE

This manual contains the descriptions of the instruction words necessary for preparing a ladder program.
The instruction words are generally classified into two types: ladder instructions and arithmetic function instructions.
The ladder instructions are used for the operations of the relay circuits, while the arithmetic function instructions are used for the arithmetic operations such as addition, subtraction, multiplication, and division.

Unless otherwise stated in this manual, the symbol "2α" represents the Models 2α, 2αE, 2αH, and 2αHf; "4α" represents the Models 4α and 4αH; and "S10mini" represents the S10mini Models S, H, F, D, and C.

<Related manuals>

- Software Manual Operation LADDER CHART For Windows® V7 (Manual number SAE-3-131)
- Software Manual Operation 4α REPLACE LADDER CHART For windows® (Manual number SAE-3-156)

For HI-FLOW programs, refer to the following manual:

- Software Manual Programming HI-FLOW For Windows® (Manual number SAE-3-122)
- Software Manual Option CPU LINK For Windows® (Manual number SAE-3-141)

See the following list when you use the NESP (Nissan Electronic Sequence Processor) series.

【HITACHI-S10α series】		【NESP series】
HITACHI-S10/2α	·········	NESP-S25E
HITACHI-S10/2αE	·········	NESP-2αE
HITACHI-S10/2αH	·········	NESP-2αH
HITACHI-S10/2αHf	·········	NESP-2αHf
HITACHI-S10/4α	·········	NESP-S25M
HITACHI-S10/4αH	·········	NESP-4αH

<Trademarks>

- Microsoft® Windows® operating system, Microsoft® Windows® 95 operating system, Microsoft® Windows® 98 operating system, Microsoft® Windows® 2000 operating system, Microsoft® Windows® XP operating system are registered trademarks of Microsoft Corporation in the United States and/or other countries.
- Ethernet® is a registered trademark of Xerox Corp.
  All trademarks are the properties of their respective companies.

CONTENTS

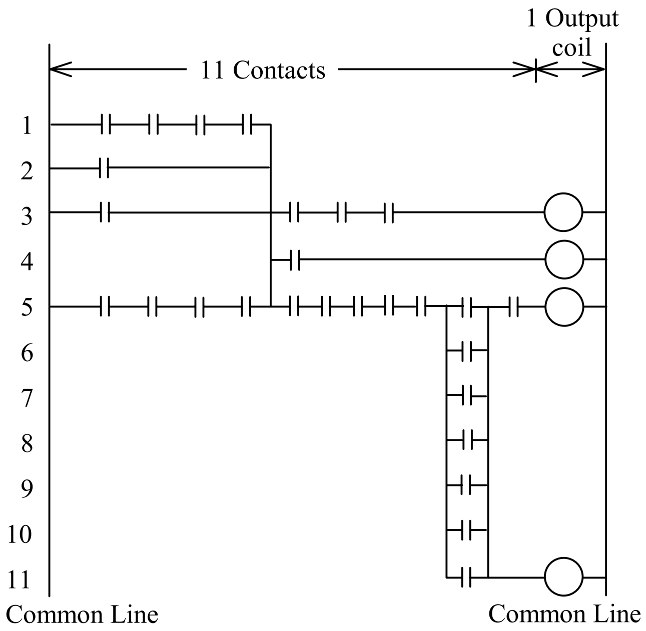
FIGURES

TABLES

THIS PAGE INTENTIONALLY LEFT BLANK.

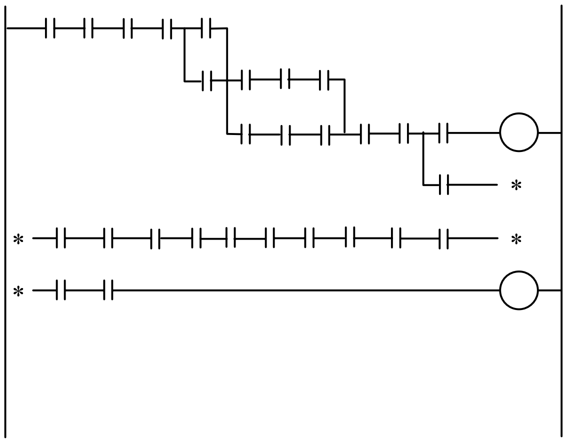
1 LADDER INSTRUCTIONS

1.1 Size of Ladder Circuit



- One ladder circuit is composed of one or more coils that start at the left common line and reach the right common line. A ladder circuit shown in one sheet of chart is called a block.
- Largest Circuit of one Block
  Horizontally: 11 contacts and 1 output coil
  Vertically: 11 lines

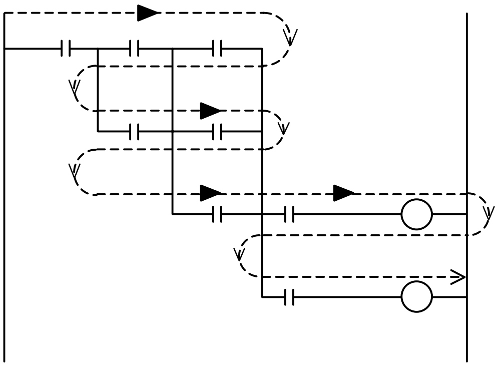
[AND Connection of more than 11 Contacts]



- If there are many AND connections, a circuit is folded as shown in the figure, permitting a circuit with more than 11 AND connections.
  Restrictions:
  - An asterisk (*) shows a connection of the coil. An asterisk at the right end means coil incomplete and an asterisk at the left end, the continuation of the coil.
  - No branch can remain before an asterisk mark.
  - No branch circuit can be made after an asterisk mark.

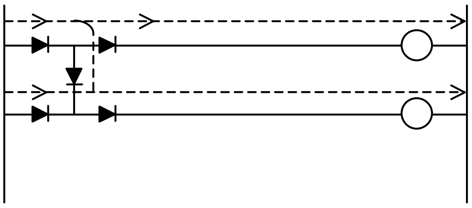
1.2 Right-Down Circuit and Sequence of Operations

[Example of Right-Down Circuit]



In a ladder circuit, contacts are placed left to right and up to down.

[Concept of Operations Sequence]
Image of Diodes Being Placed
in a Ladder circuit

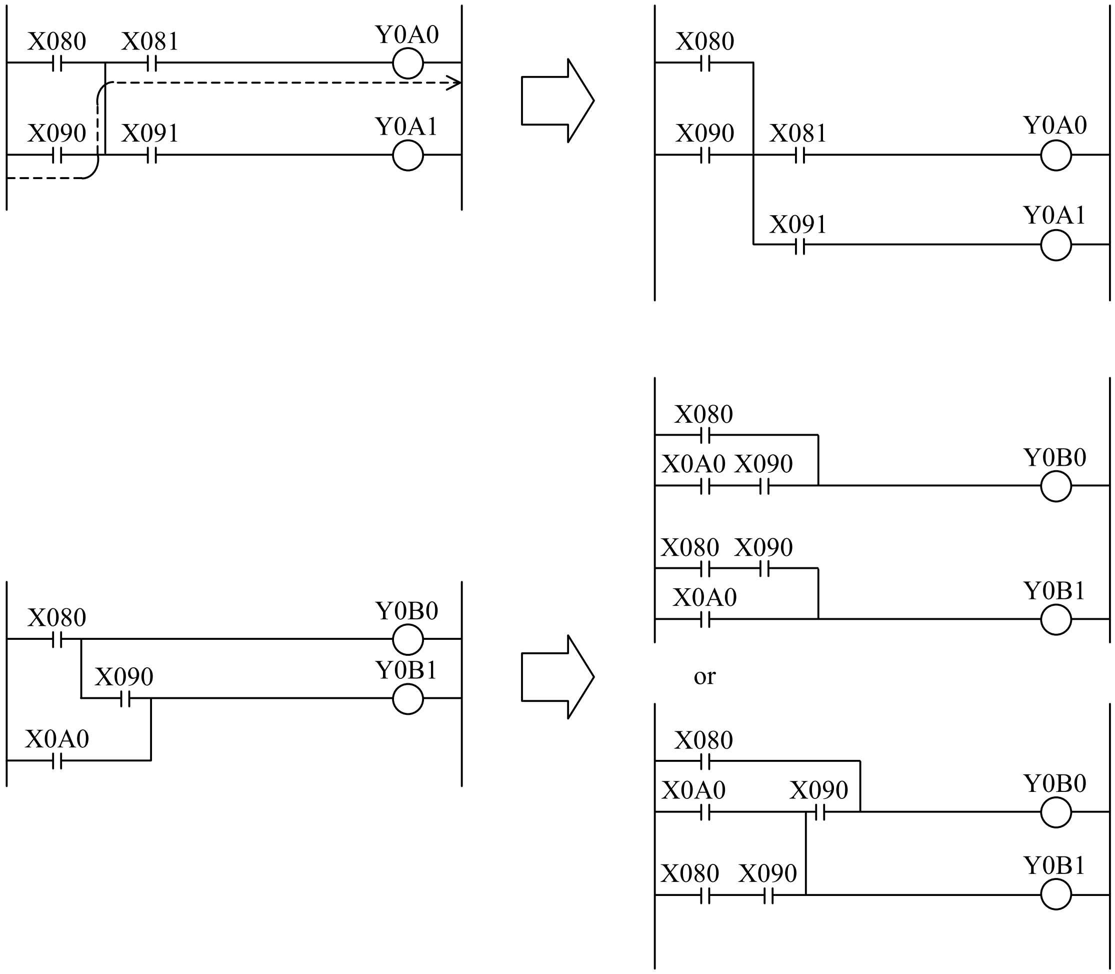


In a ladder circuit, the coils work in the order of their locations in the circuit, left to right and up to down, imaging diodes are placed at the contacts and downward branches.

[Operation of Right-Down Ladder Circuit]

To activate the relay circuits shown on the left side in the following figure, make the ladder circuits falling rightward (called right-down ladder circuits), as shown on the right side of the figure.

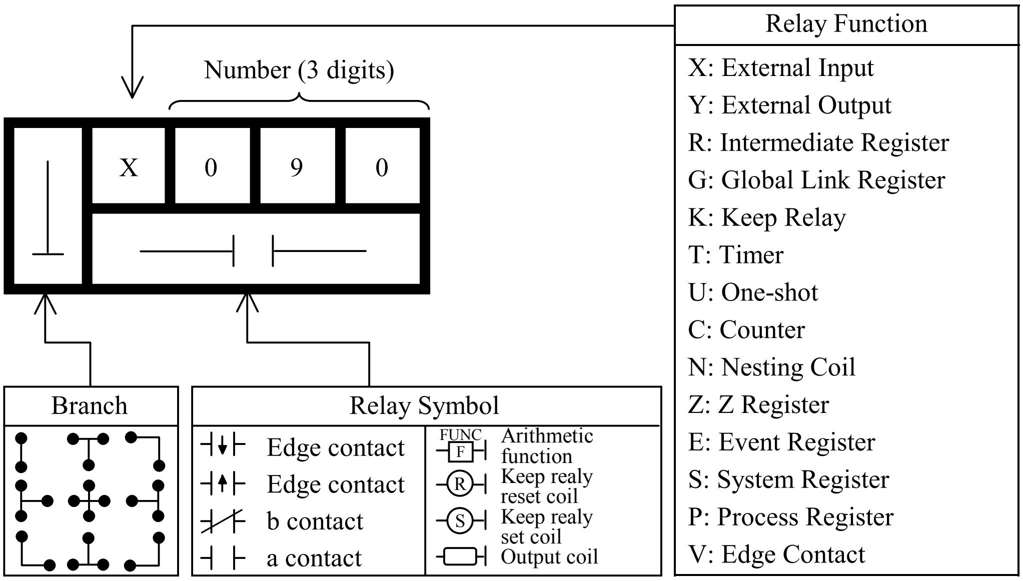
[Example of Relay Circuit] [Example of Right-Down Ladder Circuit]



1.3 Ladder Circuit and Step

One step of a ladder circuit is a conception to cover a branch, relay symbol, relay function, and number in one control unit, and is composed as shown below:

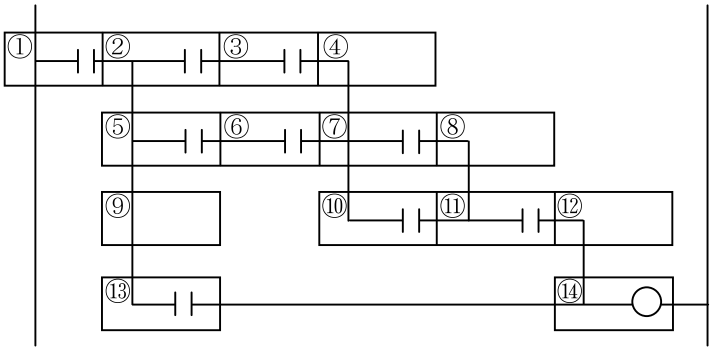
[Structure of 1 Step]



X: External Input
Y: External Output
R: Intermediate Register
G: Global Link Register
K: Keep Relay
T: Timer
U: One-shot
C: Counter
N: Nesting Coil
Z: Z Register
E: Event Register
S: System Register
P: Process Register
V: Edge Contact

A branch, ┐, ┤, or │ constitutes 1 step.

[Ladder Circuit and Instruction Word]



□: 1 step

The ladder circuit shown in the figure contains 14 steps in all. The program is stored and executed in the order of n. (n=1, 2, ...)

1.4 Status at Power Outage/Recovery and STOP to RUN

Relay Function	ON/OFF State		Discrete Value/Numeric	
	Power Outage/Recovery	STOP to RUN	Power Outage/Recovery	STOP to RUN
T, U	OFF	Hold	Clear to 0	Hold
C	Hold	Hold	Hold	Hold
K	Hold	Hold		
X, Y, R, M, G, N, Z, E, P, V	OFF	Hold		
FW, DW			Hold	Hold

1.5 Ladder Instruction List

	Name	Sym-bol	┤├ ┤/├	┤↓├	┤↑├	─( )─	─(S)─	─(R)─	Number		Remarks
									2α, S10mini	4α	
Input/Output	External Input	X	●	—	—	—	—	—	000 to FFF	000 to FFF	The input/output range for remote I/O processing is as stated below: 2α: 000 to 7FF 4α: 000 to 1FF 4αH: 000 to 3FF
	External Output	Y	●	—	—	●	—	—	000 to FFF	000 to FFF	
Internal Auxiliary Function	Internal Register	R	●	—	—	●	—	—	000 to FFF	000 to FFF	
	Keep Relay	K	●	—	—	—	●	●	000 to FFF	000 to FFF	
	Extended internal register	M	●	—	—	●	—	—	000 to FFF	000 to FFF	
	On-Delay Timer	T	●	—	—	●	—	—	000 to 1FF	000 to 1FF	000 to 00F (Changeable to 10 ms times)
	One-Shot Timer	U	●	—	—	●	—	—	000 to 0FF	000 to 0FF	
	Up-Down Counter	CU	—	—	—	●	—	—	00 to FF	00 to FF	CU: Up Counter
		CD	—	—	—	●	—	—			CD: Down Counter
		CR	—	—	—	●	—	—			CR: Reset Coil
		CO	●	—	—	—	—	—			CO: Contact
	Global Link Register	G	●	—	—	●	—	—	000 to FFF	000 to FFF	
	Nesting Coil	NM	—	—	—	●	—	—	01 to FF	—	NM: Master Reset NZ: Zone Control NM and NZ are selected for each Number input. NO: Contact (Starting State)
		NZ	—	—	—	●	—	—			
		NO	●	—	—	—	—	—			
	Process Register	P	●	—	—	●	—	—	000 to 080	—	For start of C program
	Event Register	E	●	—	—	●	—	—	000 to FFF	000 to FFF	CPU LED indications are provided 000 to 1FF at 2α only. 400 to FFF are used in 4-channel analog pulse counter.
	Edge Contact	V	—	●	●	—	—	—	000 to FFF	000 to FFF	
	Z Register	Z	●	—	—	●	—	—	000 to 3FF	000 to 3FF	OFE: Trace Start Signal OFF: Trace Stop Signal 200: H-7338 Mode Interruption Start Signal (2α and 4α only)
	System Register	S	●	—	—	—	—	—	000 to 005		Arithmetic Function Flag
									010 to 01F		Ladder Program Control Register
									100 to 15F		Ladder Program Control Counter
									300 to 47F		Remote I/O Status Display Register
									500 to 6FF	580 to 5BF	Option Module Status
										6C0 to 6FF	Display Register
									BA0 to BBF		I/O Fixed Pattern
									BF0 to BFF		CPU Status Register
	Function Data Register	DW	—	—	—	—	—	—	000 to FFF	000 to 7FF	Constant Data Area
	Function Work Register	FW	—	—	—	—	—	—	000 to BFF	000 to BFF	Work Area

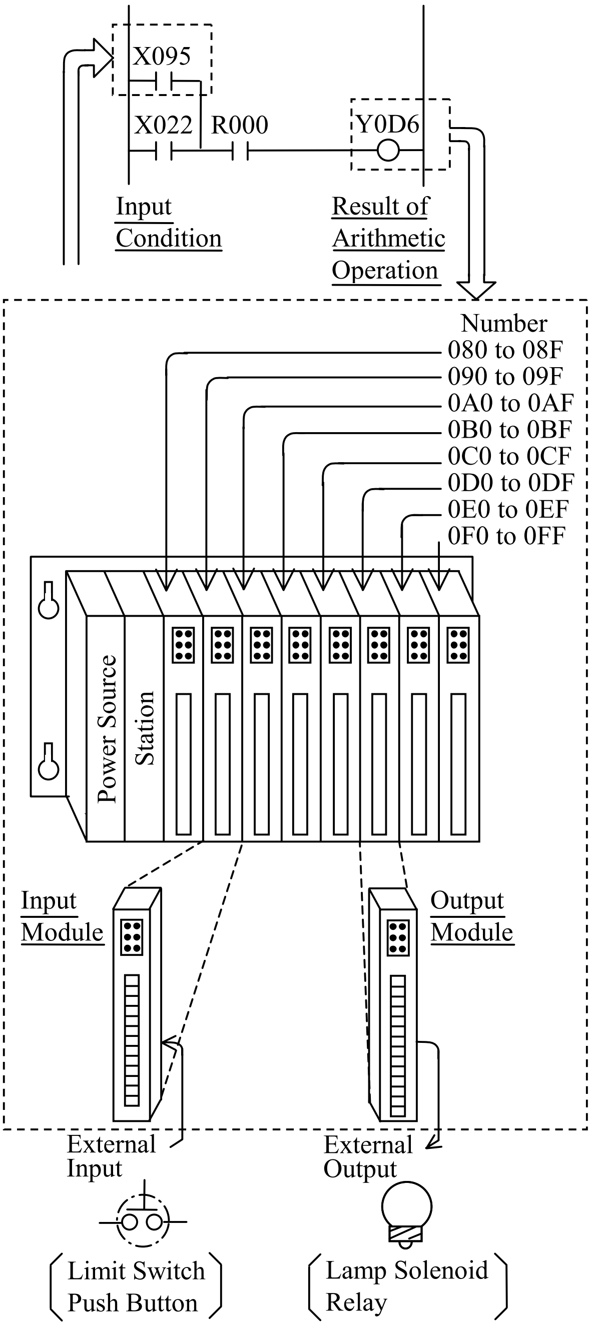
●: Permitted symbol

The external input/output sends/receives a signal via the external input/output module connected to PCs.

X: Reads an external input signal through the input module.

Y: Sends out the result of an arithmetic operation of a ladder program through the output module.

	2α, S10mini	4α
Range of Numbers	000 to FFF	000 to FFF
Input/output range for remote I/O processing	000 to 7FF	000 to 1FF (4α) 000 to 3FF (4αH)

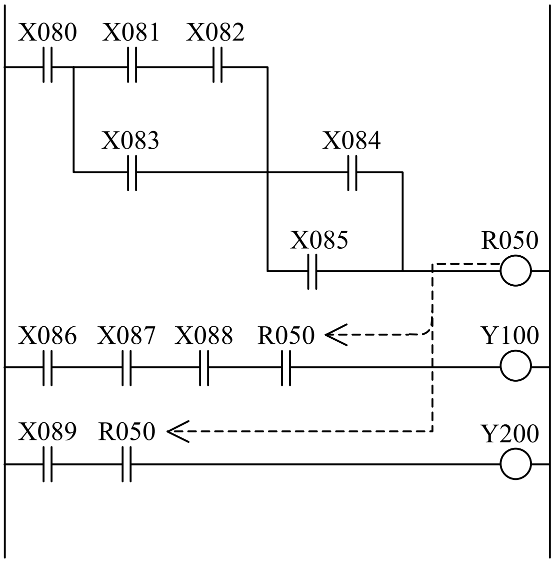


R, M　Internal Register

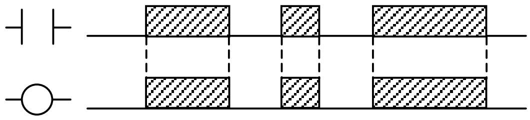
	2α, S10mini, 4α
Range of Numbers	000 to FFF

The internal register relays the arithmetic result of the ladder instruction.
Contacts (┤ ├) turn ON if a coil (-○-) is turned ON.　Contacts (┤ ├) also turn OFF if a coil (-○-) is turned OFF.

<Example of Circuit>



<Time Chart>

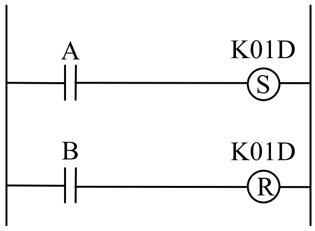


	2α, S10mini, 4α
Range of Numbers	000 to FFF
Settling Pulse Width	Minimum scan time
When a set and a reset signal are simultaneously input	The one appearing later in the program is given the priority.

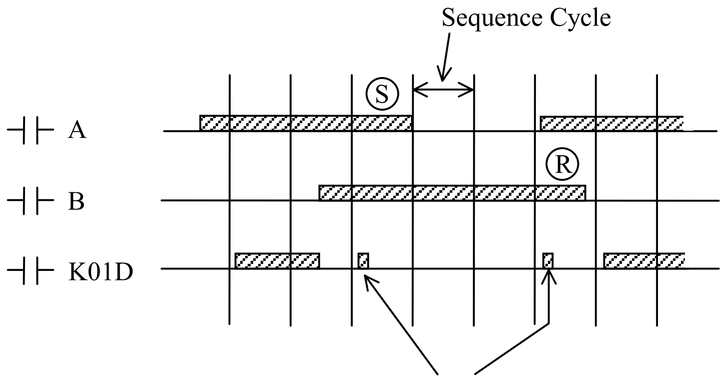
A keep relay holds a contact (┤ ├) ON until a reset coil (-Ⓡ-) is turned ON, if a set coil (-Ⓢ-) turns ON. The state of the contact is maintained even if the power is interrupted.
If both the set and reset coils are turned ON at the same time, the one appearing later in the program is given the priority.

[Reset Preceding Circuit]

<Example of Circuit>



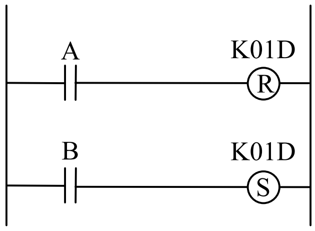
<Time Chart>



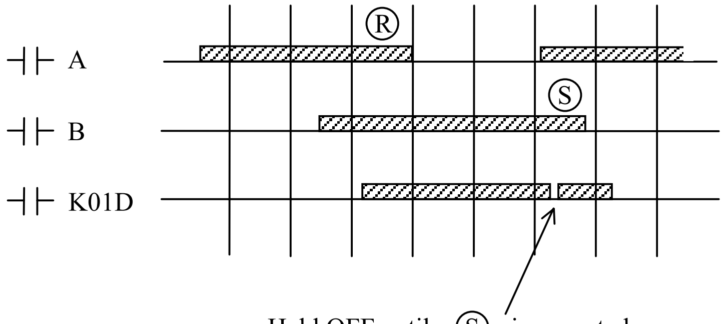
Held ON until -Ⓡ- is executed
after execution of -Ⓢ-

[Set Preceding Circuit]

<Example of Circuit>



<Time Chart>



Held OFF until -Ⓢ- is executed
after execution of -Ⓡ-

T On-Delay Timer

[100 ms Timer]

	2α, S10mini, 4α
Range of Numbers	000 to 1FF
Set Value	0.0 to 999.9 sec.
Error	100 ms + 1 scan time, minimum
Settling (*) Pulse Width	100 ms, minimum

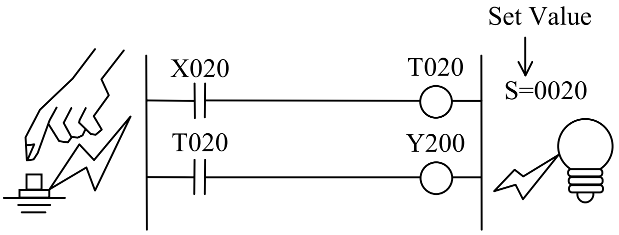
A contact (┤├) turns ON when the time specified by a set value has elapsed after the on-delay timer coil (─○─) turned ON.
The range of set value is 0.0 to 999.9 sec., and can be set in units of 0.1 seconds.
You can use the leading 16 points (T000 to T00F) as a 10 ms timer by edition setting.

[10 ms Timer (By Edition Setting)]

	2α, S10mini, 4α
Range of Numbers	000 to 0FF
Set Value	0.01 to 99.99 sec.
Error	10 ms + 1 scan time, minimum
Settling (*) Pulse Width	10 ms, minimum

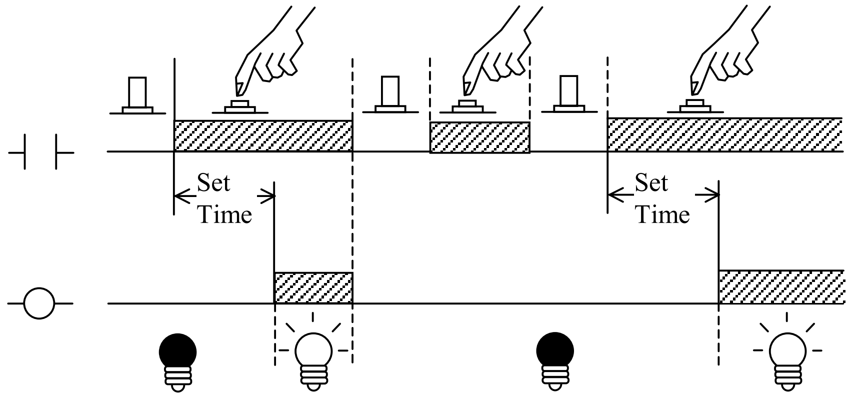
(*) The term "settling pulse width" denotes the minimum time period during which the contacts for turning the on-delay timer coil ON must remain ON (closed).

<Example of Circuit>



With the circuit shown in the figure, the lamp (Y200) is lit in a set time (2 seconds in this case) after the push button (X020) is pushed down, and extinguished as soon as the button is released.

<Time Chart>



If the coil(−◯−)-ON period is shorter than the set value, the contacts (┤ ├) will not turn ON (i.e., they remain open).
For 2α, the count value is up to 65535.　When the counter overflows exceeding 65535, counting restarts beginning with 0.

<Notice>

(1)　S10/2α series, S10mini series (Models S, H, F, and D)
When a 100-ms timer is used, the ON/OFF detection process for the on-delay timer coil is executed at 100-ms intervals, asynchronously with the ladder circuit execution cycle (sequence cycle).　(When a 10-ms timer is used, the ON/OFF detection process is performed at 10-ms intervals.)　If the coil is ON for a period shorter than 100 ms, the on-delay timer may not operate due to an unsuccessful coil-ON state detection.　To assure proper operation of the on-delay timer, prepare the ladder circuit in such a manner that the coil-ON time is 100 ms or longer.

(2)　S10/4α series, S10mini (Model C)
When a 100-ms timer is used, the ON/OFF detection process for the on-delay timer coil is executed with the ladder circuit coil execution.　(When a 10-ms timer is used, this process is performed at 10-ms intervals, asynchronously with the ladder circuit execution cycle (sequence cycle)).

U One-Shot

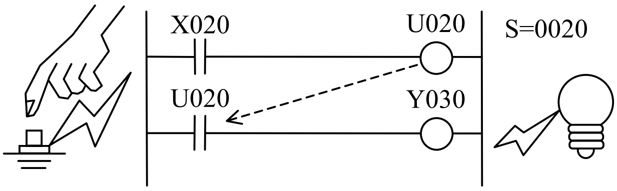
	2α, S10mini, 4α
Range of Numbers	000 to 0FF
Set Value	0.0 to 999.9 sec.
Error	100 ms + 1 scan time, minimum.
Settling (*) Pulse Width	100 ms, minimum.

A contact (⊣ ⊢) turns ON when the time specified by a set value has elapsed after the one shot coil (–○–) turned ON.

The range of set value is 0.0 to 999.9 sec., and can be set in units of 0.1 seconds.

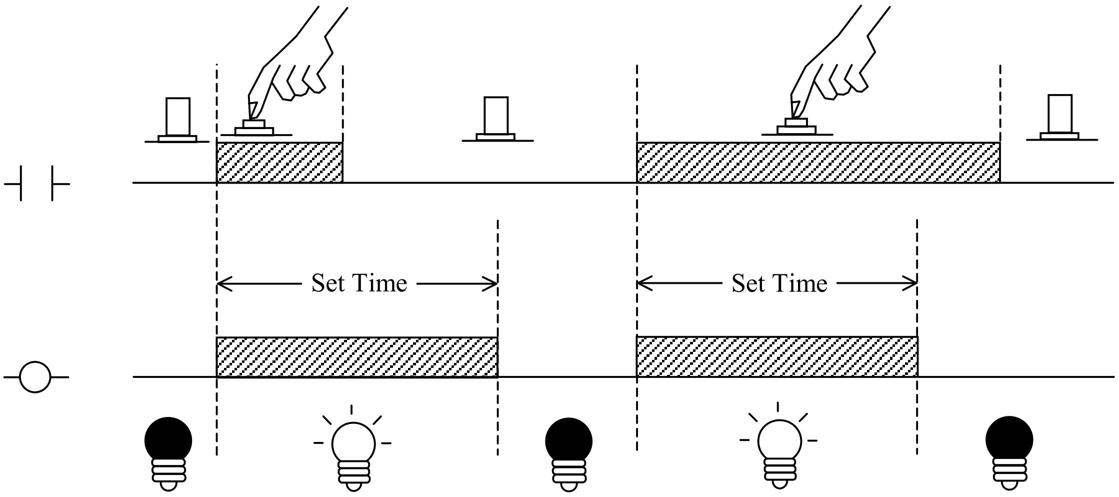
(*) The term "settling pulse width" denotes the minimum time period during which the contacts for turning the one-shot coil ON must remain ON (closed).

<Example of Circuit>



With the circuit shown in the figure, the lamp (Y030) lights only for a set time (2 seconds in this case) if the push button is pushed down.

<Time Chart>



For 2α, the count value is up to 65535. When the counter overflows exceeding 65535, counting restarts beginning with 0.

<Notice>

(1) S10/2α series, S10mini series (Models S, H, F, and D)
The ON/OFF detection process for the one-shot coil is executed at 100-ms intervals, asynchronously with the ladder circuit execution cycle (sequence cycle). If the coil is ON for a period shorter than 100 ms, the one-shot process may not operate due to an unsuccessful coil-ON state detection. To assure proper operation of the one-shot process, prepare the ladder circuit in such a manner that the coil-ON time is 100 ms or longer.

(2) S10/4α series, S10mini (Model C)
The ON/OFF detection process of the one-shot coil is performed when the ladder circuit coil is executed.

C Up-Down Counter

			2α, S10mini, 4α
Range of Numbers	-○-	CU	00 to FF
		CD	
		CR	
	-| |-	CO	
Set Value	0 to 9999 sec.		
Settling (*) Pulse Width	1 scan time, minimum		
When a set and a reset signal are simultaneously input	Reset preferred		
In event of a power failure	Nonvolatile		

The up-down counter is composed of the up counter (CU△△) and the down counter (CD△△). This counter is incremented by 1 (+1) when the coil of the up counter (-○-) rises and is decremented by 1 (-1) when the coil of the down counter rises.

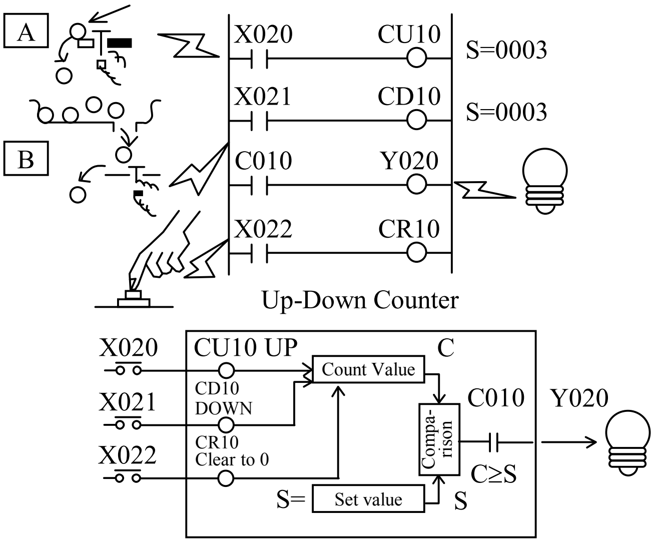
The counter contact (-| |-) is turned ON if the count value exceeds the set value.

The reset coil clears the count value to 0, turning the contact OFF.

CU: Up Counter
CD: Down Counter
CR: Reset
CO: Counter Contact

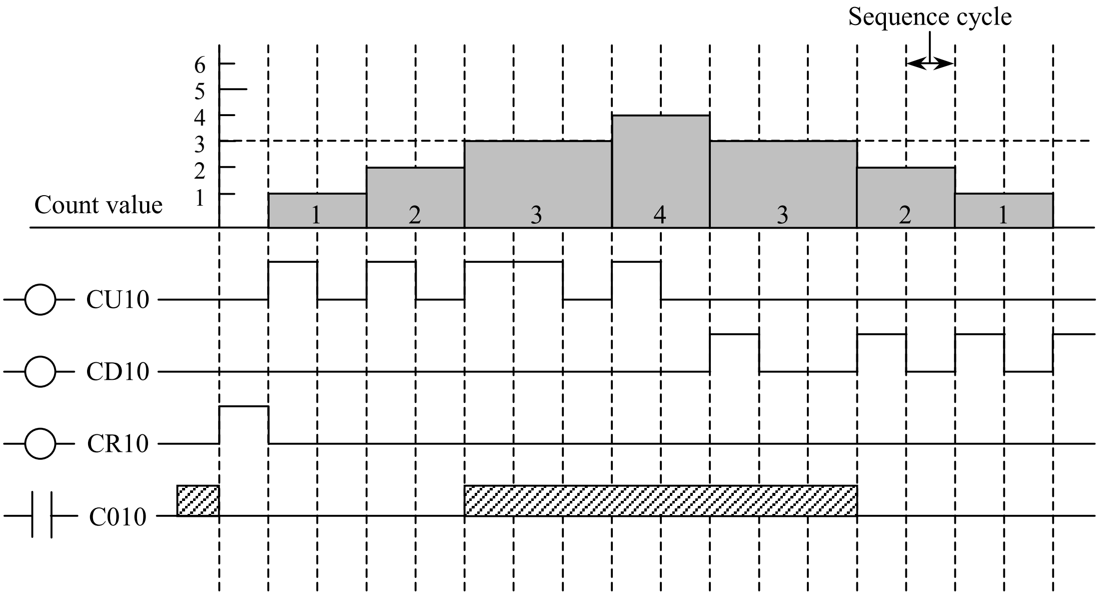
(*) The term “settling pulse width” denotes the minimum time period during which the contacts for turning ON the up counter, down counter, and reset coil must remain ON (closed).

<Example of Circuit>



- The circuit shown in the figure counts the number of the balls falling into a basket by the switch A (X020), counts the number of balls falling down from the basket to the ground by the switch B (X021), and knows the total number of balls staying in the basket.
- If the number of balls in the basket is 3 or more than 3, the lamp (Y020) lights. Pushing the button (X022) extinguishes the lamp, clearing the count value to 0.

<Time Chart>



Note:

- Even if a set value is exceeded, the count value keeps being counted for the up counter. When the count value overflows (exceeds 0xFFFF), it countes from 0 again. At that time, the counter contact point turned on is changed to the state of turning off.
- The count down stops when the count value becomes 0 for the down counter.

G Global Link Register

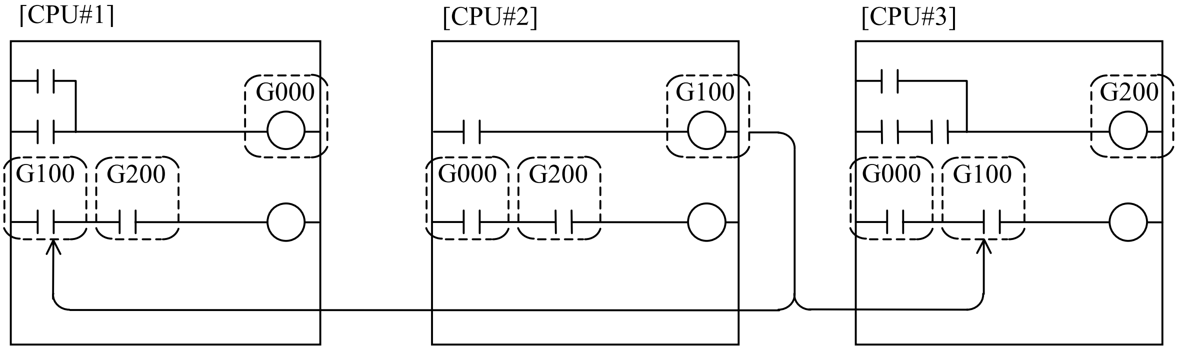
	2α, S10mini, 4α
Range of Numbers	000 to FFF

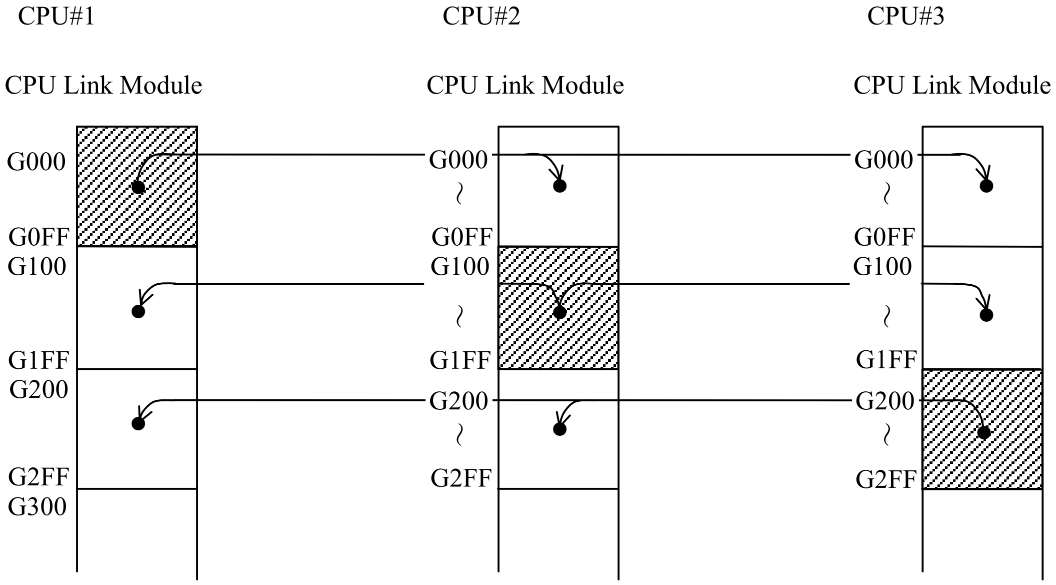
The global link register is a register used for the interlock information exchange between CPUs when an inter-CPU link module (option) is mounted.

If a coil (-○-) is turned ON or OFF, a contact (┤ ├) with the same number is turned ON or OFF on a different CPU.

For the assignment of inter-CPU link, refer to “Software Manual Option CPU Link For Windows® (manual number SAE-3-141).”

<Example of Circuit (When an inter-CPU link card is being used)>





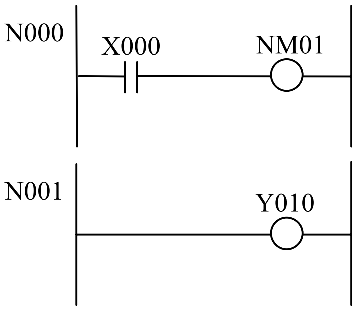
▨ : Range of Inter-CPU Link Assignment

	2α, S10mini	4α
Range of Numbers	001 to OFF	—
Nesting Maximum Level	4 Levels	—

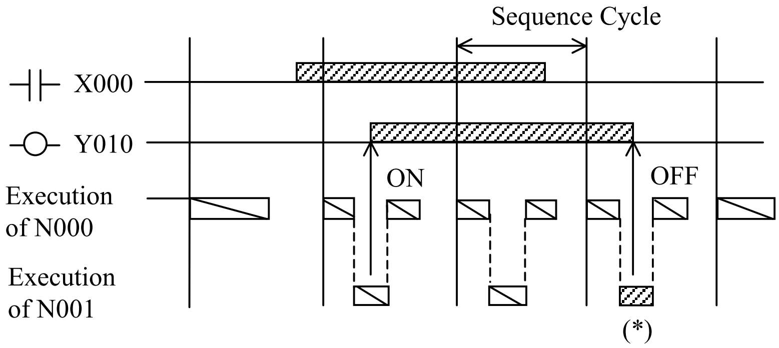
The nesting coil is provided with a function to partition a sequence program into parts for every control object, plant, for modularization. There are two kinds of nesting coil: the master control that turns OFF the coil being used at the falling (ON to OFF) of N coil and the zone control that holds the preceding state.
Nesting is permitted up to 4 levels.

[Master Control (NM)]

<Example of Circuit>



<Time Chart>

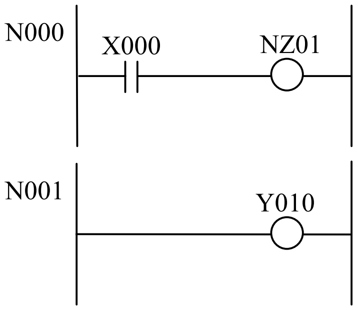


(*) When NM01 falls (ON to OFF), the coils (―○―) used at NM01 are all turned OFF.

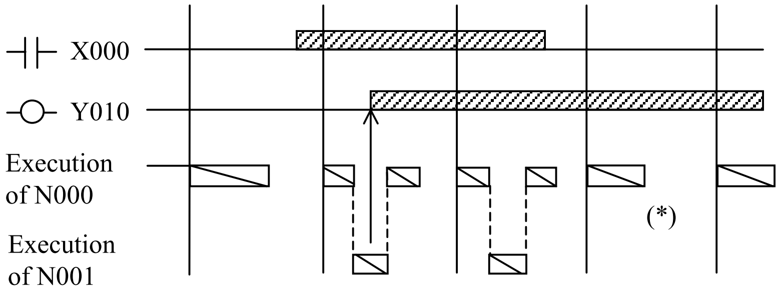
If K (Keep Relay) and ⊣↑⊢ / ⊣↓⊢ edge contacts are used in a circuit, the coils may not be turned OFF.

[Zone Control (NZ)]

<Example of Circuit>

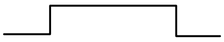


<Time Chart>



(*) Differing from that provided with master reset, a coil used by N001 holds the preceding state, even though NZ01 falls.

P Process Register

	2α, S10mini	4α
Range of Numbers	001 to OFF	—
Start Method	Level Start 	—

The process register is a register used to start a user program written in the C or Assembler language (task, hereinafter) by using a ladder program.
The task corresponding to the number is initiated (queue) by turning this coil ON.

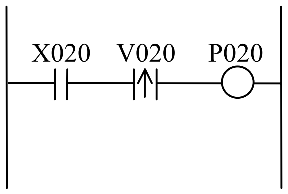
Note: Since a task is not supported on the S10mini model C, it is not started even if this coil is turned ON.

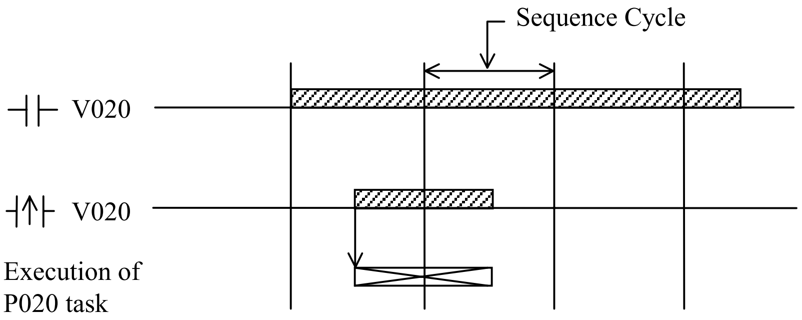
[Assignment of Process Register]

Class	Number	Name	Description
Prepared by User	P001	Initial Task	Initiated whenever the CPU is reset (GR). Assign a program to initialize the system.
	P002 to P07F	User Task	Assign a user program.
Exclusive System Use	P080	System Task	Debugger support tasks are assigned by the CPMS.　Should not be used by the user.

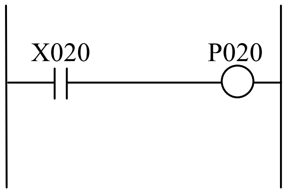
<Example of Circuit>　　<Time Chart>

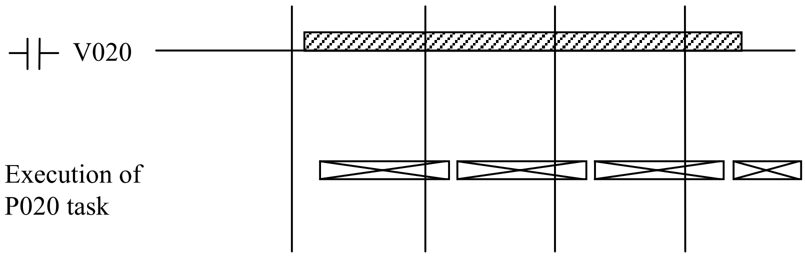
- Executed once when ON





- Executed every time when ON

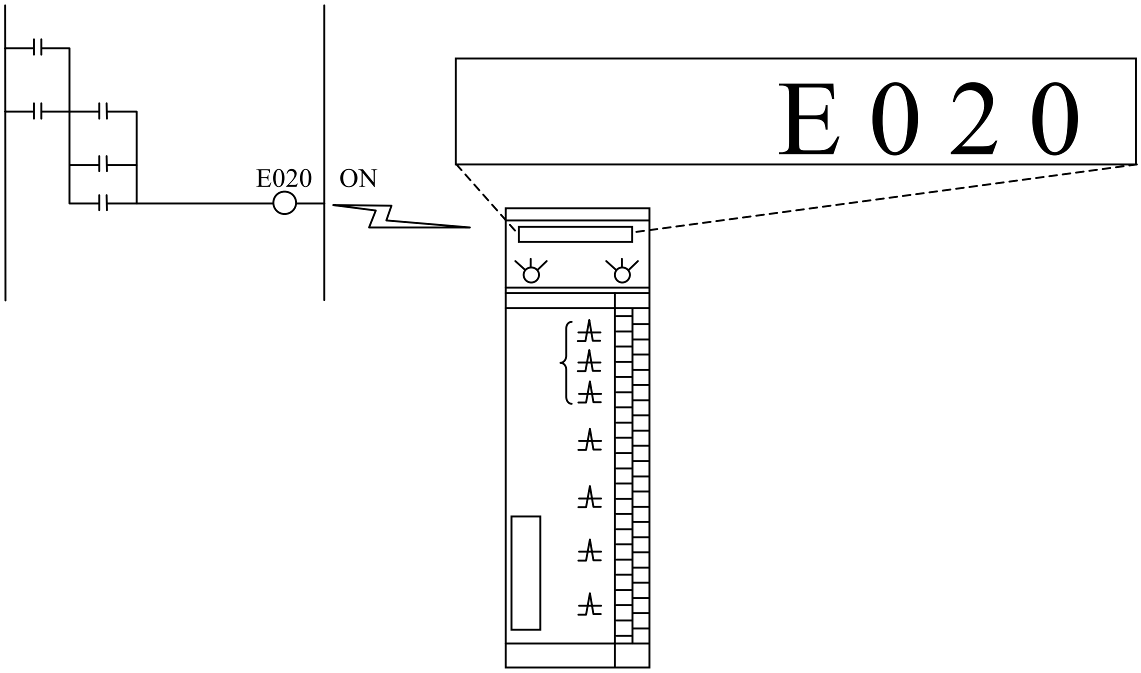




	2α, S10mini	4α
Range of Numbers	000 to FFF	000 to FFF
CPU LED indication range	000 to 1FF	—

The event register is a register used for the display of a user error on the 8-digit LED of the CPU. When the coil of this register is ON, its number is displayed on the LED.
F400 to EFFF are used in 4-channel analog pulse counter.

<Example of Circuit>



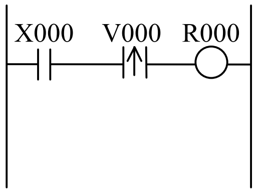
V Edge Contact

	2α, S10mini, 4α
Range of Numbers	000 to FFF

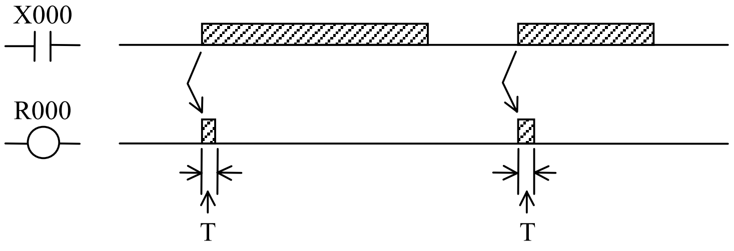
The edge contact includes the rising detection contact (⊣↑⊢) and the falling detection contact (⊣↓⊢). The respective contact is turned ON during the 1 scan that detected respective edge. The same number cannot be used for the rising and falling edges in common.

[Rising Edge Contact]

<Example of Circuit>



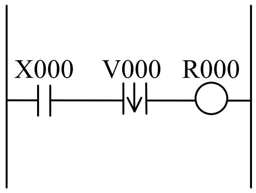
<Time Chart>



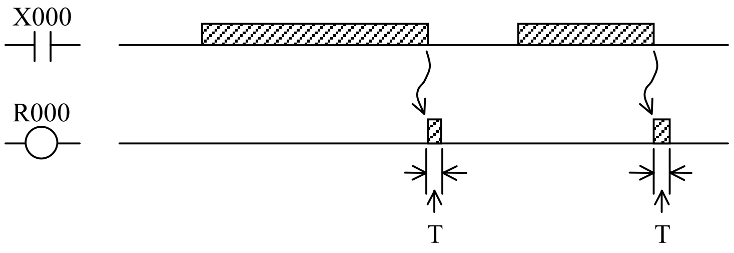
T:1 scan time

[Falling Edge Contact]

<Example of Circuit>



<Time Chart>



T:1 scan time

Range of Numbers	000 to 01F	The I/O information to be traced is set in units of sequence cycles.
	OFE	Start signal of the logic tracer function
	OFF	Stop signal of the logic tracer function

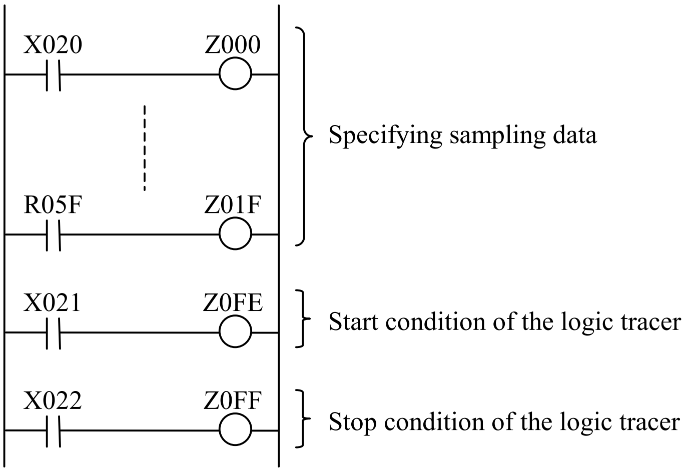
The Z register is assigned the registers for the logic tracer function (not supported in S10mini) and the host interruption, as shown in the given table.

Z200	H-7338 node interrupt is caused in the host computer.

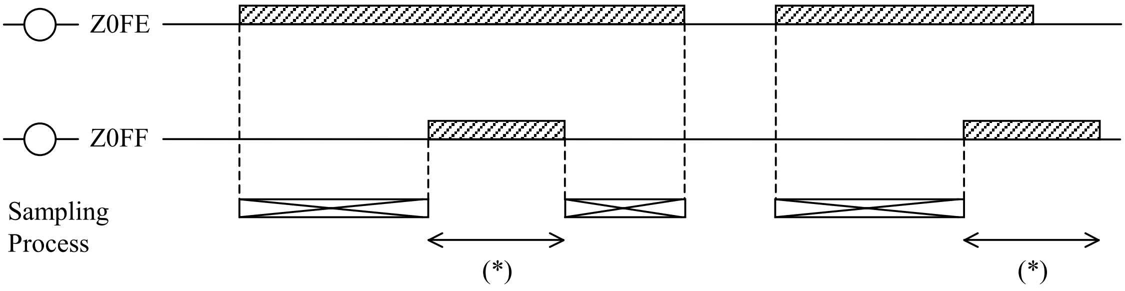
(2α and 4α only.　No interrupt occurs with the S10mini.)

- Logic Tracer and Z Coil

<Example of Circuit>


X020
Z000
R05F
Z01F
X021
Z0FE
X022
Z0FF
Specifying sampling data
Start condition of the logic tracer
Stop condition of the logic tracer

<Time Chart>

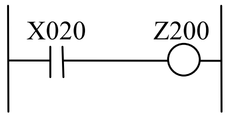


(*) Priority is given to stop.

- Host Interrupt Register (For 2α, 4α only)

[When 2α is selected]

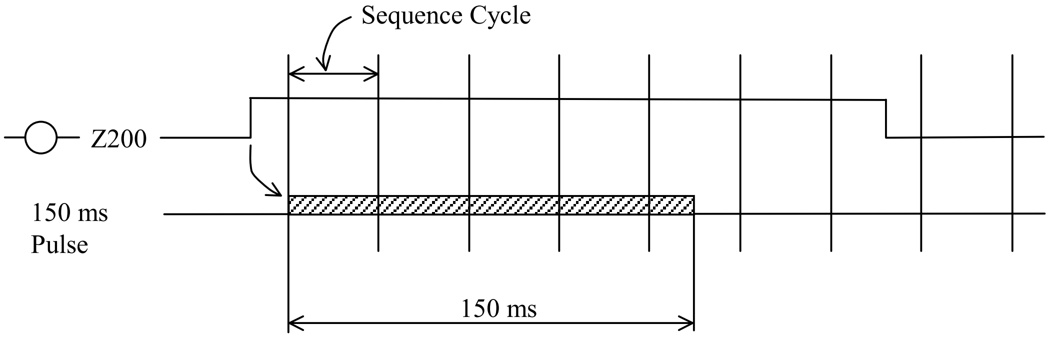
<Example of Circuit>



The host interrupt register causes an interrupt for 150 ms upon detection of the rise of a host interrupt, Z200. This processing is performed synchronizing with a sequence cycle. If another rise of Z200 is detected during the transmission of a 150 ms interrupt, a 150 ms pulse is produced again.

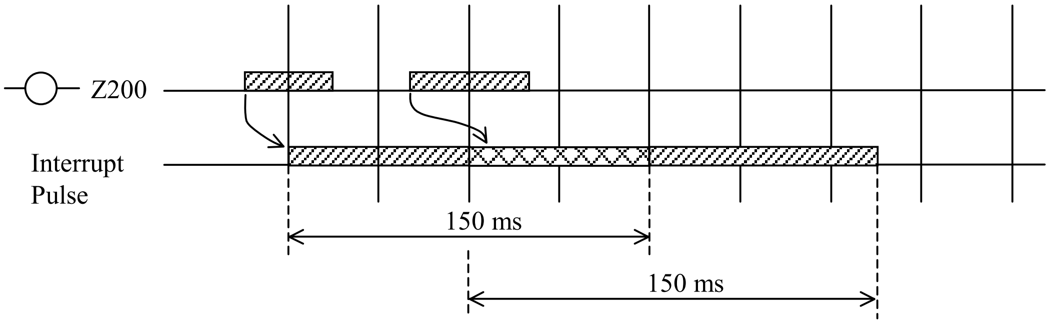
Settling Pulse Width: 1 sequence cycle,
minimum.

<Time Chart>



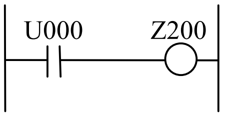
[When Interrupt Pulse becomes Lengthened]

In the above circuit, double interrupt occurs lengthening the pulse width, if ON<->OFF of Z200 occurs two or more times within 150 ms.

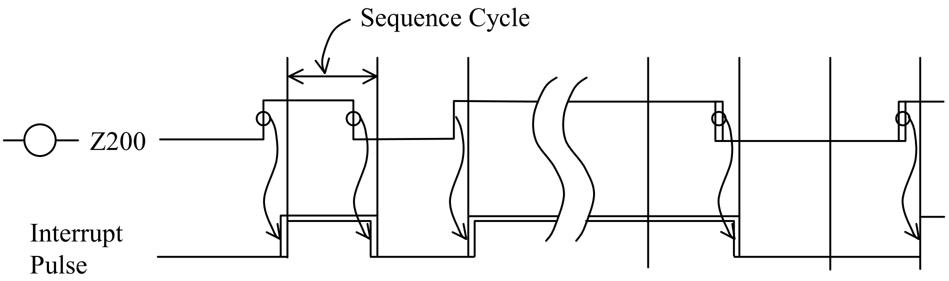


To prevent a happening of this kind, contrive a means of interlocking with the host computer that accepts the interrupt.

[When 4α is selected]
<Example of Circuit>



An interrupt signal is generated during the interval between rising edge detection and falling edge detection of the interrupt Z200 sent to the host. Note that this process is executed synchronously with the sequence cycle. (The output of the one-shot process is usually used.)



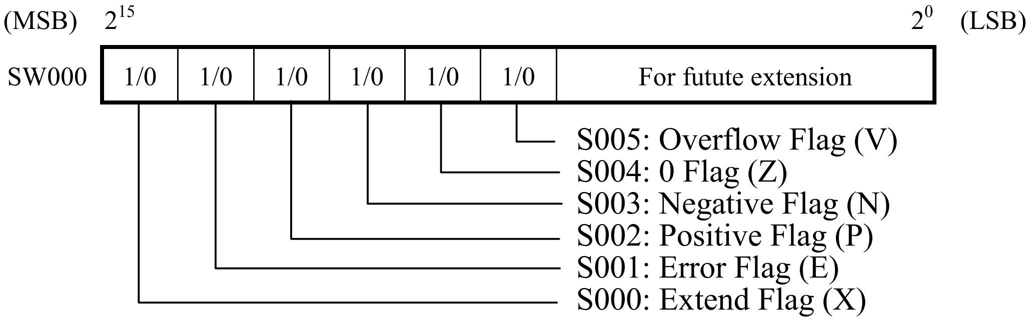
This register is a read-only register to reflect the state of the system operations and so on.

[System Register List]

No.	Register Number	Description
1	S000 to S00F	Arithmetic Function Flag Register
2	S010 to S01F	Ladder Program Control Register
3	S100 to S15F	Ladder Program Control Counter
4	S300 to S47F	Remote I/O Status Register
5	S500 to S6FF	Option Module Status Register
6	SBA0 to SBBF	I/O Fixed Pattern
7	SBF0 to SBFF	CPU Status Register

Registers other than the above are reserved for future extension.

(1) Arithmetic Function Flag Register
Register to show the flag state after the execution of a system arithmetic instruction. Each number corresponds to the state of a flag as shown below:


(MSB) 2^{15} 2^{0} (LSB)

SW000	1/0	1/0	1/0	1/0	1/0	1/0	For futute extension

S005: Overflow Flag (V)
S004: 0 Flag (Z)
S003: Negative Flag (N)
S002: Positive Flag (P)
S001: Error Flag (E)
S000: Extend Flag (X)

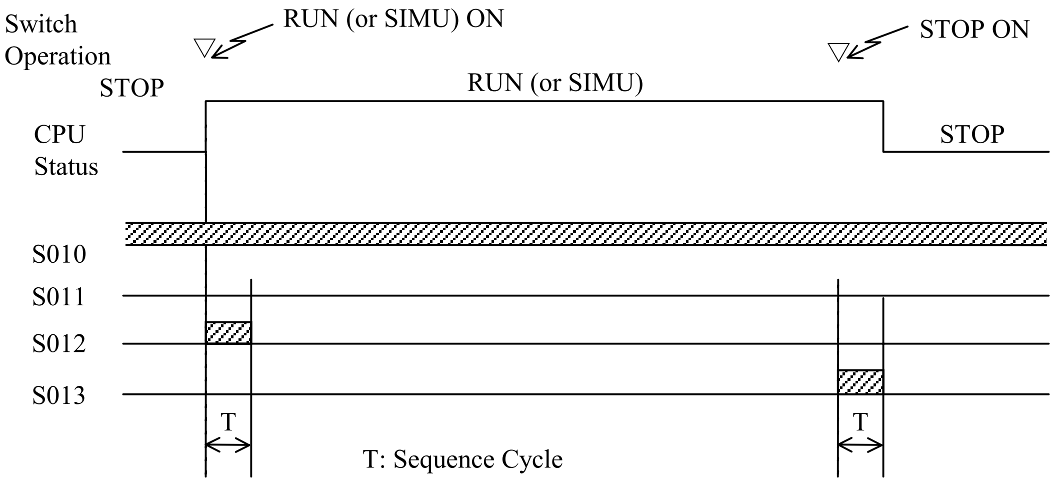
(2) Ladder Program Control Register

Register to store the information that facilitates the preparation of a ladder program.

	(MSB) 2^{15}				2^{0} (LSB)
SW010	0	1	1/0	1/0	Not used

S013: RUN→STOP 1 sequence cycle pulse
S012: STOP→RUN 1 sequence cycle pulse
S011: Normally-OFF flag
S010: Normally-ON flag

<Time Chart>



(3) Ladder Program Control Counter

Integrating counter usable in sequence control.

	(MSB) 2^{15} 2^{0} (LSB)	
SW100	10 ms Integrating Counter	Cleared to 0 at CPU power recovery and reset.
SW110	100 ms Integrating Counter	
SW120	1-second Integrating Counter	
SW140	Sequence Cycle Integrating Counter	··· Cleared to 0 at STOP→RUN of CPU.
SW150	Remote I/O Cycle Integrating Counter	··· Cleared to 0 at STOP→RUN of remote I/O.

- All the counters start counting beginning with 0 if an overflow occurs.
- Errors within ±10% are unavoidable as the accuracy of the counters is maintained by the OS (Operating System) interrupt.

(4) Remote I/O Status Register

Register to show the remote I/O registration status and the information on time out or fuse-broken stations.

[Assignment of Registers]

S300 ~	Registered stations	The registers associated with the currently registered stations are set to “1”.
S380 ~	Time out stations	The registers associated with the time out stations are set to “1”.
S400 ~ S47F	Fuse- blown stations	The registers associated with the fuse-blown stations are set to “1”.

[Correspondence Between Station and Bit]

No.	X or Y Number	Registered Station	Time 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

- “1” is set to S400 if the fuse of the output module in the 4α CPU base has blown out.
- For 2α, S10mini, the “1” flag stated in the above is set in the LED display timing (once per 2 seconds).

(5) Option Module Status Register

Register to store the error information on the inter-CPU link, external units link, etc. Each module is assigned as shown in the following. For detail of bit configuration, refer to the manual supplied at the purchase of the extension module, as data setting to this register is made by the Support Sub-OS of each module.

[Assignment of Extension Module]

S500	Reserved for the system	
S580		··· Information on inter-CPU link module
S5C0		··· Information on PSE link module (For 2α only)
S600		··· Information on external unit link module (For 2α, S10mini (*) only)
S640		··· Information on high-speed remote I/O module (For 2α only)
S680	Reserved for the system	
S6C0		··· Information on PCs I/O link module (For 4α only)

The above registers are cleared to 0 at CPU GR (when the power is recovered) (The same holds true with the 2α and S10mini even when a reset is performed).

(*) External device link modules for the S10mini model C will not be supported.

(6) I/O Fixed Pattern (For 2α, S10mini only)

An area storing the fixed pattern used by the OS for the initial check of the I/O memory. If the pattern in this area is destroyed, the system clears whole I/O memory (including the arithmetic instruction work area) to 0.

[I/O Fixed Pattern]

(MSB)	2^{15}															2^0	(LSB)
SWBA0	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0	(=H55AA)
SWBB0	1	1	1	0	1	1	1	0	1	0	0	1	1	0	0	1	(=HEE99)

(7) CPU Status Register

Register to show the current CPU status. The bit configuration of the register is shown in the following. Users can see the status of the CPU by reading this area using PSE, etc.

[Bit Configuration of CPU Status Register]

(MSB)	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	(LSB)
SWBF0	1/0	1/0	1/0	1/0	1/0	*	*	*	1/0	1/0	1/0	1/0	*	1/0	1/0	1/0	
Bit No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

*: For future extension

Bit No.	Relay	Description of Each Bit		
		"1" ON	"0" OFF	
0	SBF0	In "STOP"	In "RUN"	
1	SBF1	In simulation	In normal RUN	**
2	SBF2	In execution in FORCE mode	In execution in normal mode	**
3	SBF3	In protect SW "ON" state	In protect SW "OFF" state	**
4	SBF4	In remote I/O operation	Remote I/O is stopped.	**
5	SBF5	Not used		
6	SBF6			
7	SBF7			
8	SBF8	Basic memory battery error alarm	Normal	
9	SBF9	Timeout error station exists.	None	
10	SBFA	Fuse-blown station is found.	None	
11	SBFB	Option module error is found.	None	
12	SBFC	For future extension		
13	SBFD	Clear to 0 only at GR		
14	SBFE	CPU-down is shown.	CPU in normal operation	**
15	SBFF	In execution of IPL program	CPU in execution of OS	**

**: For 2α, S10mini only

THIS PAGE INTENTIONALLY LEFT BLANK.

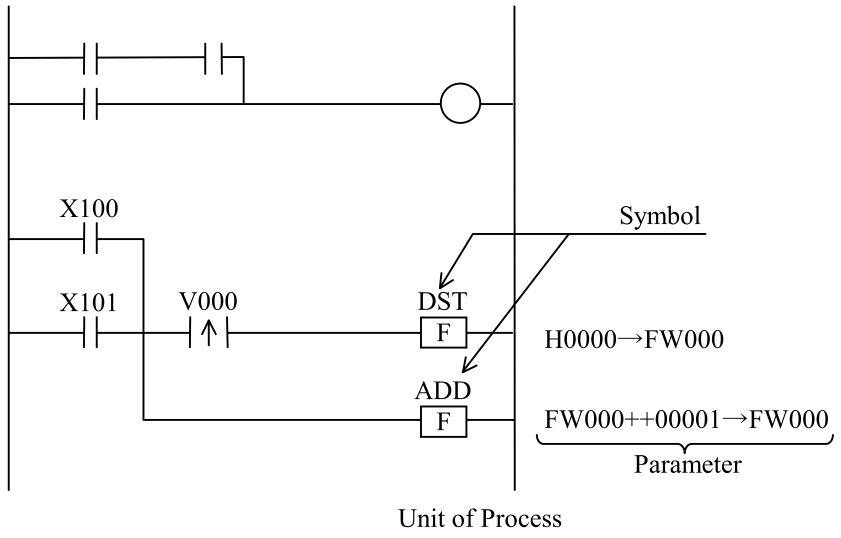
2 ARITHMETIC FUNCTION

2.1 Functional Outline

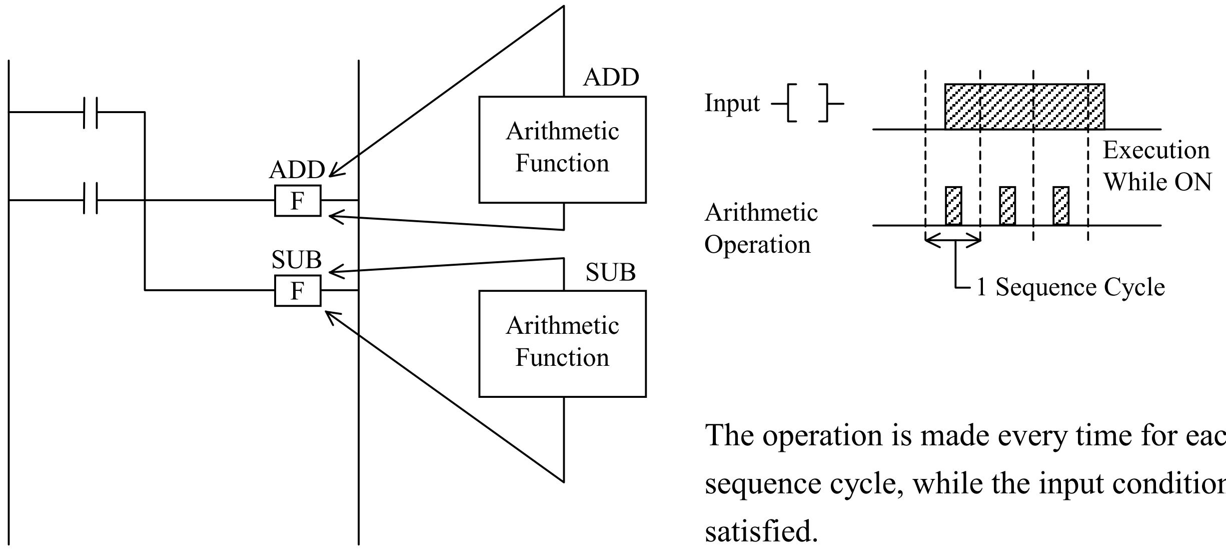
To perform a simple arithmetic, the user can prepare the program by using the arithmetic function.

[Operation of Arithmetic Function]

<Example of Circuit>



<Operation>



The operation is made every time for each sequence cycle, while the input condition is satisfied.

(1) Parameter

The arithmetic function has the symbols assigned according to the contents of the processes, and each symbol has the parameter of its own.

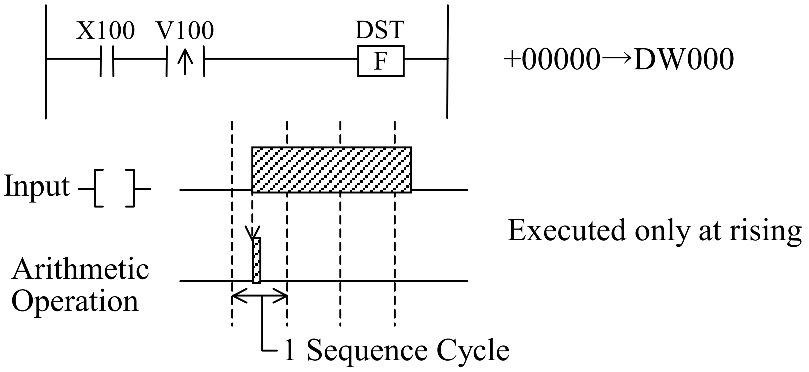
The constant data register and the work register used by the arithmetic function are shown in the following table.

Name	Symbol	Number		Use
		2α, 10mini	4α	
Function Data Register	DW	000 to FFF	000 to 7FF	Constant Data Area
Function Work Register	FW	000 to BFF		Work Area

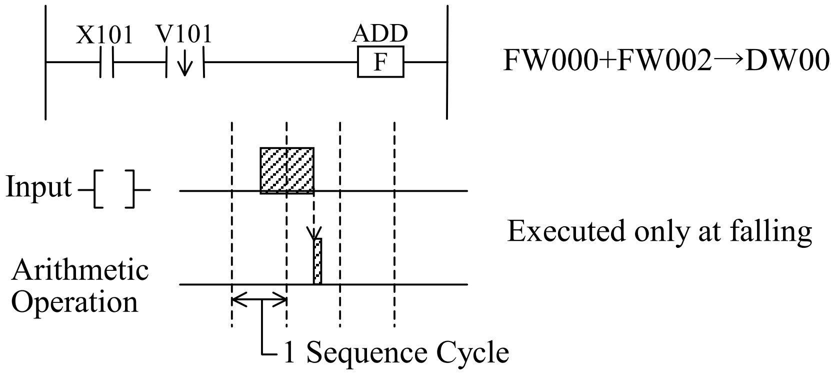
(2) Operation

The arithmetic function is initiated every time while the coil exciting signal is ON. To initiate the function only at the rising or falling edge of the exciting signal, combine the function with the rising or falling contact.

(Example 1) Combination with Rising Contact

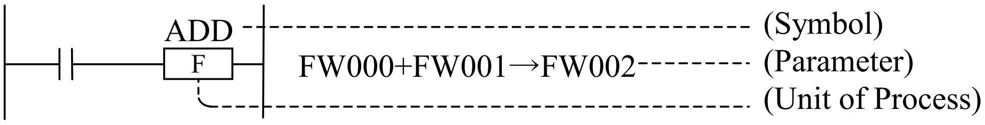


(Example 2) Combination with Falling Contact



2.2 Functional Specification

(1) Symbol



Symbol: Means the function name of an arithmetic function.

Parameter: Means the address and constant data used in the arithmetic.

Unit of Process: Means the unit of arithmetic operation.

(W: Word
L: Long Word)

(2) Data Format

[Word (XW000, YW000, RW000, ···FW000, DW000,
X000, Y000, R000, TC000, UC000, CC000, TS000)]

	MSB															LSB
Bit Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Note 1: X000, R123, etc. are treated as a word.

Note 2: TC000, UC000, CC000, TS000, US000, and CS000 are treated as a word size.

[Long Word]

	MSB																						LSB
Bit Number	0	1	2	3	4	5	6	7	8	9	10	11		22	23	24	25	26	27	28	29	30	31
													• • •										
	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}		2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

All the data used in the arithmetic function are signed 16-bit single precision integers or signed 32-bit double precision integers. Data represented in 16 bits is called a word data, while data represented in 32 bits is called a long word data, and each bit is assigned a bit number as shown above.

The bit, the bit number of which is “0”, is called the sign bit. This bit being 0 means a positive number, while being 1 means a negative number. The range of the numbers processed by the arithmetic function is given in the following:

Word:

-32768 ≤ Word Data ≤ +32767 (Decimal Number)
(H8000) (H7FFF) (Hexadecimal Number: Indicated by H)

Long Word:

-2147483648 ≤ Long Word Data ≤ +2147483647 (Decimal Number)
(H80000000) (H7FFFFFFF)
(Hexadecimal Number: Indicated by H)

(3) Flag Setting

The arithmetic function sets various flags according to the result of the arithmetic operations. The following table contains the kinds of flags, flag areas, and conditions of flag setting.

X: Extend	S000	X
E: Error	S001	E
P: Positive	S002	P
N: Negative	S003	N
Z: Zero	S004	Z
V: Overflow	S005	V
F.U.: Future Use	S006	F.U.
	S007	F.U.

No.	Class	Flag						Specific Condition	
		X	E	P	N	Z	V	When Word:	When Long Word:
1	ADD SUB IVC DEC MUL	-	-	-	-	-	↕	V: When result<-32768 or 32767<result: 1 Otherwise: 0	V: When result<-2147483648 or 2147483647<result: 1 Otherwise: 0
2	DIV MOD	-	↕	-	-	-	↕	E: When divisor=0: 1 Otherwise: 0 V: When Quotient=32768: 1 Otherwise: 0	E: When divisor=0: 1 Otherwise: 0 V: When Quotient=2147483648: 1 Otherwise: 0
3	SCL	-	↕	-	-	-	↕	E: When divisor=0: 1 Otherwise: 0 V: When result<-32768 or 32767<result:1 Otherwise: 0	
4	TST	-	-	↕	↕	↕	-	P: When data>0: 1 Otherwise: 0 N: When data<0: 1 Otherwise: 0 Z: When data>0: 1 Otherwise: 0	
5	BTD	-	↕	-	-	-	↕	E: When data<0: 1 Otherwise: 0 V: When data>9999: 1 Otherwise: 0	E: When data<0: 1 Otherwise: 0 V: When data>99999999: 1 Otherwise: 0
6	DTB	-	↕	-	-	-	-	E: When HA to HF is detected at a certain digit(4 bits): 1 Otherwise: 0	
7	APB AUB	-	↕	-	-	-	-	E: When data other than H30 to H39 and H41 to H46 is detected: 1 Otherwise: 0	
8	DTS	-	-	-	-	-	↕		V: When data<-32768 or 32767< data: 1 Otherwise: 0
9	ABS NEG	-	-	-	-	-	↕	V: When data=32768: 1 Otherwise: 0	V: When data=-2147483648: 1 Otherwise: 0
10	ECD	-	↕	-	-	-	-	E: When data=0: 1 Otherwise: 0	
11	ASL	-	-	-	-	-	↕	V: When the sign bit changed during shift operation: 1 Otherwise: 0	
12	LIM	-	↕	-	-	-	-	E: When upper limit<lower limit: 1 Otherwise: 0	
13	BND ZON	-	↕	-	-	-	↕	E: When upper limit <lower limit: 1 Otherwise: 0 V: When result<-32768 or 32767< result: 1 Otherwise: 0	E: When upper limit <lower limit: 1 Otherwise: 0 V: When result <-2147483648 or 2147483647 < result: 1 Otherwise: 0
14	Other than the above	-	-	-	-	-	-	Held in any case	

- : The value immediately before the execution of the arithmetic operation is held.

↕: Refer to a specific condition.

2.3 Input to an Arithmetic Function

Input to an arithmetic function is accomplished using an arithmetic function input diagram. The general form of function inputs is as follows:

symbol␣parameter1, parameter2, parameter3 [Enter]

As shown above, a blank space must be inserted between the symbol and parameter1 and each parameter must be separated a comma from the next one.

(1) Entering Mnemonics

Setting Permitted Area	Example of Input	Remarks
I/O Area (Bit)	X000	Specifying a bit.
I/O Area (Word)	YW000	W means a word.
Function Work Register Area	FW025	Work Area
Function Data Register Area	DW050	Constant Data Area
T, U, and C Set Value Area	TS003	S means a set value.
T, U, and C Count Value Area	UC007	C means a count value.
High-Speed I/O (Word) Area	IW000	

- I/O Area: X, Y, R, K, T, U, C, G, N, P, E, V, Z, and S
- A number is entered in 3 digits.
- If an I/O area having multiple areas (T, U, C set coil and contact areas, N set coil and contact areas, and so on) is set, an access is made to the contact area.
  Example: TW010 means the contact area of T010 to T01F.

(2) Entering Data

(a) Entering a Decimal Number

(i) Enter a number directly.
(Positive decimal number)

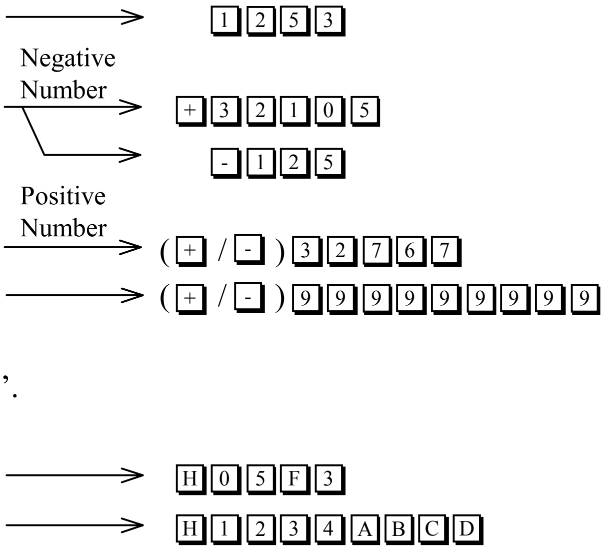
(ii) Enter the number after
entering +/-.

(iii) Number of Input Digits
Word: 5 digits, maximum.
Long Word: 9 digits, maximum.

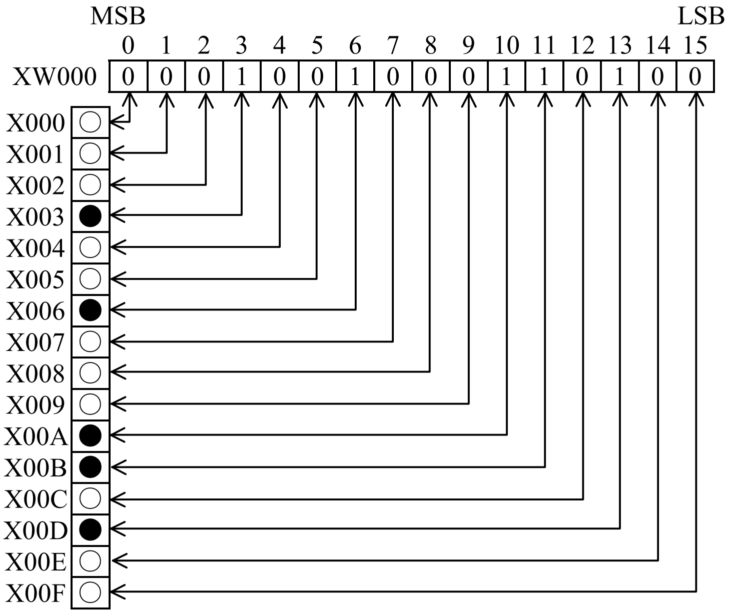
(b) Entering a hexadecimal Number

(i) Enter the number after entering ‘H’.

(ii) Number of Input Digits
Word: 4 digits, maximum.
Long Word: 8 digits, maximum.



(3) Bit and Word Areas of I/O Area



(●: ON
○: OFF)

These areas correspond to each other as shown below:

XW000 ⇔ X000 to X00F
XW010 ⇔ X010 to X10F
⋮ ⋮
XW100 ⇔ X100 to X10F
⋮ ⋮

2.4 Arithmetic Function List

(1/7)

Major Class	Class	Parameter Format	Contents of Process	Unit of Process	Flag						Number of steps	Time of Process (ms)			Page
					X	E	P	N	Z	V		2α	2αE, 2αH(f)	4α	
Arithmetic Operation Instruction	Addition	Word	(S)+(D)→(R)	Word	-	-	-	-	-	⇕	4	0.06	0.036	0.6	44
		Word Constant	(S)+n→(R)	Word							4	0.06	0.036	0.5	
		Long Word	(S)L+(D)L→(R)L	Long Word							4	0.06	0.036	0.8	
	Subtraction	Word	(S)-(D)→(R)	Word	-	-	-	-	-	⇕	4	0.06	0.036	0.6	46
		Word Constant	(S)-n→(R)	Word							4	0.06	0.036	0.5	
		Long Word	(S)L-(D)L→(R)L	Long Word							4	0.06	0.036	0.8	
	+1	Word	(S)+1→(S)	Word	-	-	-	-	-	⇕	2	0.04	0.024	0.5	48
		Long Word	(S)L+1→(S)L	Long Word							2	0.04	0.024	0.6	
	-1	Word	(S)-1→(S)	Word	-	-	-	-	-	⇕	2	0.04	0.024	0.5	50
		Long Word	(S)L-1→(S)L	Long Word							2	0.04	0.024	0.6	
	Multiplication	Word	(S)×(D)→(R)	Word	-	-	-	-	-	⇕	4	0.08	0.048	0.9	52
		Word Constant	(S)×n→(R)	Word							4	0.08	0.048	0.8	
		Long Word	(S)L×(D)L→(R)L	Long Word							4	0.13	0.078	2.3	
	Division	Word	(S)÷(D)→(R) (Quotient)	Word	-	⇕	-	-	-	⇕	4	0.09	0.054	1.0	54
		Word Constant	(S)÷n→(R) (Quotient)	Word							4	0.09	0.054	0.9	
		Long Word	(S)L÷(D)L→(R)L (Quotient)	Long Word							4	0.38	0.228	3.0	
	Residue Operation	Word	(S)%(D)→(R) (Remainder)	Word	-	⇕	-	-	-	⇕	4	0.11	0.066	1.0	56
		Word Constant	(S)%n→(R) (Remainder)	Word							4	0.11	0.066	0.9	
		Long Word	(S)L%(D)L→(R)L (Remainder)	Long Word							4	0.46	0.276	3.0	
	Scale Conversion	Word	(S)×(D1)÷(D2)→(R)	Word	-	⇕	-	-	-	⇕	5	0.12	0.072	3.1	58
		Word Constant	(S)×n1÷n2→(R)	Word							5	0.11	0.066	2.9	
Logical Operation Instruction	AND	Word	(S)∧(D)→(R)	Word	-	-	-	-	-	-	4	0.05	0.03	0.6	60
		Word Constant	(S)∧n→(R)	Word							4	0.05	0.03	0.5	
		Long Word	(S)L∧(D)L→(R)L	Long Word							4	0.05	0.03	0.8	
	OR	Word	(S)∨(D)→(R)	Word	-	-	-	-	-	-	4	0.05	0.03	0.6	62
		Word Constant	(S)∨n→(R)	Word							4	0.05	0.03	0.5	
		Long Word	(S)L∨(D)L→(R)L	Long Word							4	0.05	0.03	0.8	
	Exclusive OR	Word	(S)⊕(D)→(R)	Word	-	-	-	-	-	-	4	0.05	0.03	0.6	64
		Word Constant	(S)⊕n→(R)	Word							4	0.05	0.03	0.5	
		Long Word	(S)L⊕(D)L→(R)L	Long Word							4	0.05	0.03	0.8	
	NOT	Word	$\overline{(S)}$→(R)	Word	-	-	-	-	-	-	3	0.04	0.024	0.4	66
		Long Word	$\overline{(S)L}$→(R)L	Long Word							3	0.04	0.024	0.6	
Compare Operation Instruction	=	Word	When (S)=(D): 1→(R) When (S) (D): 0→(R)	Word	-	-	-	-	-	-	4	0.05	0.3	0.6	68
		Word Constant	When (S)=(n): 1→(R) When (S) (n): 0→(R)	Word							4	0.05	0.3	0.5	
		Long Word	When (S)L=(D)L: 1→(R) When (S)L (D)L: 0→(R)	Long Word							4	0.06	0.036	0.7	
	≠	Word	When (S)=(D): 0→(R) When (S) (D): 1→(R)	Word	-	-	-	-	-	-	4	0.05	0.03	0.6	70
		Word Constant	When (S)=n: 0→(R) When (S) n: 1→(R)	Word							4	0.05	0.03	0.5	
		Long Word	When (S)L=(D)L: 0→(R) When (S)L (D)L: 1→(R)	Long Word							4	0.06	0.036	0.7	
	>	Word	When (S) >(D): 1→(R) When (S) ≤(D): 0→(R)	Word	-	-	-	-	-	-	4	0.05	0.03	0.06	72
		Word Constant	When (S) >n: 1→(R) When (S) ≤n: 0→(R)	Word							4	0.05	0.03	0.5	
		Long Word	When (S)L >(D)L: 1→(R) When (S)L ≤(D)L: 0→(R)	Long Word							4	0.06	0.036	0.7	

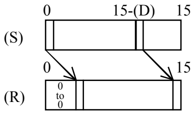
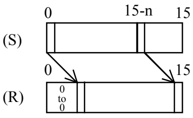
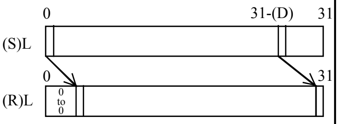
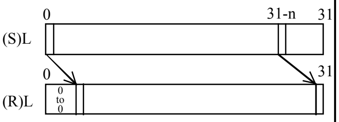
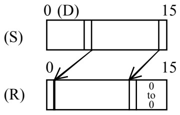
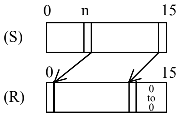
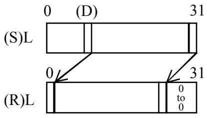
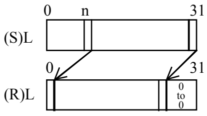
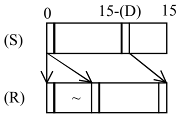
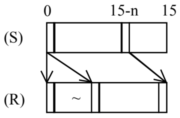
(2/7)

Major Class	Class	Parameter Format	Contents of Process	Unit of Process	Flag						Number of steps	Time of Process (ms)			Page
					X	E	P	N	Z	V		2α	2αE, 2αH(f)	4α	
Compare Operation Instruction	≥	Word	When (S)≥(D): 1→(R) When (S)<(D): 0→(R)	Word	-	-	-	-	-	-	4	0.05	0.03	0.6	74
		Word Constant	When (S)≥n: 1→(R) When (S)<n: 0→(R)	Word							4	0.05	0.03	0.5	
		Long Word	When (S)L≥(D)L: 1→(R) When (S)L<(D)L: 0→(R)	Long Word							4	0.06	0.036	0.7	
	<	Word	When (S)<(D): 1→(R) When (S)≥(D): 0→(R)	Word	-	-	-	-	-	-	4	0.05	0.03	0.6	76
		Word Constant	When (S)<n: 1→(R) When (S)≥n: 0→(R)	Word							4	0.05	0.03	0.5	
		Long Word	When (S)L<(D)L: 1→(R) When (S)L≥(D)L: 0→(R)	Long Word							4	0.06	0.036	0.7	
	≤	Word	When (S)≤(D): 1→(R) When (S)>(D): 0→(R)	Word	-	-	-	-	-	-	4	0.05	0.03	0.6	78
		Word Constant	When (S)≤n: 1→(R) When (S)>n: 0→(R)	Word							4	0.05	0.03	0.5	
		Long Word	When (S)L≤(D)L: 1→(R) When (S)L>(D)L: 0→(R)	Long Word							4	0.06	0.036	0.7	
	Test	Word	Tests (S), setting P, N, and Z flags.	Word	-	-	⇕	⇕	⇕	-	2	0.04	0.024	0.4	80
		Long Word	Tests (S)L, setting P, N, and Z flags.	Long Word							2	0.04	0.024	0.5	
Data Move Instruction	Move	Word	(S)→(D)	Word	-	-	-	-	-	-	3	0.04	0.024	0.4	82
		Long Word	(S)L→(D)L	Word							3	0.04	0.024	0.6	
	Block Move	Word	(S) or (S)L → (D) or (D)L n	Word	-	-	-	-	-	-	4	0.06+ 0.007n	0.036 0.005n	0.3+ 0.4n	84
		Long Word		Long Word							4	0.06+ 0.008n	0.036 0.005n	0.3+ 0.4n	
	Indirect Move	Word	i(S) → (D) n	Word	-	-	-	-	-	-	4	0.05+ 0.007n	0.036 0.005n	0.6+ 0.3n	86
		Word	(S) → i(D) n	Word							4	0.05+ 0.007n	0.036 0.005n	0.6+ 0.3n	88
	Same Data Block Move	Word	(S) or n or (S)L → (D) or (D)L n	Word	-	-	-	-	-	-	4	0.05+ 0.004n	0.03 0.003n	0.3+ 0.2n	90
		Word Constant		Word							4	0.05+ 0.004n	0.03 0.003n	0.3+ 0.2n	
		Long Word		Long Word							4	0.06+ 0.004n	0.036 0.003n	0.4+ 0.2n	
	Exchange	Word	(S)⇔(D)	Word	-	-	-	-	-	-	3	0.04	0.024	0.7	92
		Long Word	(S)L⇔(D)L	Long Word							3	0.04	0.024	1.0	
	FIFO Write	Word	FIFO (S)→Table	Word	-	-	-	-	-	-	3	0.06	0.036	1.2	94
	FIFO Read	Word	FIFO Table →(D)	Word	-	-	-	-	-	-	3	0.07+ 0.006n (is the pointer)	0.042 0.004n	1.3+ 0.2n	96
	Data Set	Word Constant	n→(D)	Word	-	-	-	-	-	-	3	0.04	0.024	0.3	98
		Long Word	nL→(D)L	Long Word							4	0.05	0.03	0.4	
	Address Set	Long Word	S(Address Data)→(D)L	Long Word	-	-	-	-	-	-	3	0.04	0.024	0.4	100
	Search	Word	S D m Matched Number n→(R)	Word	-	-	-	-	-	-	5	0.07+ 0.004m	0.042 0.003m	0.5+ 0.1m	102
		Long Word		Long Word							5	0.08+ 0.004m	0.048 0.003m	0.5+ 0.2m	

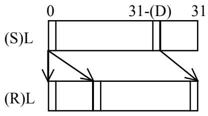
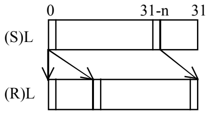
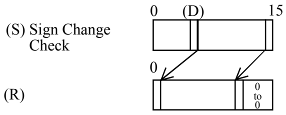
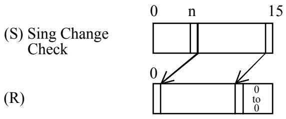
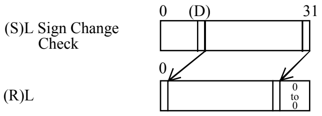
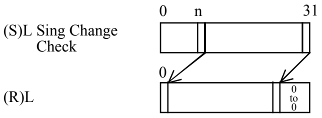
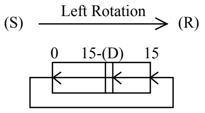
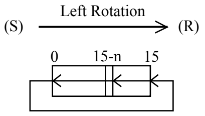
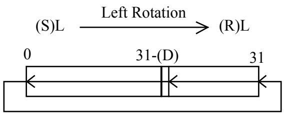
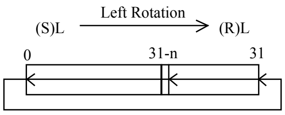
(3/7)

Major Class	Class	Parameter Format	Contents of Process	Unit of Process	Flag						Number of steps	Time of Process (ms)			Page
					X	E	P	N	Z	V		2α	2αE, 2αH(f)	4α	
Data Conversion Instruction	BIN ↓ BCD	Word	BIN→BCD (S) ⟶ (R)	Word	-	⇕	-	-	-	⇕	3	0.26	0.156	1.6	104
		Long Word	BIN→BCD (S)L ⟶ (R)L	Long Word							3	0.51	0.306	9.8	
	BCD ↓ BIN	Word	BCD→BIN (S) ⟶ (R)	Word	-	⇕	-	-	-	-	3	0.21	0.126	1.3	106
		Long Word	BCD→BIN (S)L ⟶ (R)L	Long Word							3	0.37	0.222	7.1	
	BIN ↓ 7SEG	Word	BIN→7seg (S) ⟶ (R, R+1)	Word	-	-	-	-	-	-	3	0.11	0.066	0.6	108
		Word	BIN→7seg n ⟶ (R, R+1)	Long Word							3	0.21	0.126	0.5	
		Long Word	BIN→7seg (S)L ⟶ {R(L), R+2(L)}	Word							3	0.11	0.066	0.9	
	BIN ↓ ASCII	Word	BIN→ASCII (Packed Mode) (S) ⟶ (R, R+1)	Word	-	-	-	-	-	-	3	0.13	0.078	0.6	110
		Word	BIN→ASCII (Unpacked Mode) (S) ⟶ (R, R+1, R+2, R+3)	Word	-	-	-	-	-	-	3	0.14	0.084	0.8	112
	ASCII ↓ BIN	Word	ASCII→BIN (Packed Mode) (S, S+1) ⟶ (R)	Word	-	⇕	-	-	-	-	3	0.13	0.078	0.6	114
		Word	ASCII→BIN (Unpacked Mode) (S, S+1, S+2, S+3) ⟶ (R)	Word	-	⇕	-	-	-	-	3	0.13	0.078	0.8	116
	SINGLE ↓ DOUBLE	Word	(S)→(R)L	Word	-	-	-	-	-	-	3	0.04	0.024	0.5	118
	DOUBLE ↓ SINGLE	Long Word	(S)L→(R)	Long Word	-	-	-	-	-	⇕	3	0.05	0.03	0.6	120
	Absolute value	Word	| (S) |→(R)L	Word	-	-	-	-	-	⇕	3	0.05	0.03	0.5	122
		Long Word	| (S)L |→(R)	Long Word							3	0.05	0.03	0.6	
	+/-	Word	-(S)→(R)	Word	-	-	-	-	-	⇕	3	0.05	0.03	0.5	124
		Long Word	-(S)L→(R)L	Long Word							3	0.05	0.03	0.6	
	Decode	Word	(S) (R)0 n 15(LSB) n → 0 to 0 | 1 | 0 to 0	Word	-	-	-	-	-	-	3	0.05	0.03	0.4	126
		Long Word	(S) (R)L0 n 31(LSB) n → 0 to 0 | 1 | 0 to 0	Long Word							3	0.05	0.03	0.5	
	Encode	Word	(S)0 n 15(LSB) (R) 0 to 0 | 1 | X to X → n (X to X: Optional)	Word	-	⇕	-	-	-	-	3	0.11	0.066	0.6	128
		Long Word	(S)L0 n 31(LSB) (R) 0 to 0 | 1 | X to X → n (X to X: Optional)	Long Word							3	0.11	0.066	0.7	

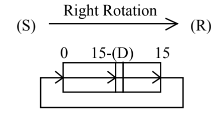
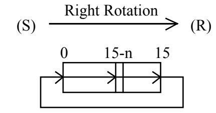
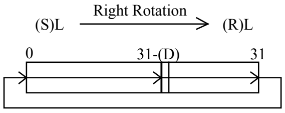
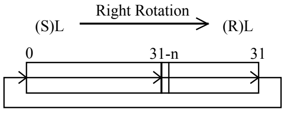
(4/7)

Major Class	Class	Parameter Format	Contents of Process	Unit of Process	Flag						Number of steps	Time of Process (ms)			Page
					X	E	P	N	Z	V		2α	2αE, 2αH(f)	4α	
Shift Instruction	Logical Right Shift	Word		Word	-	-	-	-	-	-	4	0.06	0.036	0.6	130
		Word Constant		Word							4	0.06	0.036	0.5	
		Long Word		Long Word							4	0.06	0.036	1.0	
		Long Constant		Long Word							4	0.06	0.036	0.9	
	Logical Left Shift	Word		Word	-	-	-	-	-	-	4	0.06	0.036	0.6	132
		Word Constant		Word							4	0.06	0.036	0.5	
		Long Word		Long Word							4	0.06	0.036	1.0	
		Long Constant		Long Word							4	0.06	0.036	0.9	
	Arithmetic Right Shift	Word		Word	-	-	-	-	-	-	4	0.06	0.036	0.6	134
		Word Constant		Word							4	0.06	0.036	0.5	

(5/7)

Major Class	Class	Parameter Format	Contents of Process	Unit of Process	Flag X	E	P	N	Z	V	Number of steps	Time of Process (ms) 2α	2αE, 2αH(f)	4α	Page
Shift Instruction	Arithmetic Right Shift	Long Word		Long Word	-	-	-	-	-	-	4	0.06	0.036	1.0	134
		Long Constant		Long Word							4	0.06	0.036	0.9	
	Arithmetic Left Shift	Word	 0 (D) 15 (S) Sign Change Check 0 (R) 0 to 0	Word	-	-	-	-	-	⇕	4	0.07	0.042	0.8	136
		Word Constant		Word							4	0.07	0.042	0.7	
		Long Word	 0 (D) 31 (S)L Sign Change Check 0 (R)L 0 to 0	Long Word							4	0.07	0.042	1.3	
		Long Constant	 0 n 31 (S)L Sing Change Check 0 (R)L 0 to 0	Long Word							4	0.07	0.042	1.2	
Rotate Instruction	Left Rotation	Word		Word	-	-	-	-	-	-	4	0.06	0.036	0.6	138
		Word Constant		Word							4	0.06	0.036	0.5	
		Long Word		Long Word							4	0.06	0.036	1.1	
		Long Word		Long Word							4	0.06	0.036	1.0	

(6/7)

Major Class	Class	Parameter Format	Contents of Process	Unit of Process	Flag						Number of steps	Time of Process (ms)			Page
					X	E	P	N	Z	V		2α	2αE, 2αH(f)	4α	
Rotate Instruction	Right Rotation	Word		Word	-	-	-	-	-	-	4	0.06	0.036	0.6	140
		Word Constant		Word							4	0.06	0.036	0.5	
		Long Word		Long Word							4	0.06	0.036	1.1	
		Long Word		Long Word							4	0.06	0.036	1.0	
Function Process Instruction	LIMITER	Word	When (D1)<(S): (D1)→(R) When (D2)≤(S)≤(D1): (S)→(R) When (S)<(D2): (D2)→(R)	Word	-	⇕	-	-	-	-	5	0.07	0.042	0.7	142
		Word Constant	When n1<(S): n1→(R) When n2≤(S)≤n1: (S)→(R) When (S)<n2: n2→(R)	Word							5	0.07	0.042	0.5	
		Long Word	When (D1)L<(S)L: (D1)L→(R)L When (D2)L≤(S)L≤(D1)L: (S)L→(R)L When (S)L<(D2)L: (D2)L→(R)L	Long Word							5	0.08	0.048	1.0	
	DEAD BAND	Word	When (D1)<(S): (S)-(D1)→(R) When (D2)≤(S)≤(D1): 0→(R) When (S)<(D2) : (S)-(D2)→(R)	Word	-	⇕	-	-	-	⇕	5	0.08	0.048	0.7	144
		Word Constant	When n1<(S): (S)-n1→(R) When n2≤(S)≤n1: 0→(R) When (S)<n2 : (S)-n2→(R)	Word							5	0.08	0.048	0.5	
		Long Word	When (D1)L<(S)L: (S)L-(D1)L→(R)L When (D2)L≤(S)L≤(D1)L: 0→(R)L When (S)L<(D2)L: (S)L-(D2)L→(R)L	Long Word							5	0.08	0.048	1.0	
	DEAD ZONE	Word	When (S)>0: (S)+(D1)→(R) When (S)=0: 0→(R) When (S)<0: (S)+(D2)→(R)	Word	-	⇕	-	-	-	⇕	5	0.08	0.048	0.7	146
		Word Constant	When (S)>0: (S)+n1→(R) When (S)=0: 0→(R) When (S)<0: (S)+n2→(R)	Word							5	0.08	0.048	0.5	
		Long Word	When (S)>0: (S)L+(D1)L→(R)L When (S)=0: 0→(R)L When (S)<0: (S)L+(D2)L→(R)L	Long Word							5	0.08	0.048	1.0	
	Root	Word	When (S)≥0: $\sqrt{(S)}$ →(R) When (S)<0: 0→(R)	Word	-	-	-	-	-	-	3	0.17	0.102	0.8	148
		Long Word	When (S)≥0: $\sqrt{(S)L}$ →(R)L When (S)<0: 0→(R)L	Long Word							3	0.20	0.12	2.2	

(7/7)

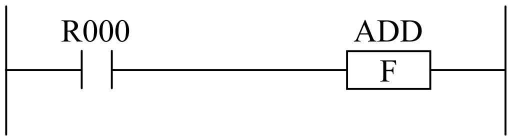
Major Class	Class	Parameter Format	Contents of Process	Unit of Process	Flag						Number of steps	Time of Process (ms)			Page
					X	E	P	N	Z	V		2α	2αE, 2αH(f)	4α	
Function Process Instruction	MAX	Word	When (S)≥(D): (S)→(R) When (S)<(D): (D)→(R)	Word	-	-	-	-	-	-	4	0.05	0.03	0.6	150
		Word Constant	When (S)≥n: (S)→(R) When (S)<n: n→(R)	Word							4	0.05	0.03	0.5	
		Long Word	When (S)L≥(D)L: (S)L→(R)L When (S)L<(D)L: (D)L→(R)L	Long Word							4	0.05	0.03	0.8	
	MIN	Word	When (S)≤(D): (S)→(R) When (S)>(D): (D)→(R)	Word	-	-	-	-	-	-	4	0.05	0.03	0.6	152
		Word Constant	When (S)≤n: (S)→(R) When (S)>n: n→(R)	Word							4	0.05	0.03	0.5	
		Long Word	When (S)L≤(D)L: (S)L→(R)L When (S)L>(D)L: (D)L→(R)L	Long Word							4	0.05	0.03	0.8	
Special Instruction	Clear	-	Clears X area.	-	-	-	-	-	-	-	1	0.77	0.462	2.3	154
		-	Clears Y area.	-	-	-	-	-	-	-	1	0.77	0.462	2.3	
		-	Clears C area.	-	-	-	-	-	-	-	1	0.77	0.462	2.3	
		-	Clears R area.	-	-	-	-	-	-	-	1	0.77	0.462	2.3	
		-	Clears K area.	-	-	-	-	-	-	-	1	0.77	0.462	2.3	
		-	Clears T and Count Value areas.	-	-	-	-	-	-	-	1	2.21	1.326	7.0	
		-	Clears U and Count Value areas.	-	-	-	-	-	-	-	1	1.50	0.9	4.0	
		-	Clears C and Count Value areas.	-	-	-	-	-	-	-	1	1.50	0.9	4.0	
		-	Clears V area.	-	-	-	-	-	-	-	1	0.77	0.462	2.3	
		-	Clears E area.	-	-	-	-	-	-	-	1	0.77	0.462	2.3	
		-	Clears a function flag.	-	0	0	0	0	0	0	1	0.007	0.005	0.4	

ADD: ADD

<table>
<tr><td>Function</td><td colspan="10">ADD performs an addition of the contents of the source and destination or immediate data and stores the sum in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>√</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>↕</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>ADD
F</td><td colspan="7">S.W+D.W→R.W</td></tr>
<tr><td>Registers with immediate data</td><td>ADD
F</td><td colspan="7">S.W+n.W→R.W</td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>ADD
F</td><td colspan="7">S.L+D.L→R.L</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word variable
D: Address storing the destination L: Long Word variable
n: Immediate data
R: Address storing the result (result of operation)</td></tr>
<tr><td rowspan="3">Contents of Process</td><td>Word</td><td colspan="9">(S)W+(D)W→(R)W</td></tr>
<tr><td>Word Constant</td><td colspan="9">(S)W+nW→(R)W</td></tr>
<tr><td>Long Word</td><td colspan="9">(S)L+(D)L→(R)L</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) A D D ␣Parameter, Parameter, Parameter Enter</td></tr>
<tr><td>Flag Setting</td><td colspan="10">V: (Word) 1, if (R) < -32768 or 32767 < (R)
0, otherwise
(Long Word) 1, if (R) < -2147483648 or 2147483647 < (R)
0, otherwise
Others: Not affected</td></tr>
<tr><td>Notes</td><td colspan="10">The following full scale values are set in the result, if an overflow occurs.<table><tr><td></td><td>Word</td><td>Long Word</td></tr><tr><td>Positive overflow</td><td>H7FFF</td><td>H7FFFFFFF</td></tr><tr><td>Negative overflow</td><td>H8000</td><td>H80000000</td></tr></table></td></tr>
</table>

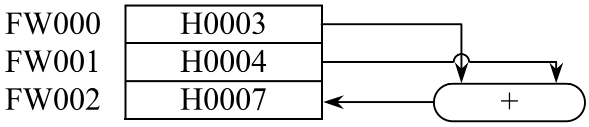
√: Applicable
na: Not applicable

[Word]

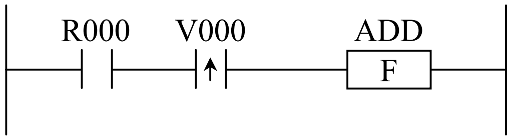


FW000+FW001→FW002

If an input condition, R000, is ON, the contents of FW000 and FW001 are added, and the sum is stored in FW002.

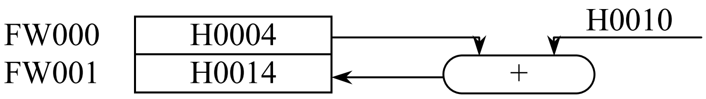


[Word Constant]

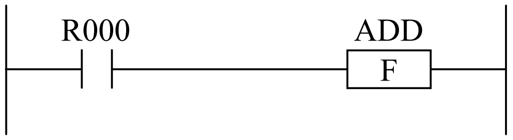


FW100+H010→FW001

The contents of FW000 and immediate data, H0010, are added only once when an input condition, R000, turns OFF → ON. The sum is stored in FW001.

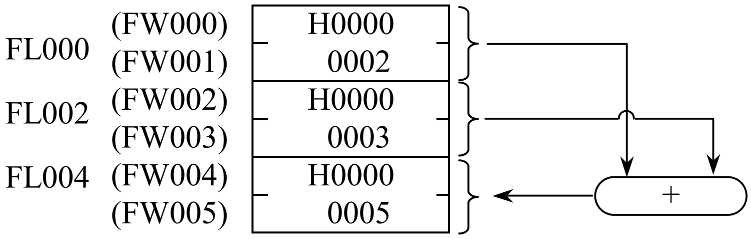


[Long Word]



FL000*FL002→FL004

If an input condition, R000, is ON, the contents of FL000 and FL002 are added, and the sum is stored in FL004.



SUB: SUBTRACT

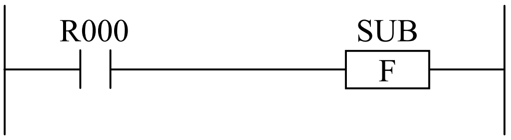
<table>
<tr><td>Function</td><td colspan="11">SUB performs a subtraction of the contents of the source and the destination or immediate data, and stores the difference in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>√</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>↕</td></tr>
</table>

<table>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>SUB
–$\boxed{\text{F}}$–</td><td>SW-DW→RW</td></tr>
<tr><td>Registers with immediate data</td><td>SUB
–$\boxed{\text{F}}$–</td><td>SW-nW→RW</td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>SUB
–$\boxed{\text{F}}$–</td><td>SL-nL→RL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="2"></td></tr>
<tr><td colspan="4">S: Address storing the source W: Word
D: Address storing the destination L: Long Word
n: Immediate data
R: Address storing the result (result of operation)</td></tr>
<tr><td rowspan="3">Contents of Process</td><td>Word</td><td colspan="3">(S)W-(D)W→(R)W</td></tr>
<tr><td>Word Constant</td><td colspan="3">(S)W-nW→(R)W</td></tr>
<tr><td>Long Word</td><td colspan="3">(S)L-(D)L→(R)L</td></tr>
<tr><td>Input Procedure</td><td colspan="4">(Shift + F) S U B ⌴Parameter, Parameter, Parameter Enter</td></tr>
<tr><td>Flag Setting</td><td colspan="4">V: (Word) 1, if (R) < -32768 or 32767 < (R)
0, otherwise
(Long Word) 1, if (R) < -2147483648 or 2147483647 < (R)
0, otherwise
Others: Not affected</td></tr>
<tr><td>Notes</td><td colspan="4">The following full scale values are set in the result, if an overflow occurs.
<table><tr><td></td><td>Word</td><td>Long Word</td></tr><tr><td>Positive overflow</td><td>H7FFF</td><td>H7FFFFFFF</td></tr><tr><td>Negative overflow</td><td>H8000</td><td>H80000000</td></tr></table></td></tr>
</table>

√: Applicable

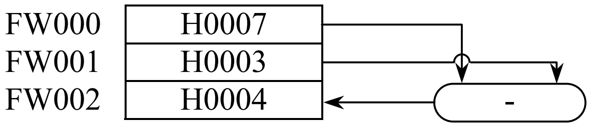
na: Not applicable

[Word]

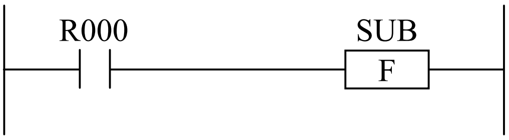


FW000-FW001→FW002

If an input condition, R000, is ON, the contents of FW001 is subtracted from the contents of FW000, and the difference is stored in FW002.

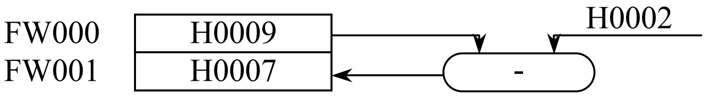


[Word Constant]

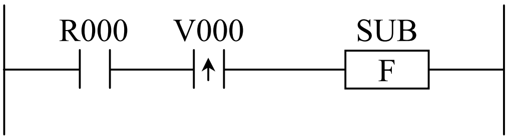


FW000-H0002→FW001

Immediate data, H0002, is subtracted from the contents of FW000, if an input condition, R000, is ON, and the result is stored in FW001.

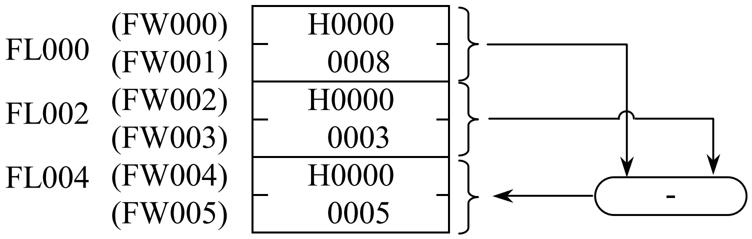


[Long Word]



FL000: FL002→FL004

The contents of FL002 are subtracted from the contents of FL000 only once when the input condition, R000, turns OFF → ON, and the difference is stored in FL004.



INC: INCREMENT (+1)

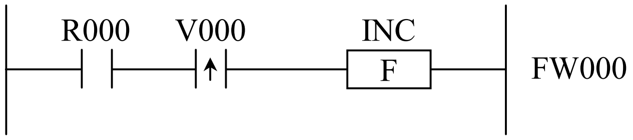
Function	INC adds 1 to the contents of the source.									
Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	na	√	na	-	-	-	-	-	↕
Parameter	Word	Registers	INC ─[F]─	SW						
		Registers with immediate data								
	Long Word	Registers	INC ─[F]─	SL						
		Registers with immediate data								
	S: Address storing the source			W: Word L: Long Word						
Contents of Process	Word	(S)W+1→(S)W								
	Long Word	(S)L+1→(S)L								
Input Procedure	(Shift + F) I N C ␣Parameter Enter									
Flag Setting	V: (Word) 1, if result = -32768 0, otherwise (Long Word) 1, if result = -2147483648 0, otherwise Others: Not affected									
Notes	The following full scale values are set in the result, if an overflow occurs.									

Word	Long Word
H7FFF	H7FFFFFFF

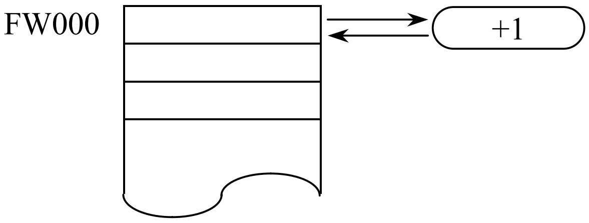
√: Applicable

na: Not applicable

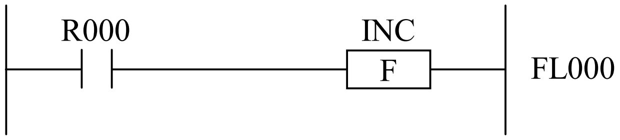
[Word]



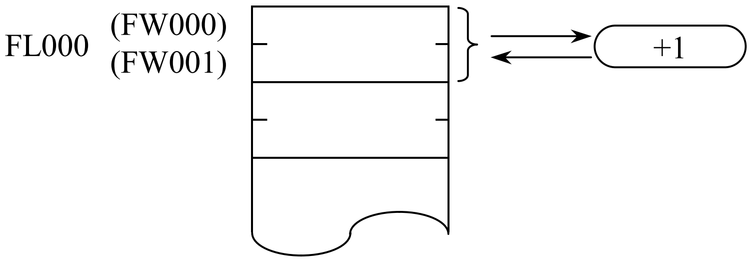
One (1) is added to the contents of FW000 only once when the input condition, R000, turns OFF to ON.



[Long Word]



If the input condition, R000, is ON, one (1) is added to the contents of FL000.



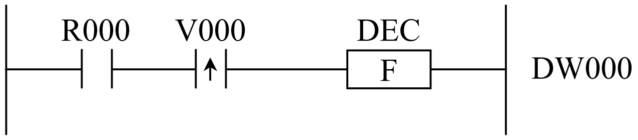
DEC: DECREMENT (-1)

<table>
<tr><td>Function</td><td colspan="10">DEC subtract 1 from the contents of the source.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>na</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>↕</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td colspan="2">Registers</td><td>DEC
─[F]─</td><td colspan="6">SW</td></tr>
<tr><td colspan="2">Registers with immediate data</td><td colspan="7"></td></tr>
<tr><td rowspan="2">Long Word</td><td colspan="2">Registers</td><td>DEC
─[F]─</td><td colspan="6">SL</td></tr>
<tr><td colspan="2">Registers with immediate data</td><td colspan="7"></td></tr>
<tr><td colspan="5">S: Address storing the source</td><td colspan="5">W: Word
L: Long Word</td></tr>
<tr><td rowspan="2">Contents of Process</td><td colspan="2">Word</td><td colspan="8">(S)W-1→(S)W</td></tr>
<tr><td colspan="2">Long Word</td><td colspan="8">(S)L-1→(S)L</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) D E C ␣Parameter Enter</td></tr>
<tr><td>Flag Setting</td><td colspan="10">V: (Word) 1, if result = -32767
0, otherwise
(Long Word) 1, if result = -2147483647
0, otherwise
Others: Not affected</td></tr>
<tr><td>Notes</td><td colspan="10">The following full scale values are set in the result, if an overflow occurs.<table><tr><th>Word</th><th>Long Word</th></tr><tr><td>H8000</td><td>H80000000</td></tr></table></td></tr>
</table>

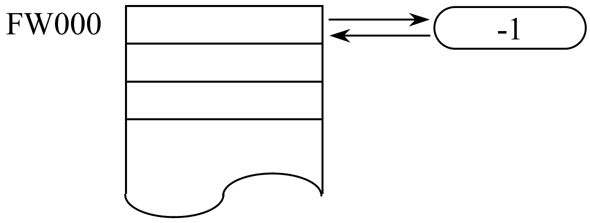
√: Applicable

na: Not applicable

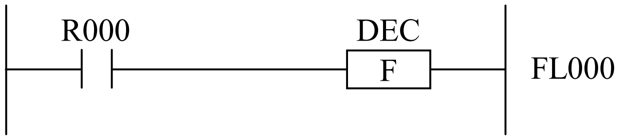
[Word]



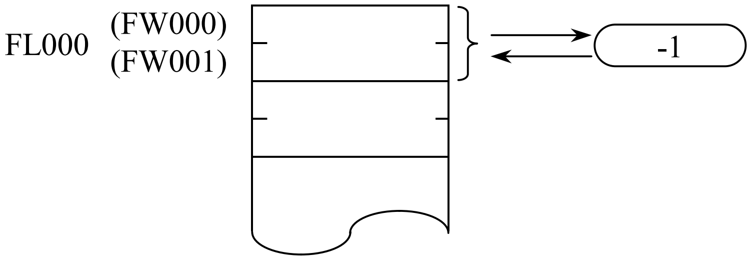
One (1) is subtracted from the contents of FW000 only once when the input condition, R000, turns OFF to ON.



[Long Word]



If the input condition, R000, is ON, one (1) is subtracted from the contents of FL000.



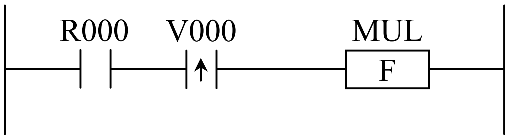
MUL: MULTIPLY

<table>
<tr><td>Function</td><td colspan="10">MUL performs a multiplication on the contents of the source and destination or immediate data, and stores the product in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>√</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>↕</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>MUL
┤F├</td><td colspan="7">SW*DW→RW</td></tr>
<tr><td>Registers with immediate data</td><td>MUL
┤F├</td><td colspan="7">SW*nW→RW</td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>MUL
┤F├</td><td colspan="7">SL*DL→RL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
D: Address storing the destination L: Long Word
n: Immediate data
R: Address storing the product (result of operation)</td></tr>
<tr><td rowspan="3">Contents of Process</td><td>Word</td><td colspan="9">(S)W×(D)W→(R)W</td></tr>
<tr><td>Word Constant</td><td colspan="9">(S)W×nW→(R)W</td></tr>
<tr><td>Long Word</td><td colspan="9">(S)L×(D)L→(R)L</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) M U L ␣Parameter, Parameter, Parameter Enter</td></tr>
<tr><td>Flag Setting</td><td colspan="10">V: (Word) 1, if (R) < -32768 or 32767 < (R)
0, otherwise
(Long Word) 1, if (R) < -2147483648 or 2147483647 < (R)
0, otherwise
Others: Not affected</td></tr>
<tr><td>Notes</td><td colspan="10">The following full scale values are set in the result, if an overflow occurs.<table><tr><td></td><td>Word</td><td>Long Word</td></tr><tr><td>Positive overflow</td><td>H7FFF</td><td>H7FFFFFFF</td></tr><tr><td>Negative overflow</td><td>H8000</td><td>H80000000</td></tr></table></td></tr>
</table>

√: Applicable

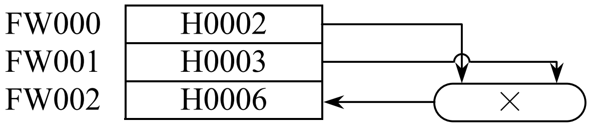
na: Not applicable

[Word]

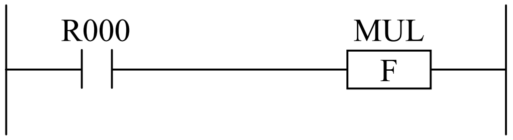


FW000*FW001→FW002

If the input condition, R000, is ON, the contents of FW000 and FW001 are added, and the product stored in FW002.

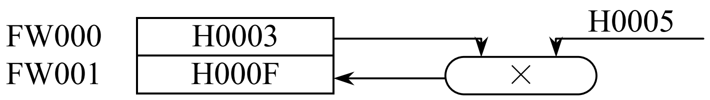


[Word Constant]

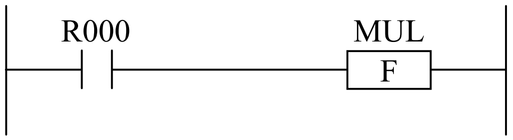


FW000*H0005→FW001

If an input condition, R000, is ON, the contents of FW000 is multiplied by immediate data, H0005, and the product is stored in FW001.

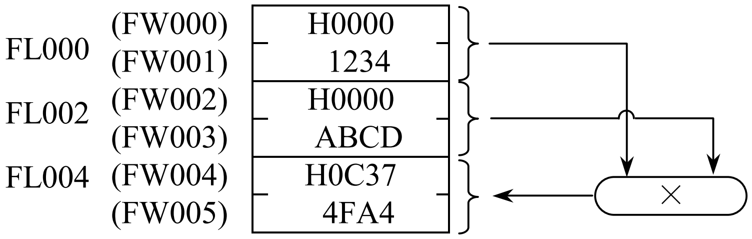


[Long Word]



FL000*FL002→FL004

If the input condition, R000, is ON, the contents of FL000 is multiplied by the contents of FL002, and the product is stored in FL004.



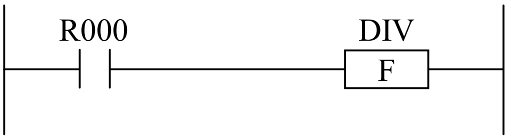
DIV: DIVIDE

<table>
<tr><td>Function</td><td colspan="10">DIV performs a division on the contents of the source and the destination or immediate data and stores the quotient (integer part, only) in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>√</td><td>√</td><td>na</td><td>-</td><td>↕</td><td>-</td><td>-</td><td>-</td><td>↕</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>DIV
-[F]-</td><td colspan="7">SW/DW→RW</td></tr>
<tr><td>Registers with immediate data</td><td>DIV
-[F]-</td><td colspan="7">SW/nW→RW</td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>DIV
-[F]-</td><td colspan="7">SL/DL→RL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
D: Address storing the destination L: Long Word
n: Immediate data
R: Address storing the result (result of operation)</td></tr>
<tr><td rowspan="3">Contents of Process</td><td colspan="2">Word</td><td colspan="8">(S)W÷(D)W→(R)W</td></tr>
<tr><td colspan="2">Word Constant</td><td colspan="8">(S)W÷nW→(R)W</td></tr>
<tr><td colspan="2">Long Word</td><td colspan="8">(S)L÷(D)L→(R)L</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) D I V ␣Parameter, Parameter, Parameter Enter</td></tr>
<tr><td>Flag Setting</td><td colspan="10">E: When (D) = 0 or n = 0: 1 Otherwise: 0
V: (Word) 1, if result = 32768
0, otherwise
(Long Word) 1, if result = 2147483648
0, otherwise
Others: Not affected</td></tr>
<tr><td>Notes</td><td colspan="10">• If divided by 0, the error flag (E) turns ON, and the overflow flag, OFF. The result is not affected.
• The following full scale values are set in result, if an overflow occurs.<table><tr><th>Word</th><th>Long Word</th></tr><tr><td>H7FFF</td><td>H7FFFFFFF</td></tr></table></td></tr>
</table>

√: Applicable

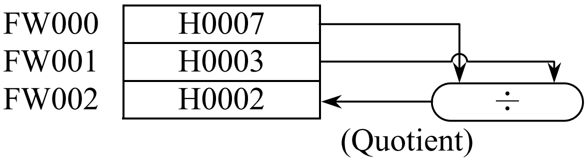
na: Not applicable

[Word]

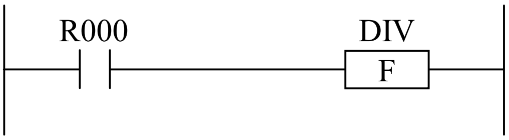


FW000/FW001→FW002

If the input condition, R000, is ON, the contents of FW000 is divided by the contents of FW001, and the result (quotient) is stored in FW002.

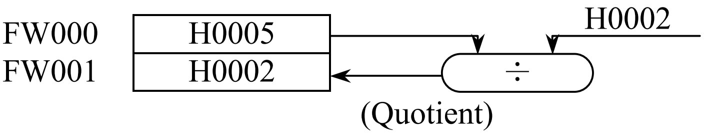


[Word Constant]

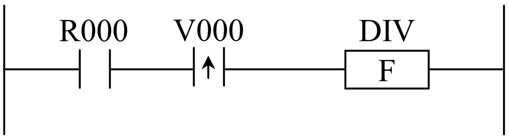


FW000/H0002→FW001

If the input condition, R000, is ON, the contents of FW000 is divided by immediate data, H0002, and the result (quotient) is stored in FW001.

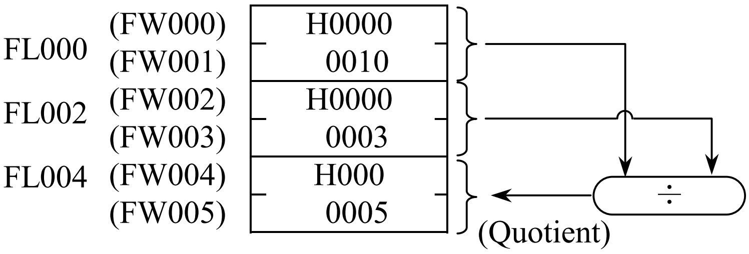


[Long Word]



FL000/FL002→FL004

The contents of FL000 is divided by the contents of FL002 only once when the input condition, R000, changes from OFF to ON, and the result (quotient) is stored in FL004.



MOD: MOD (Remainder)

Function	MOD performs a division on the contents of the source and the destination or immediate data and stores the remainder in Result.

	Word		Long Word		Flag					
Parameter Format	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	√	√	na	-	↕	-	-	-	↕

Parameter	Word	Registers	MOD F	SW%DW→RW
		Registers with immediate data	MOD F	SW%nW→RW
	Long Word	Registers	MOD F	SL%DL→RL
		Registers with immediate data		

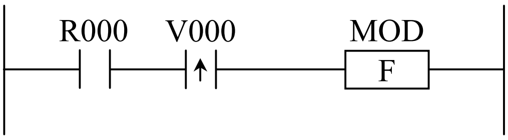
S: Address storing the source W: Word
D: Address storing the destination L: Long Word
n: Immediate data
R: Address storing the result (result of operation)

Contents of Process	Word	(S)W%(D)W→(R)W
	Word Constant	(S)W%nW→(R)W
	Long Word	(S)L%(D)L→(R)L

Input Procedure	(Shift + F) M O D ␣Parameter, Parameter, Parameter Enter
Flag Setting	E: When (D) = 0 or n = 0: 1 Otherwise: 0 V: (Word) 1, if result = 32768 0, otherwise (Long Word) 1, if result = 2147483648 0, otherwise Others: Not affected
Notes	• If divided by 0, the error flag (E) turns ON, and the overflow flag, OFF. The result is not affected. • Zero (0) is stored in Result, if an overflow occurs.

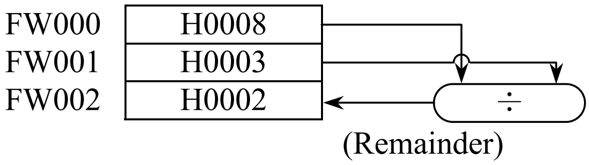
√: Applicable
na: Not applicable

[Word]

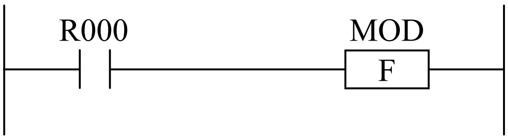


FW000%FW001→FW002

The contents of FW000 is divided by the contents of FW001 only once when the input condition, R000, changes from OFF to ON, and the remainder is stored in FL002.

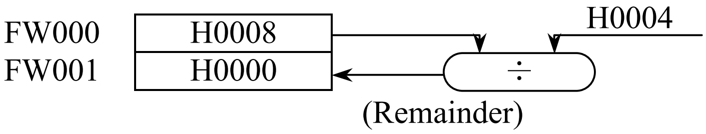


[Word Constant]

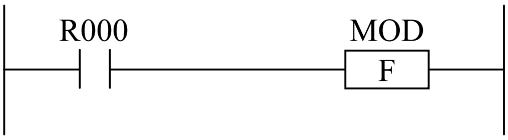


FW000%H0004→FW001

If the input condition, R000, is ON, the contents of FW000 is divided by immediate data, H0004, and the remainder is stored in FW001.

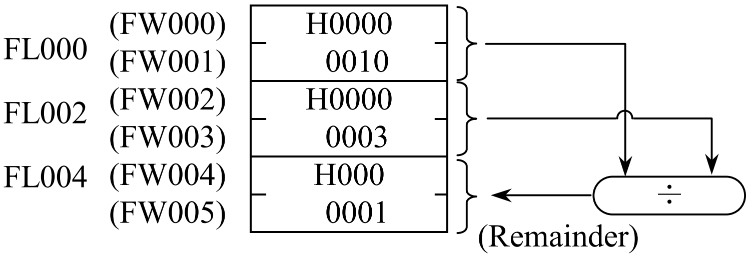


[Long Word]



FL000%FL002→FL004

If the input condition, R000, is ON, the contents of FW000 is divided by the contents of FW002, and the remainder is stored in FW004.



SCL: SCALE CHANGE

Function	SCL performs a scale change of the contents of the source (multiplication by destination, D1/D2 or immediate data, n1/n2), and stores the result in Result.

Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	√	na	na	-	↕	-	-	-	↕

Parameter				
	Word	Registers	SCL F	SW: D1W: D2W→RW
		Registers with immediate data	SCL F	SW: n1W: n2W→RW
	Long Word	Registers		
		Registers with immediate data		

S: Address storing the source W: Word

D1 and D2: Address storing the destination L: Long Word

n1 and n2: Immediate data

R: Address storing the result (result of operation)

Contents of Process		
	Word	(S)W×(D1)W÷(D2)W→(R)W
	Word Constant	(S)W×n1W÷n2W→(R)W

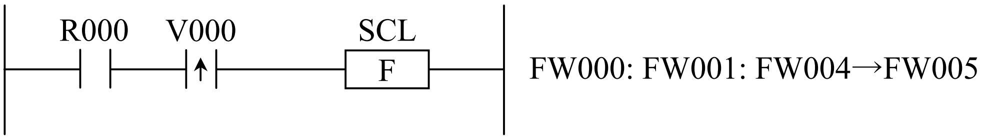
Input Procedure	(Shift + F) S C L ␣Parameter, Parameter, Parameter, Parameter Enter
Flag Setting	E: 1, if (D2)=0 or n2=0 0, otherwise V: 1, if (R) < -32768 or 32767 < (R) 0, otherwise Others: Not affected
Notes	• If divided by 0, the error flag (E) turns ON, and the overflow flag (V), OFF. The result is not affected. • The following full scale values are stored in result, if an overflow occurs.

Positive overflow	H7FFF
Negative overflow	H8000

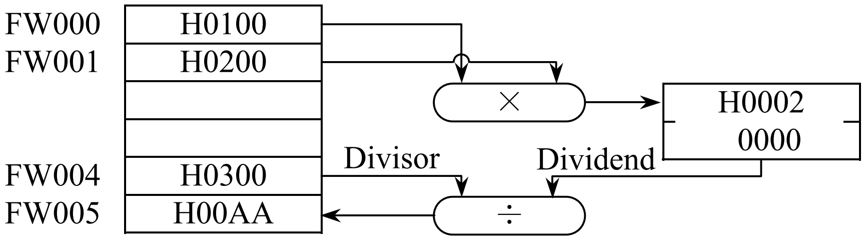
√: Applicable

na: Not applicable

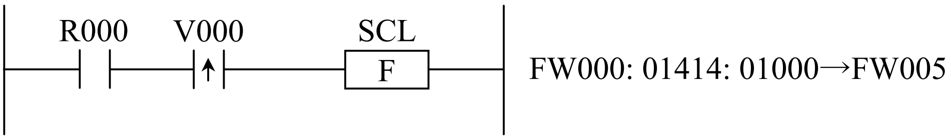
[Word]



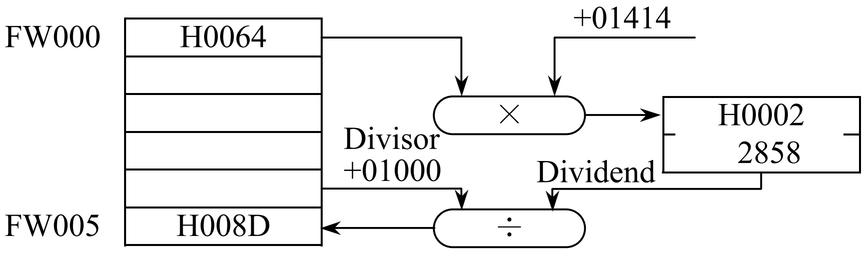
If the input condition, R000, is ON, a scale change is made on the contents of FW000, and the result is stored in FW005.



[Word Constant]



A scale change is made on the contents of FW000 only once when the input condition, R000, changes from OFF to ON, and the result is stored in FW005.



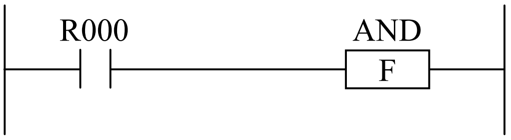
AND: AND

<table>
<tr><td>Function</td><td colspan="10">AND performs an AND operation on the contents of the source and the destination or immediate data, and stores the result in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>√</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>AND
─┤F├─</td><td colspan="7">SW: DW→RW</td></tr>
<tr><td>Registers with immediate data</td><td>AND
─┤F├─</td><td colspan="7">SW: nW→RW</td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>AND
─┤F├─</td><td colspan="7">SL: DL→RL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
D: Address storing the destination L: Long Word
n: Immediate data
R: Address storing the result (result of operation)</td></tr>
<tr><td rowspan="3">Contents of Process</td><td colspan="2">Word</td><td colspan="8">(S)W∧(D)W→(R)W</td></tr>
<tr><td colspan="2">Word Constant</td><td colspan="8">(S)W∧nW→(R)W</td></tr>
<tr><td colspan="2">Long Word</td><td colspan="8">(S)L∧(D)L→(R)L</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) A N D ␣Parameter, Parameter, Parameter Enter</td></tr>
<tr><td>Notes</td><td colspan="10">All flags remain intact.</td></tr>
</table>

√: Applicable

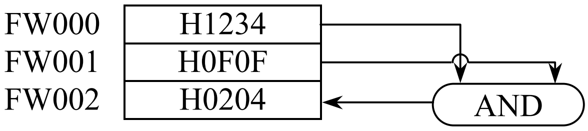
na: Not applicable

[Word]

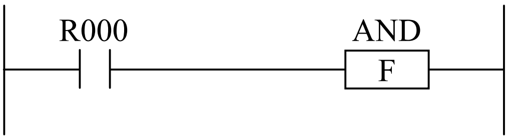


FW000: FW001→FW002

If the input condition, R000, is ON, an AND operation is made on the contents of FW000 and FW001, and the result is stored in FW002.

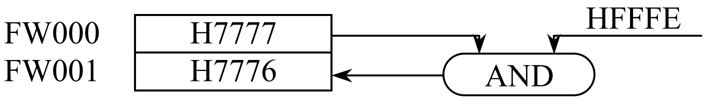


[Word Constant]

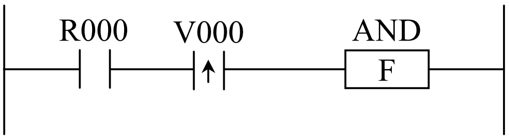


FW000: HFFFE→FW001

If the input condition, R000, is ON, an AND operation is made on the contents of FW000 and immediate data, HFFFE, and the result is stored in FW001.

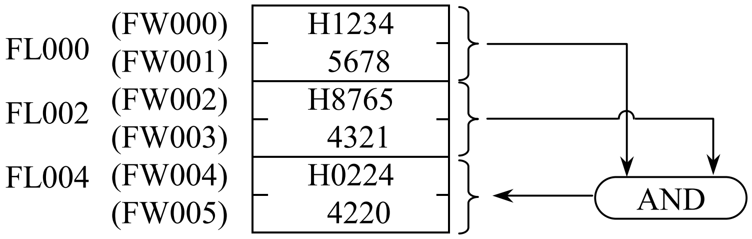


[Long Word]



FL000: FL002→FL004

An AND operation is made on the contents of FL000 and FL002 only once when the input condition, R000, changes from OFF to ON, and the result is stored in FW004.



OR: OR

Function	OR performs an OR operation on the contents of the source and the destination or immediate data, and stores the result in Result.

Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	√	√	na	-	-	-	-	-	-

Parameter	Word	Registers	OR $\boxed{F}$	SW: DW→RW
		Registers with immediate data	OR $\boxed{F}$	SW: nW→RW
	Long Word	Registers	OR $\boxed{F}$	SL: DL→RL
		Registers with immediate data		
	S: Address storing the source D: Address storing the destination n: Immediate data R: Address storing the result (result of operation)	W: Word L: Long Word		

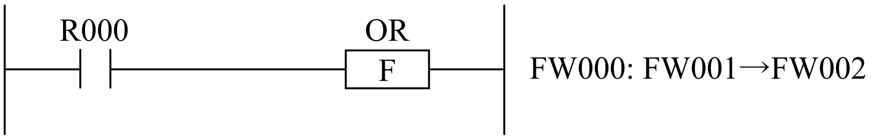
Contents of Process	Word	(S)W∨(D)W→(R)W
	Word Constant	(S)W∨nW→(R)W
	Long Word	(S)L∨(D)L→(R)L

Input Procedure	(Shift + F) O R ⌴Parameter, Parameter, Parameter Enter
Notes	All flags remain intact.

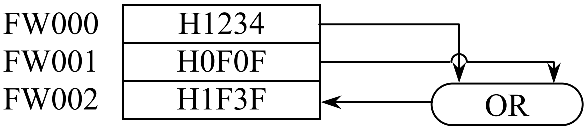
√: Applicable

na: Not applicable

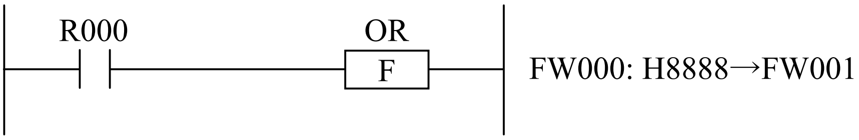
[Word]



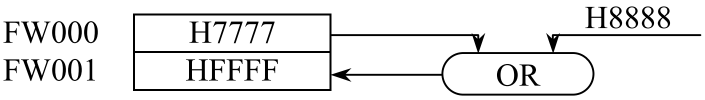
If the input condition, R000, is ON, an OR operation is made on the contents of FW000 and FW001, and the result is stored in FW002.



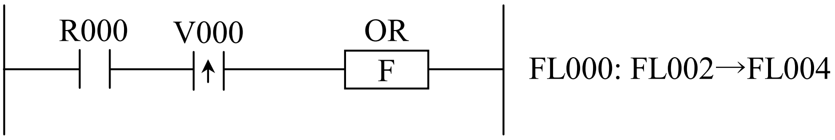
[Word Constant]



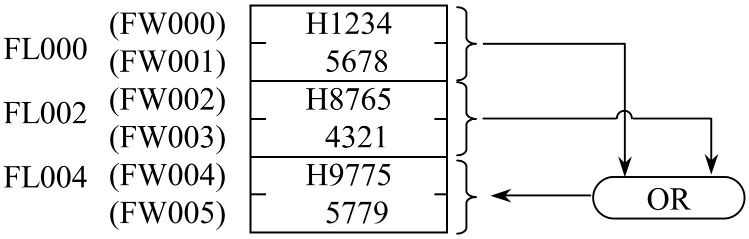
If the input condition R000 is ON, OR is made on the contents of FW000 and immediate data, H8888, and the result is stored in FW001.



[Long Word]



An OR operation is made on the contents of FL000 and FL002 only once when the input condition, R000, changes from OFF to ON, and the result is stored in FW004.



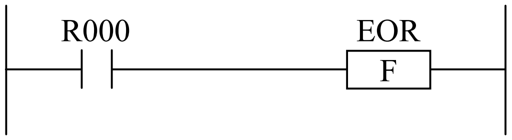
EOR: EXCLUSIVE OR

<table>
<tr><td>Function</td><td colspan="10">EOR performs an exclusive OR operation on the contents of the source and the destination or immediate data, and stores the result in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>√</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>EOR
┤F├</td><td colspan="7">SW: DW→RW</td></tr>
<tr><td>Registers with immediate data</td><td>EOR
┤F├</td><td colspan="7">SW: nW→RW</td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>EOR
┤F├</td><td colspan="7">SL: DL→RL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
D: Address storing the destination L: Long Word
n: Immediate data
R: Address storing the result (result of operation)</td></tr>
<tr><td rowspan="3">Contents of Process</td><td colspan="2">Word</td><td colspan="8">(S)W⊕(D)W→(R)W</td></tr>
<tr><td colspan="2">Word Constant</td><td colspan="8">(S)W⊕nW→(R)W</td></tr>
<tr><td colspan="2">Long Word</td><td colspan="8">(S)L⊕(D)L→(R)L</td></tr>
<tr><td>Input Procedure</td><td colspan="10">([Shift] + [F]) [E] [O] [R] ␣Parameter, Parameter, Parameter [Enter]</td></tr>
<tr><td>Notes</td><td colspan="10">All flags remain intact.</td></tr>
</table>

√: Applicable

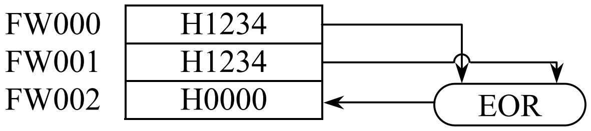
na: Not applicable

[Word]

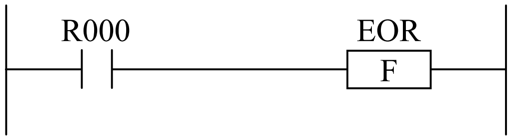


FW000: FW001→FW002

If the input condition, R000, is ON, an exclusive OR is made on the contents of FW000 and FW001, and the result is stored in FW002.

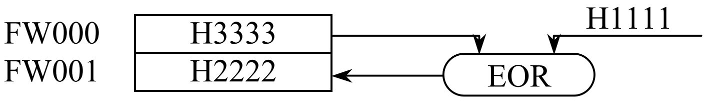


[Word Constant]

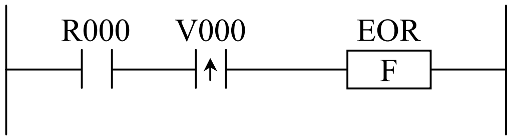


FW000: H1111→FW001

If the input condition R000 is ON, an exclusive OR is made on the contents of FW000 and immediate data, H1111, and the result is stored in FW001.

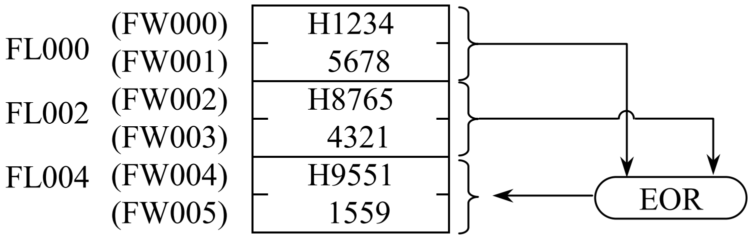


[Long Word]



FL000: FL002→FL004

An exclusive OR is made once on the contents of FL000 and FL002 only once when the input condition, R000, changes from OFF to ON, and the result is stored in FW004.



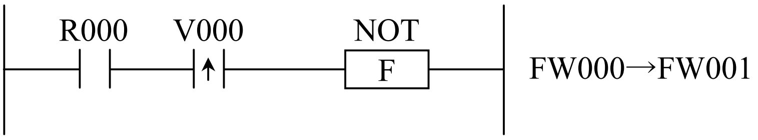
NOT: NOT (Negation)

<table>
<tr><td>Function</td><td colspan="10">NOT stores the negation (Bit inversion) of the contents of the source in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>na</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>NOT
─[F]─</td><td colspan="7">SW→RW</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>NOT
─[F]─</td><td colspan="7">SL→RL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
R: Address storing the result L: Long Word
(result of operation)</td></tr>
<tr><td rowspan="2">Contents of Process</td><td colspan="2">Word</td><td colspan="8">$\overline{(S)}W \rightarrow (R)W$</td></tr>
<tr><td colspan="2">Long Word</td><td colspan="8">$\overline{(S)L} \rightarrow (R)L$</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) N O T ⌴Parameter, Parameter Enter</td></tr>
<tr><td>Notes</td><td colspan="10">All flags remain intact.</td></tr>
</table>

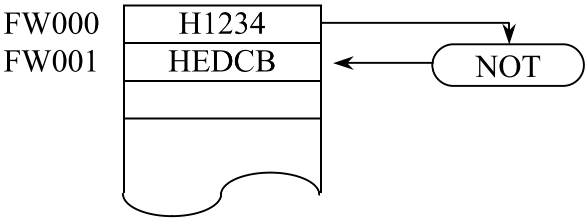
√: Applicable

na: Not applicable

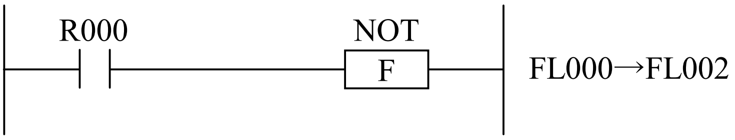
[Word]



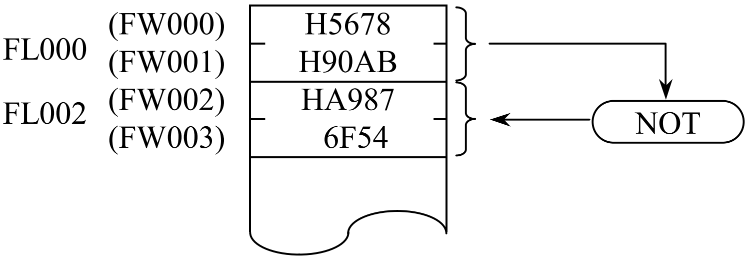
A negation (NOT) is made on the contents of FW000 only once and the result is stored in FW001, when the input condition, R000, changes from OFF to ON.



[Long Word]



If the input condition R000 is ON, a negation (NOT) is made on the contents of FL000, and the result is stored in FL002.



EQU: EQU (=)

Function	EQU checks the true and false of the equality of the contents of the source and destination or immediate data, and stores 1, if equal, or 0, if not equal, in Result.

Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	√	√	na	-	-	-	-	-	-

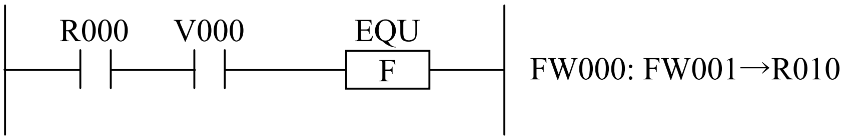
Parameter	Word	Registers	EQU -[F]-	SW: DW→RW
		Registers with immediate data	EQU -[F]-	SW: nW→RW
	Long Word	Registers	EQU -[F]-	SL: DL→RL
		Registers with immediate data		
	S: Address storing the source D: Address storing the destination n: Immediate data R: Address storing the result (result of operation)	W: Word L: Long Word		

Contents of Process	Word	When (S)W = (D)W: 1→(R)W When (S)W≠(D)W: 0→(R)W
	Word Constant	When (S)W = n W: 1→(R)W When (S)W≠n W: 0→(R)W
	Long Word	When (S)L = (D)L: 1→(R)W When (S)L ≠(D)L: 0→(R)W
Input Procedure	(Shift + F) E Q U ␣Parameter, Parameter, Parameter Enter	
Notes	All flags remain intact.	

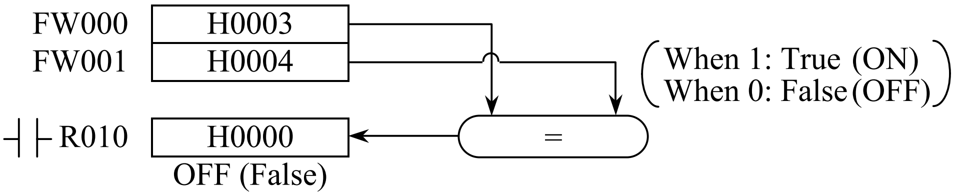
√: Applicable

na: Not applicable

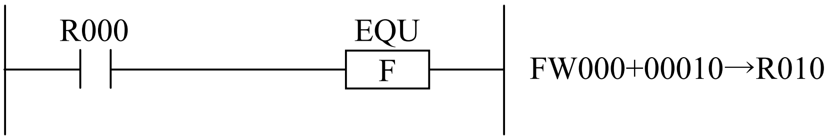
[Word]



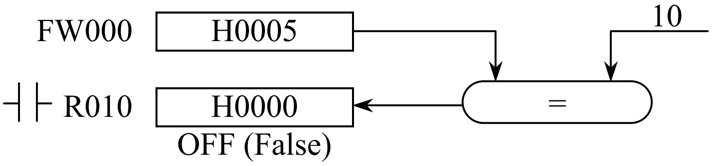
A true and false check is made on the contents of FW000 and FW001 only once when the input condition changes from OFF to ON, and the result is stored in R010.



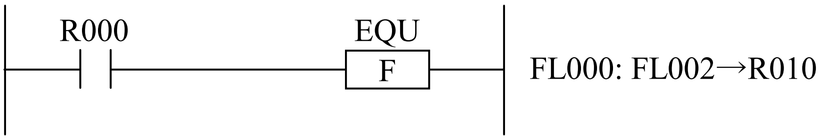
[Word Constant]



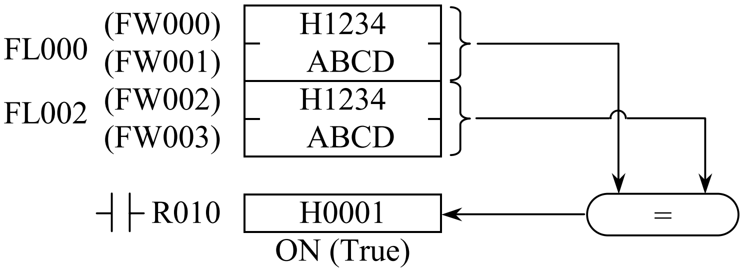
If the input condition is ON, a true and false check is made on the contents of FW000 and immediate data, 10, and the result is stored in R010.



[Long Word]



If the input condition is ON, a true and false check is made on the contents of FL000 and FL002, and the result is stored in R010.



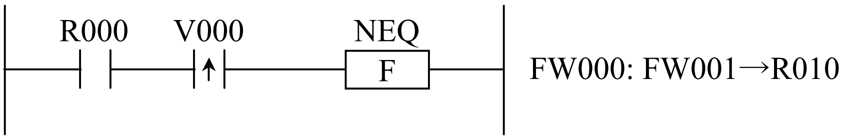
NEQ: NOT EQUAL (≠)

<table>
<tr><td>Function</td><td colspan="10">NEQ checks the true and false of the equality of the contents of the source and destination or immediate data, and stores 0, if equal, or 1, if not equal, in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>√</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>NEQ
─[F]─</td><td colspan="7">SW: DW→RW</td></tr>
<tr><td>Registers with immediate data</td><td>NEQ
─[F]─</td><td colspan="7">SW: nW→RW</td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>NEQ
─[F]─</td><td colspan="7">SL: DL→RL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source　　W: Word
D: Address storing the destination　　L: Long Word
n: Immediate data
R: Address storing the result (result of operation)</td></tr>
<tr><td rowspan="3">Contents of Process</td><td colspan="2">Word</td><td colspan="8">When (S)W = (D)W: 0→(R)W
When (S)W≠(D)W: 1→(R)W</td></tr>
<tr><td colspan="2">Word Constant</td><td colspan="8">When (S)W = n W: 0→(R)W
When (S)W≠n W: 1→(R)W</td></tr>
<tr><td colspan="2">Long Word</td><td colspan="8">When (S)L = (D)L: 0→(R)W
When (S)L ≠(D)L: 1→(R)W</td></tr>
<tr><td>Input Procedure</td><td colspan="10">([Shift] + [F]) [N] [E] [Q] ⌴Parameter, Parameter, Parameter [Enter]</td></tr>
<tr><td>Notes</td><td colspan="10">All flags remain intact.</td></tr>
</table>

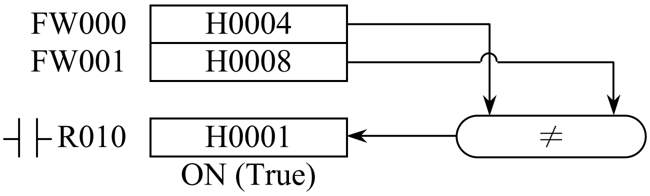
√: Applicable

na: Not applicable

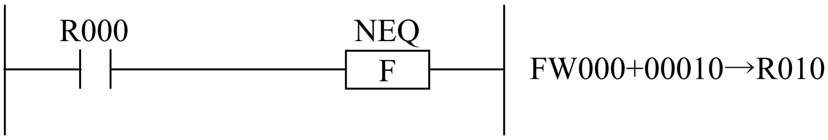
[Word]



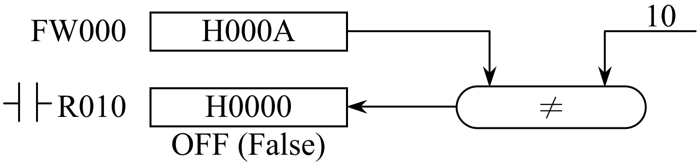
A true and false check is made on the contents of FW000 and FW001 only once when the input condition, R000, changes from OFF to ON, and the result is stored in R010.



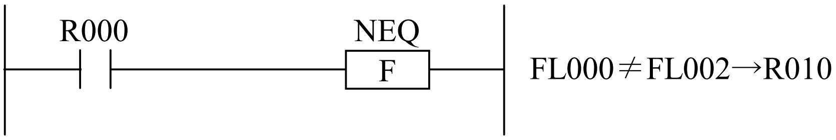
[Word Constant]



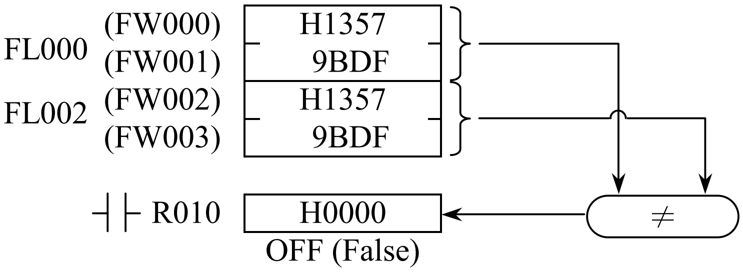
If the input condition, R000, is ON, a true and false check is made on the contents of FW000 and immediate data, 10, and the result is stored in R010.



[Long Word]



If the input condition, R000, is ON, a true and false check is made on the contents of FL000 and FL002 and the result is stored in R010.



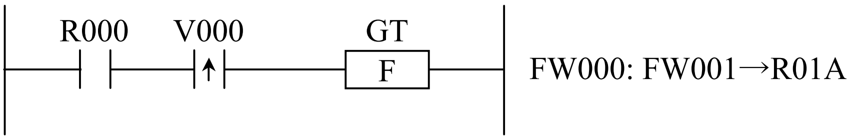
GT: GREATER THAN (>)

<table>
<tr><td>Function</td><td colspan="10">GT performs a comparison on the sizes of the contents of the source and destination, or immediate data, and stores the result in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>√</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>GT
┤F├</td><td colspan="7">SW: DW→RW</td></tr>
<tr><td>Registers with immediate data</td><td>GT
┤F├</td><td colspan="7">SW: nW→RW</td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>GT
┤F├</td><td colspan="7">SL: DL→RL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
D: Address storing the destination L: Long Word
n: Immediate data
R: Address storing the result (result of operation)</td></tr>
<tr><td rowspan="3">Contents of Process</td><td>Word</td><td colspan="9">When (S)W > (D)W: 1→(R)W
When (S)W ≤ (D)W: 0→(R)W</td></tr>
<tr><td>Word Constant</td><td colspan="9">When (S)W > n W: 1→(R)W
When (S)W ≤ n W: 0→(R)W</td></tr>
<tr><td>Long Word</td><td colspan="9">When (S)L > (D)L: 1→(R)W
When (S)L ≤ (D)L: 0→(R)W</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) G T ␣ Parameter, Parameter, Parameter Enter</td></tr>
<tr><td>Notes</td><td colspan="10">All flags remain intact.</td></tr>
</table>

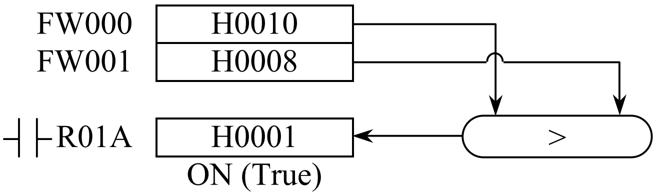
√: Applicable

na: Not applicable

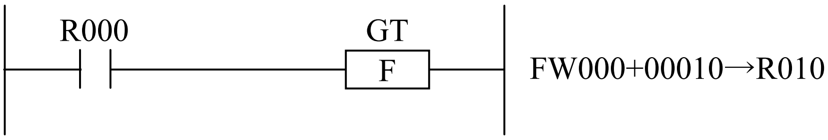
[Word]



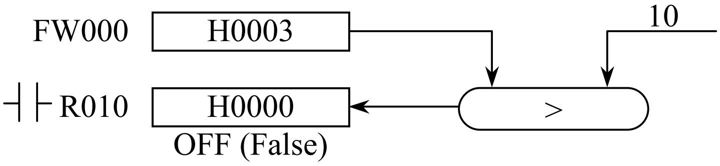
The sizes of the contents of FW000 and FW001 are compared only once when the input condition, R000, changes from OFF to ON, and the result is stored in R01A.



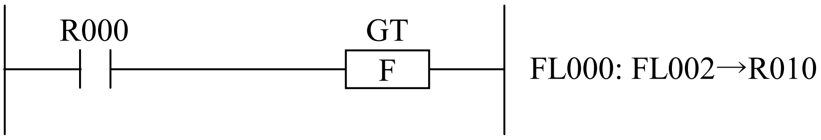
[Word Constant]



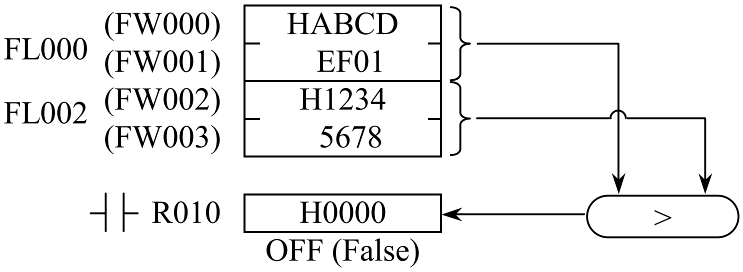
If the input condition, R000, is ON, the sizes of the contents of FW000 and immediate data, 10, are compared and the result is stored in R010.



[Long Word]



If the input condition, R000, is ON, the sizes of the contents of FL000 and FL002 are compared and the result is stored in R010.



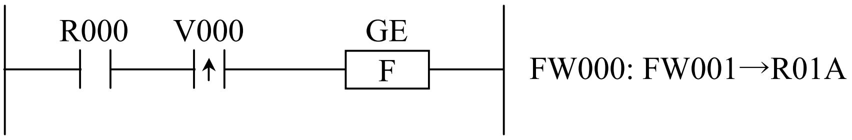
GE: GREATER OR EQUAL (≥)

<table>
<tr><td>Function</td><td colspan="12">GE performs a comparison on the sizes of the contents of the source and destination, or immediate data, and stores the result in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="3">Word</td><td colspan="3">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td colspan="2">Registers with immediate data</td><td>Registers</td><td colspan="2">Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td colspan="2">√</td><td>√</td><td colspan="2">na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>GE
—[F]—</td><td colspan="9">SW: DW→RW</td></tr>
<tr><td>Registers with immediate data</td><td>GE
—[F]—</td><td colspan="9">SW: nW→RW</td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>GE
—[F]—</td><td colspan="9">SL: DL→RL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="10"></td></tr>
<tr><td colspan="12">S: Address storing the source W: Word
D: Address storing the destination L: Long Word
n: Immediate data
R: Address storing the result (result of operation)</td></tr>
<tr><td rowspan="3">Contents of Process</td><td>Word</td><td colspan="11">When (S)W ≥ (D)W: 1→(R)W
When (S)W < (D)W: 0→(R)W</td></tr>
<tr><td>Word Constant</td><td colspan="11">When (S)W ≥ n W: 1→(R)W
When (S)W < n W: 0→(R)W</td></tr>
<tr><td>Long Word</td><td colspan="11">When (S)L ≥ (D)L: 1→(R)W
When (S)L < (D)L: 0→(R)W</td></tr>
<tr><td>Input Procedure</td><td colspan="12">([Shift] + [F]) [G] [E] ␣ Parameter, Parameter, Parameter [Enter]</td></tr>
<tr><td>Notes</td><td colspan="12">All flags remain intact.</td></tr>
</table>

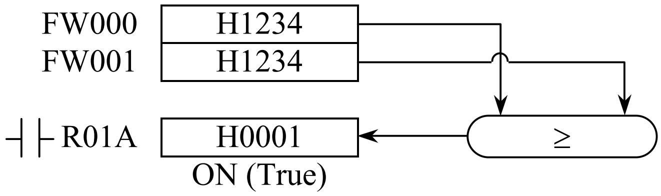
√: Applicable

na: Not applicable

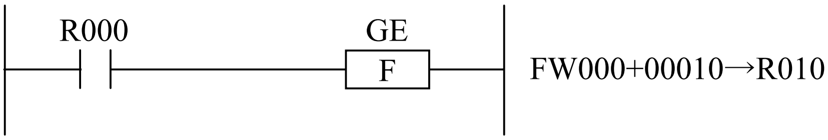
[Word]



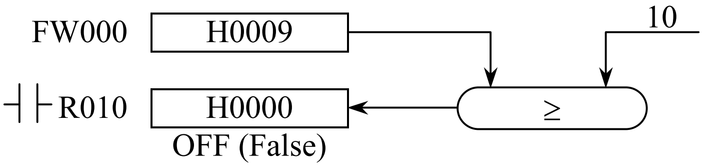
The sizes of the contents of FW000 and FW001 are compared only once when the input condition, R000, changes from OFF to ON, and the result is stored in R01A.



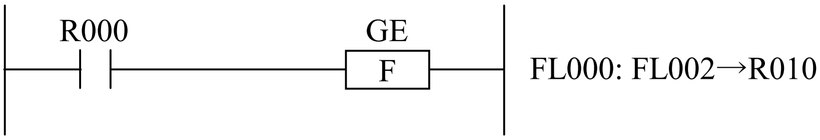
[Word Constant]



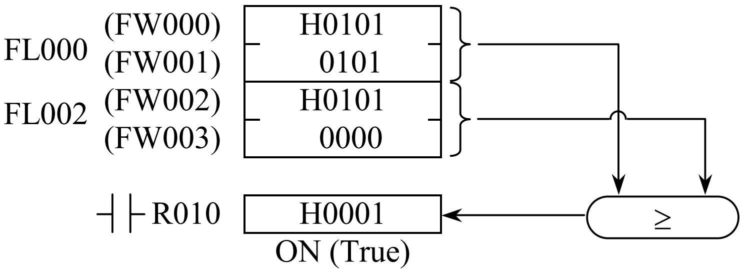
If the input condition, R000, is ON, the sizes of the contents of FW000 and immediate data, 10, are compared and the result is stored in R010.



[Long Word]



If the input condition, R000, is ON, the sizes of the contents of FL000 and FL002 are compared and the result is stored in R010.

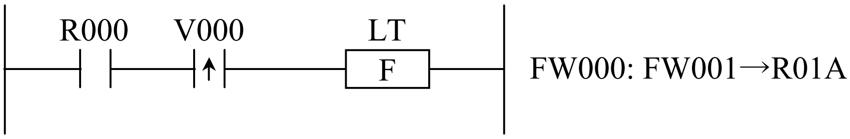


<table>
<tr><td>Function</td><td colspan="10">LT performs a comparison on the sizes of the contents of the source and destination, or immediate data, and stores the result in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>√</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td colspan="2">Registers</td><td>LT
┤F├</td><td colspan="6">SW: DW→RW</td></tr>
<tr><td colspan="2">Registers with immediate data</td><td>LT
┤F├</td><td colspan="6">SW: nW→RW</td></tr>
<tr><td rowspan="2">Long Word</td><td colspan="2">Registers</td><td>LT
┤F├</td><td colspan="6">SL: DL→RL</td></tr>
<tr><td colspan="2">Registers with immediate data</td><td colspan="7"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
D: Address storing the destination L: Long Word
n: Immediate data
R: Address storing the result (result of operation)</td></tr>
<tr><td rowspan="3">Contents of Process</td><td colspan="2">Word</td><td colspan="8">When (S)W < (D)W: 1→(R)W
When (S)W ≥ (D)W: 0→(R)W</td></tr>
<tr><td colspan="2">Word Constant</td><td colspan="8">When (S)W < n W: 1→(R)W
When (S)W ≥ n W: 0→(R)W</td></tr>
<tr><td colspan="2">Long Word</td><td colspan="8">When (S)L < (D)L: 1→(R)W
When (S)L ≥ (D)L: 0→(R)W</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) L T ␣ Parameter, Parameter, Parameter Enter</td></tr>
<tr><td>Notes</td><td colspan="10">All flags remain intact.</td></tr>
</table>

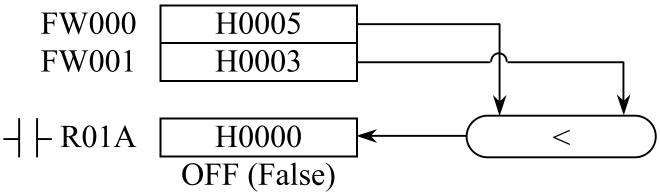
√: Applicable

na: Not applicable

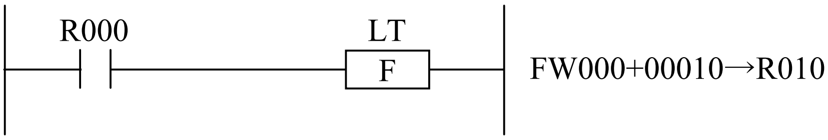
[Word]



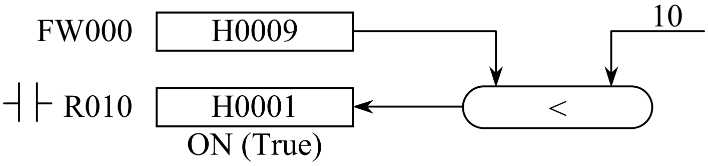
The sizes of the contents of FW000 and FW001 are compared only once when the input condition, R000, changes from OFF to ON, and the result is stored in R01A.



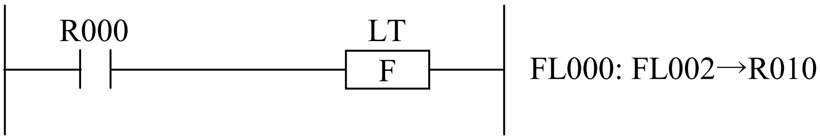
[Word Constant]



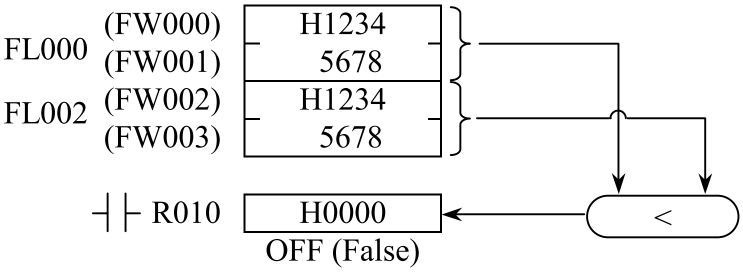
If the input condition, R000, is ON, the sizes of the contents of FW000 and immediate data, 10, are compared and the result is stored in R010.



[Long Word]



If the input condition, R000, is ON, the sizes of the contents of FL000 and FL002 are compared and the result is stored in R010.



LE: LESS OR EQUAL (≤)

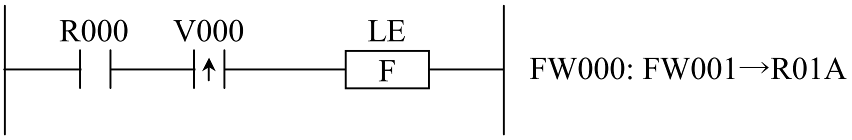
<table>
<tr><td>Function</td><td colspan="10">LE performs a comparison on the sizes of the contents of the source and destination, or immediate data, and stores the result in Result.</td></tr>
<tr><td>Parameter
Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td></td><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td></td><td>√</td><td>√</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
</table>

<table>
<tr><td>Parameter</td><td>Word</td><td>Registers</td><td>LE
┤F├</td><td>SW: DW→RW</td></tr>
<tr><td></td><td></td><td>Registers with immediate data</td><td>LE
┤F├</td><td>SW: nW→RW</td></tr>
<tr><td></td><td>Long Word</td><td>Registers</td><td>LE
┤F├</td><td>SL: DL→RL</td></tr>
<tr><td></td><td></td><td>Registers with immediate data</td><td colspan="2"></td></tr>
<tr><td></td><td colspan="4">S: Address storing the source W: Word
D: Address storing the destination L: Long Word
n: Immediate data
R: Address storing the result (result of operation)</td></tr>
<tr><td>Contents of Process</td><td>Word</td><td colspan="3">When (S)W ≤ (D)W: 1→(R)W
When (S)W > (D)W: 0→(R)W</td></tr>
<tr><td></td><td>Word Constant</td><td colspan="3">When (S)W ≤ n W: 1→(R)W
When (S)W > n W: 0→(R)W</td></tr>
<tr><td></td><td>Long Word</td><td colspan="3">When (S)L ≤ (D)L: 1→(R)W
When (S)L > (D)L: 0→(R)W</td></tr>
<tr><td>Input Procedure</td><td colspan="4">(Shift + F) L E ⌴ Parameter, Parameter, Parameter Enter</td></tr>
<tr><td>Notes</td><td colspan="4">All flags remain intact.</td></tr>
</table>

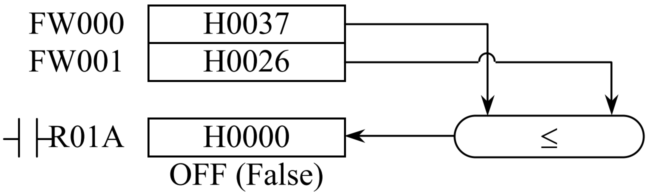
√: Applicable

na: Not applicable

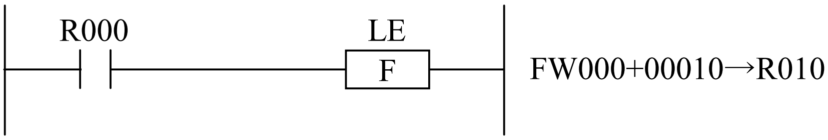
[Word]



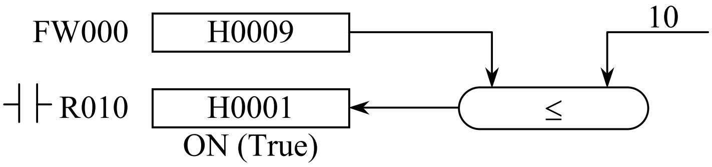
The sizes of the contents of FW000 and FW001 are compared only once when the input condition, R000, changes from OFF to ON, and the result is stored in R01A.



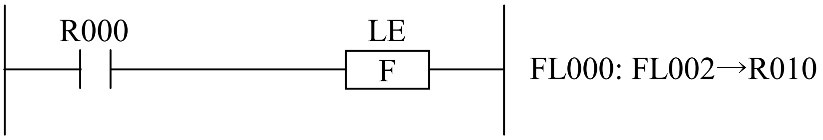
[Word Constant]



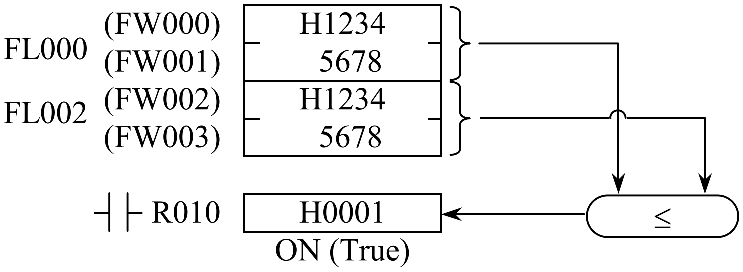
If the input condition, R000, is ON, the sizes of the contents of FW000 and immediate data, 10, are compared and the result is stored in R010.



[Long Word]



If the input condition, R000, is ON, the sizes of the contents of FL000 and FL002 are compared and the result is stored in R010.



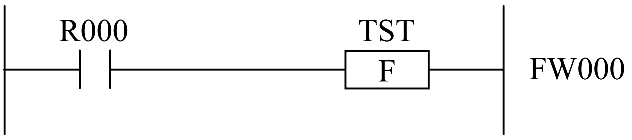
TST: TEST

<table>
<tr><td>Function</td><td colspan="10">TST makes a test of the polarity of the contents of the source and sets the positive (P), negative (N), and zero (Z) flags, respectively.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>na</td><td>√</td><td>na</td><td>-</td><td>-</td><td>↕</td><td>↕</td><td>↕</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>TST
F</td><td colspan="7">SW</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>TST
F</td><td colspan="7">SL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="5">S: Address storing the source</td><td colspan="5">W: Word
L: Long Word</td></tr>
<tr><td rowspan="2">Contents of Process</td><td>Word</td><td colspan="9">(S)W>0: P ON (N, Z OFF)
(S)W=0: Z ON (P, N OFF)
(S)W<0: N ON (P, Z OFF)</td></tr>
<tr><td>Long Word</td><td colspan="9">(S)L>0: P ON (N, Z OFF)
(S)L=0: Z ON (P, N OFF)
(S)L<0: N ON (P, Z OFF)</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) T S T ␣ Parameter Enter</td></tr>
<tr><td>Notes</td><td colspan="10">All flags except P, N, and Z remain intact.</td></tr>
</table>

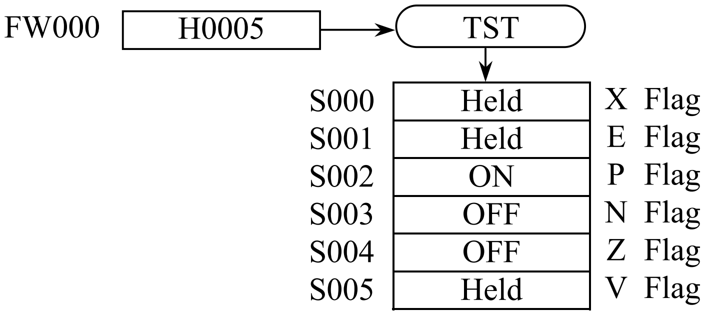
√: Applicable

na: Not applicable

[Word]

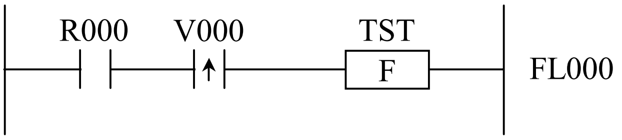


If the input condition, R000, is ON, the polarity of the contents of FW000 is tested to set the appropriate flag.

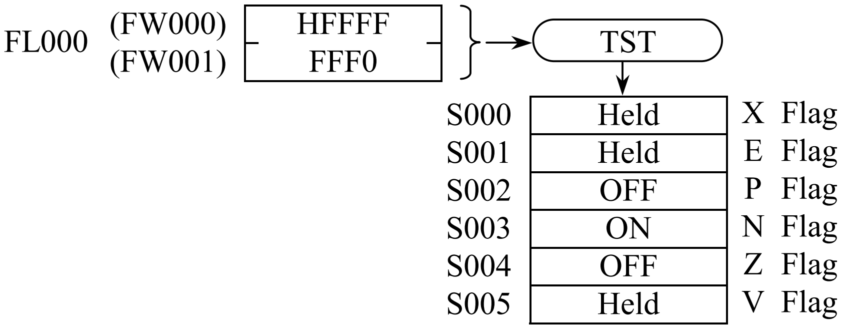


S000	Held	X Flag
S001	Held	E Flag
S002	ON	P Flag
S003	OFF	N Flag
S004	OFF	Z Flag
S005	Held	V Flag

[Long Word]



The polarity of the contents of FL000 is tested to set the appropriate flag, when the input condition, R000, changes from OFF to ON.



S000	Held	X Flag
S001	Held	E Flag
S002	OFF	P Flag
S003	ON	N Flag
S004	OFF	Z Flag
S005	Held	V Flag

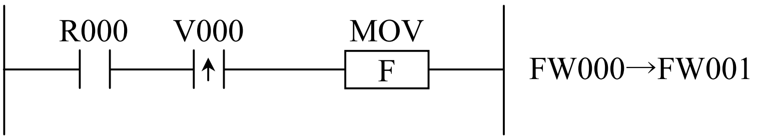
MOV: MOVE

Function	MOV moves the contents of the source to the destination.									
Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	na	√	na	-	-	-	-	-	-
Parameter	Word	Registers	MOV –[F]–	SW→DW						
		Registers with immediate data								
	Long Word	Registers	MOV –[F]–	SL→DL						
		Registers with immediate data								
	S: Address storing the source R: Address storing the result			W: Word L: Long Word						
Contents of Process	Word	(S)W→(D)W								
	Long Word	(S)L→(D)L								
Input Procedure	(Shift + F) M O V ⌴Parameter, Parameter Enter									
Notes	All flags remain intact.									

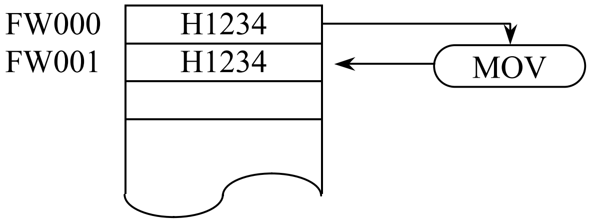
√: Applicable

na: Not applicable

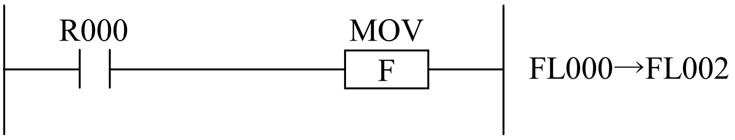
[Word]



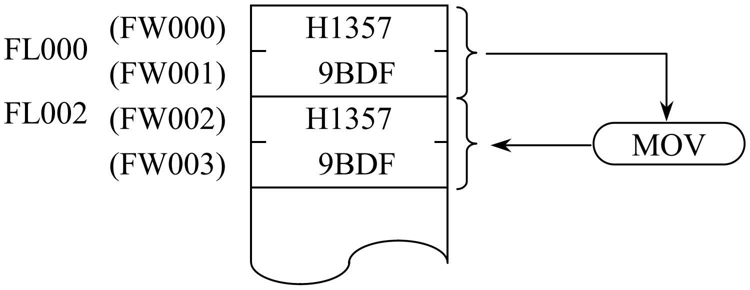
The contents of FW000 is moved into FW001 only once when the input condition, R000, changes from OFF to ON.



[Long Word]



If the input condition, R000, is ON, the contents of FL000 is moved into FL002.



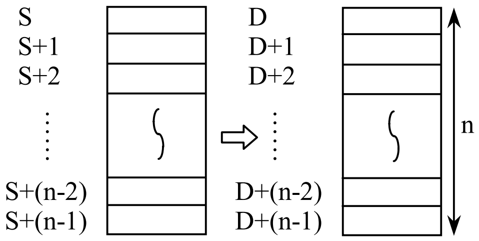
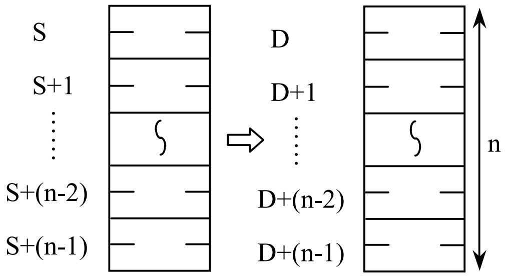
MOM: MOVE MULTI (Block Move)

Function	MOM moves n steps from the top in the source into the n steps from the top in the destination.

Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	na	√	na	-	-	-	-	-	-

Parameter				
	Word	Registers	MOM F	SW: nW→DW
		Registers with immediate data		
	Long Word	Registers	MOM F	SL: nW→DL
		Registers with immediate data		

S: Address storing the source	W: Word
R: Address storing the result	L: Long Word
D: Address storing the destination	

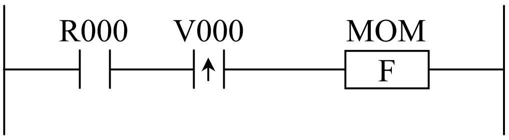
Contents of Process	Word	Long Word
	 S S+1 S+2 S+(n-2) S+(n-1) D D+1 D+2 D+(n-2) D+(n-1) n	 S S+1 S+(n-2) S+(n-1) D D+1 D+(n-2) D+(n-1) n

Input Procedure	(Shift + F) M O M ⌴Parameter, Parameter, Parameter Enter
Notes	• All flags remain intact. • No operation, if the number of steps n≤0 or 256<n.

√: Applicable

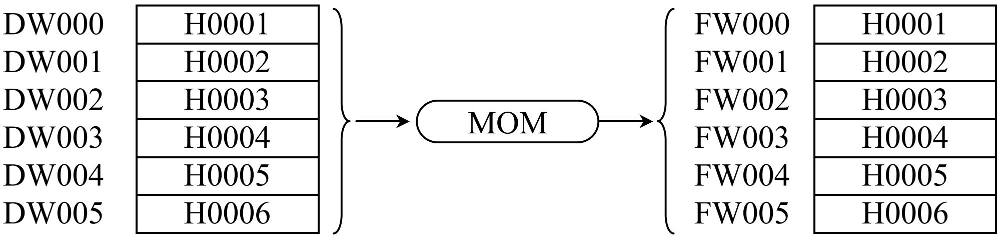
na: Not applicable

[Word]



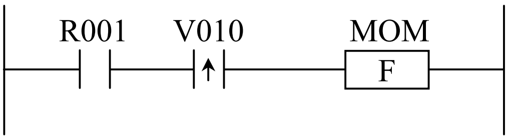
DW000: H006→FW000

Six steps from DW000 are moved to the six steps from FW00 only once when the input condition, R000, changes from OFF to ON.



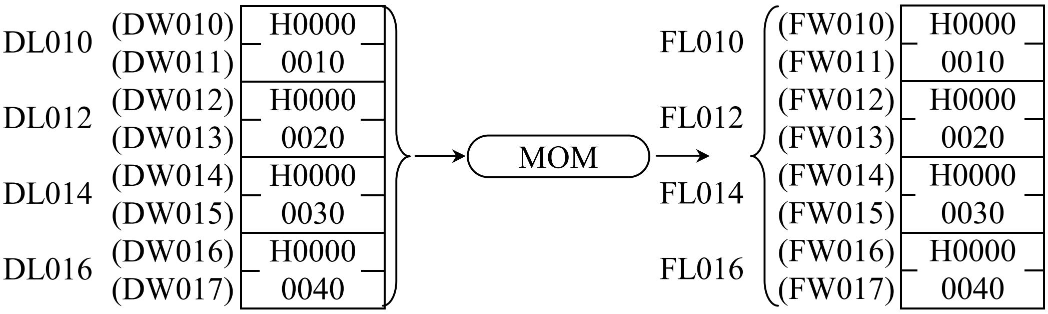
DW000	H0001	FW000	H0001
DW001	H0002	FW001	H0002
DW002	H0003	FW002	H0003
DW003	H0004	FW003	H0004
DW004	H0005	FW004	H0005
DW005	H0006	FW005	H0006

[Long Word]



DL010: +00004→FL010

Four steps from DL010 are moved to the four steps from FL010 only once when the input condition, R001, changes from OFF to ON.



DL010	(DW010)	H0000	FL010	(FW010)	H0000
	(DW011)	0010		(FW011)	0010
DL012	(DW012)	H0000	FL012	(FW012)	H0000
	(DW013)	0020		(FW013)	0020
DL014	(DW014)	H0000	FL014	(FW014)	H0000
	(DW015)	0030		(FW015)	0030
DL016	(DW016)	H0000	FL016	(FW016)	H0000
	(DW017)	0040		(FW017)	0040

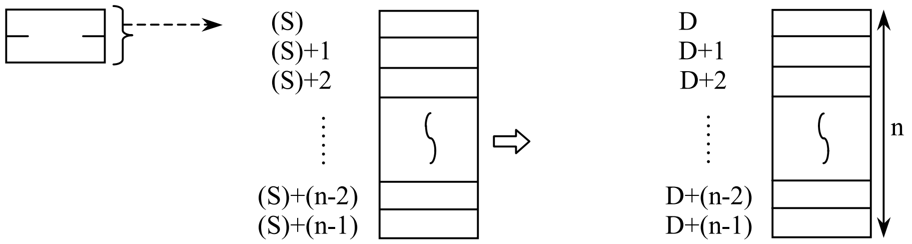
MSI: MOVE MULTI SOURCE INDIRECT (Source Indirect Move)

Function	MSI moves n steps from the top in the source (indirect mode) into the n steps from the top in the destination.									
Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	na	na	na	-	-	-	-	-	-

Parameter	Word	Registers	MSI F	SL: nW→DW
		Registers with immediate data		
	Long Word	Registers		
		Registers with immediate data		

S: Address storing the source W: Word
D: Address storing the destination L: Long Word
n: Immediate data
R: Address storing the result (result of operation)

Contents of Process — Word



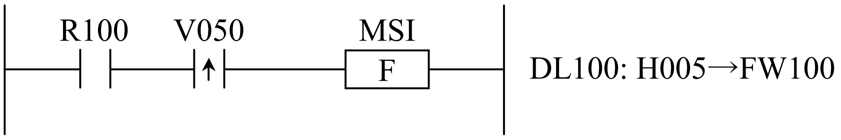
Input Procedure: (Shift + F) M S I ␣Parameter, Parameter, Parameter Enter

Notes

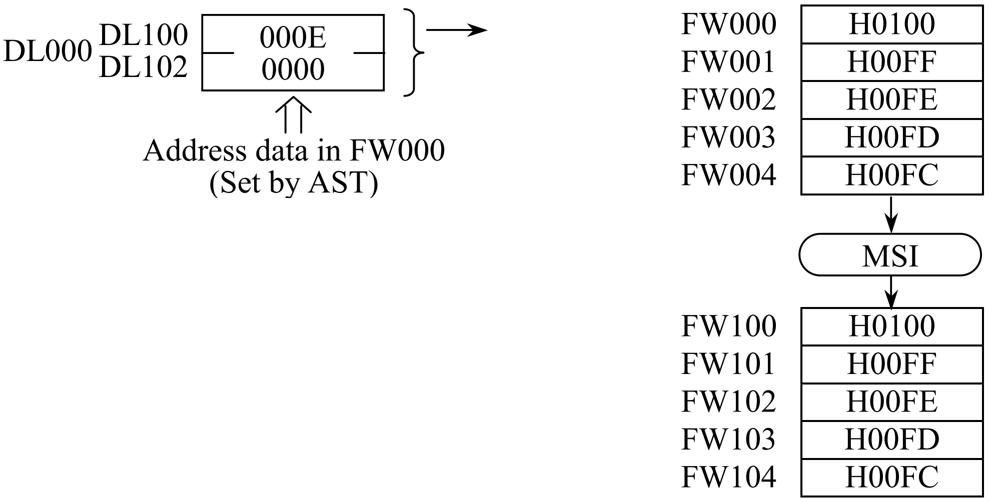
- All flags remain intact.
- No operation, if the number of steps n≤0 or 256<n.

√: Applicable
na: Not applicable

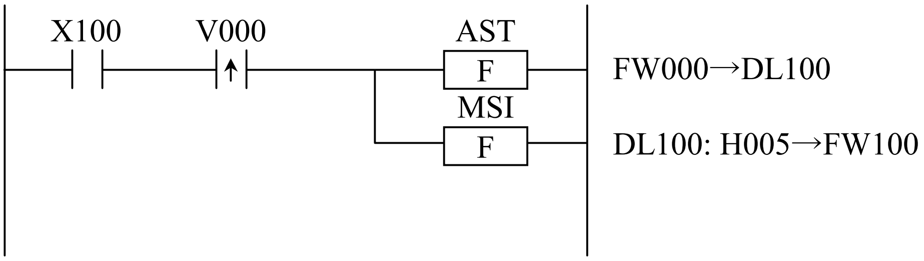
[Word]



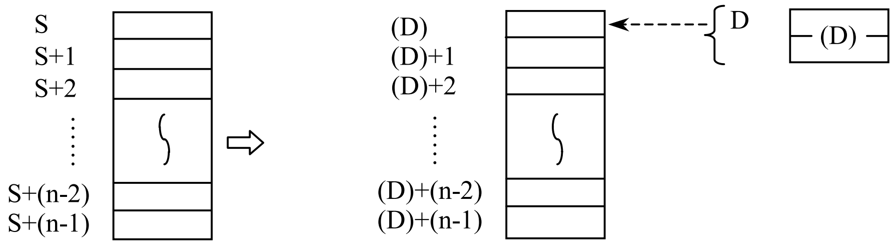
Five steps from the address data stored in DL100 are moved to the five steps from FW100 only once when the input condition, R100, changes from OFF to ON.



<Circuit combined with AST>



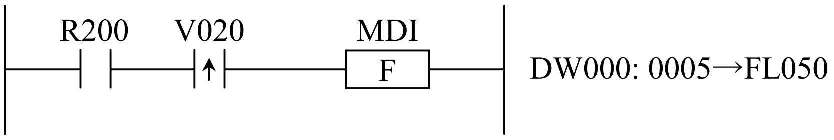
MDI: MOVE MULTI DESTINATION INDIRECT (Destination Indirect Move)

<table>
<tr><td>Function</td><td colspan="10">MDI moves n steps from the top in the source into the n steps from the top in the destination (indirect mode).</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>na</td><td>na</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>MDI
F</td><td colspan="7">SW: nW→DL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td colspan="8"></td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
n: Immediate data L: Long Word
D: Address storing the destination</td></tr>
<tr><td rowspan="2">Contents of Process</td><td colspan="10">Word</td></tr>
<tr><td colspan="10"> </td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) M D I ␣Parameter, Parameter, Parameter Enter</td></tr>
<tr><td>Notes</td><td colspan="10">• All flags remain intact.
• No operation, if the number of steps n≤0 or 256&lt;n.</td></tr>
</table>

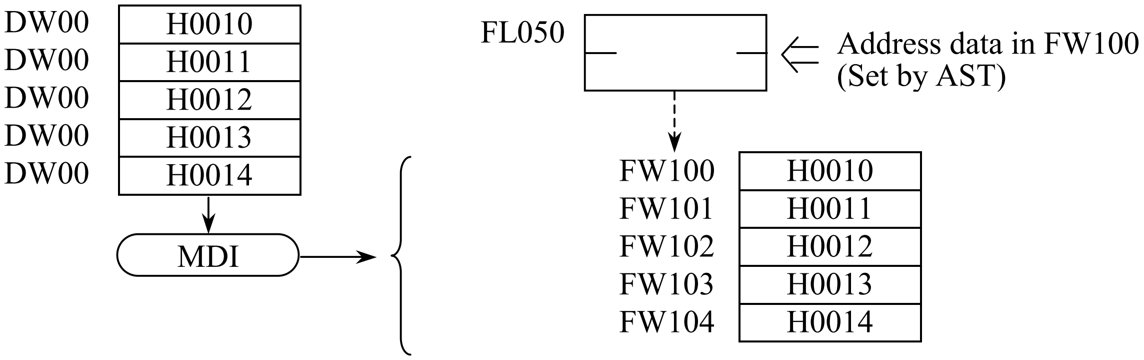
√: Applicable

na: Not applicable

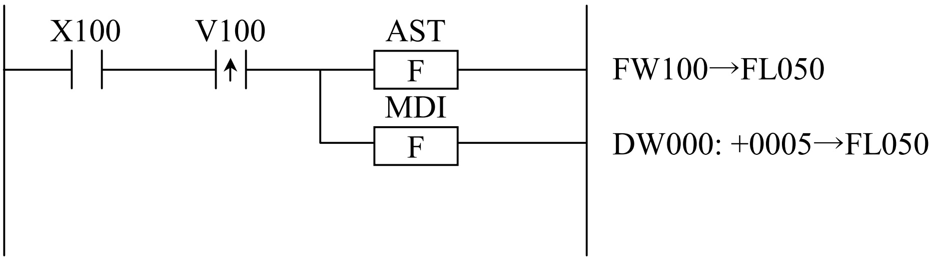
[Word]



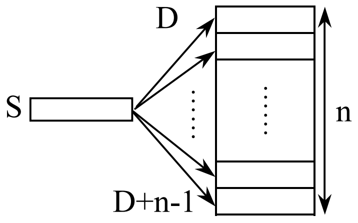
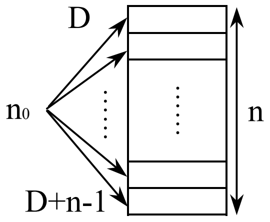
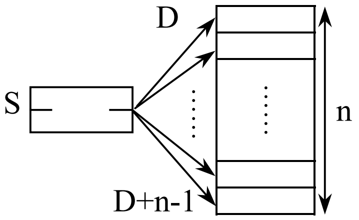
Five steps from DW000 are moved to the five steps from the address data stored in FW050 only once when the input condition, R200, changes from OFF to ON.



<Circuit combined with AST>



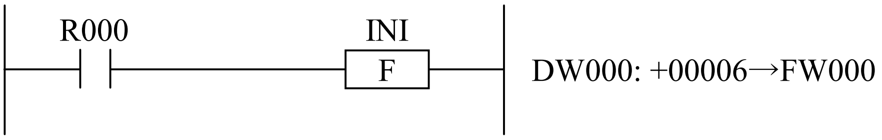
INI: INITIAL (Same Data Block Move)

<table>
<tr><td>Function</td><td colspan="10">INI moves the contents of the source or immediate data, n_0, into the n steps from the top in the destination.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>√</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td colspan="2">Registers</td><td>INI
-[F]-</td><td colspan="6">SW: nW→DW</td></tr>
<tr><td colspan="2">Registers with immediate data</td><td>INI
-[F]-</td><td colspan="6">n1W: n2W→DW</td></tr>
<tr><td rowspan="2">Long Word</td><td colspan="2">Registers</td><td>INI
-[F]-</td><td colspan="6">SL: nW→DL</td></tr>
<tr><td colspan="2">Registers with immediate data</td><td colspan="7"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
n_0, n: Immediate data L: Long Word
D: Address storing the destination</td></tr>
<tr><td rowspan="2">Contents of Process</td><td colspan="3">Word</td><td colspan="3">Word Constant</td><td colspan="4">Long Word</td></tr>
<tr><td colspan="3"> </td><td colspan="3"> </td><td colspan="4"> </td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) I N I ␣Parameter, Parameter, Parameter Enter</td></tr>
<tr><td>Notes</td><td colspan="10">• All flags remain intact.
• No operation, if the number of steps n≤0 or 256&lt;n.</td></tr>
</table>

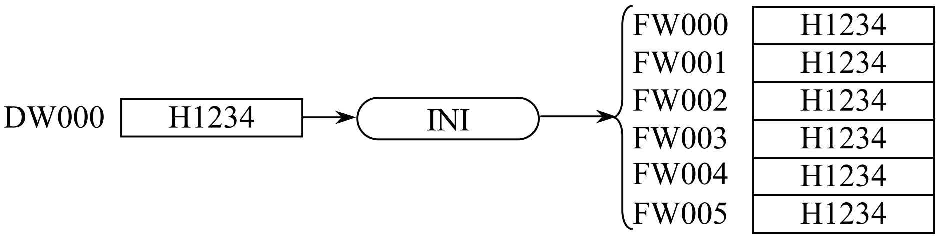
√: Applicable

na: Not applicable

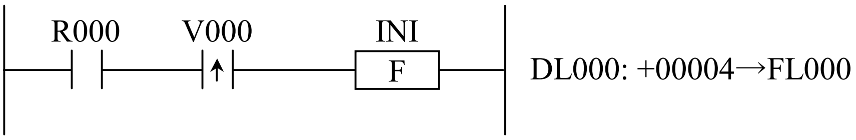
[Word]



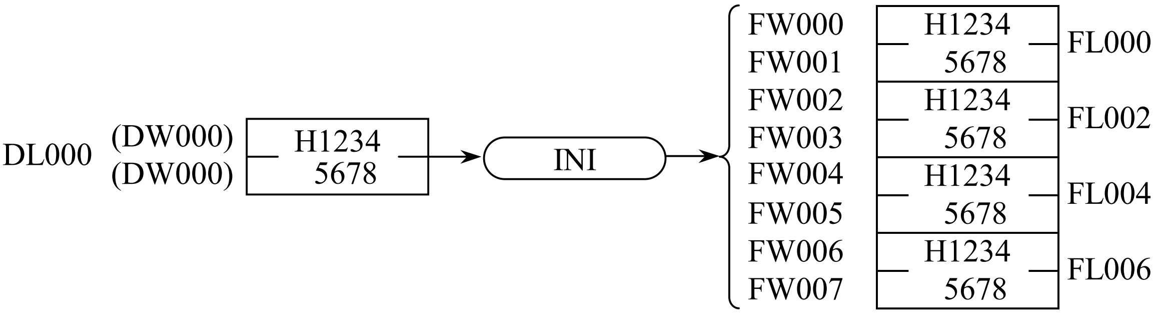
If the input condition, R000, is ON, the contents of DW000 are moved into the 6 steps from FW000.



[Long Word]



The contents of DL000 are moved into the 4 steps from FL000 when the input condition, R000, changes from OFF to ON.



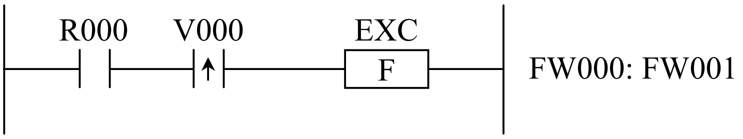
EXC: EXCHANGE

<table>
<tr><td>Function</td><td colspan="10">EXC exchanges the contents of the source for the contents of the destination.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>na</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>EXC
─[F]─</td><td colspan="7">SW: DW</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>EXC
─[F]─</td><td colspan="7">SL: DL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
D: Address storing the destination L: Long Word</td></tr>
<tr><td rowspan="3">Contents of Process</td><td>Word</td><td colspan="9">(S)W ↔ (D)W</td></tr>
<tr><td>Word Constant</td><td colspan="9">(S)L ↔ (D)L</td></tr>
<tr><td>Long Word</td><td colspan="9"></td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) E X C ␣Parameter, Parameter Enter</td></tr>
<tr><td>Notes</td><td colspan="10">All flags remain intact.</td></tr>
</table>

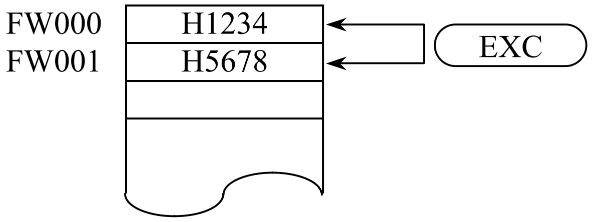
√: Applicable

na: Not applicable

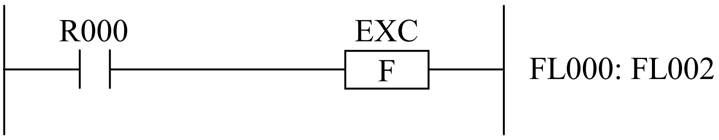
[Word]



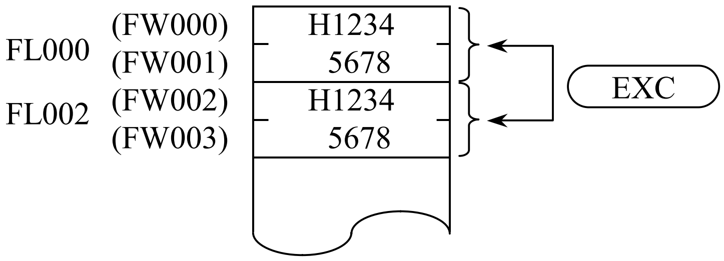
The contents of FW000 are exchanged for the contents of FW001 only once when the input condition, R000, changes from OFF to ON.



[Long Word]

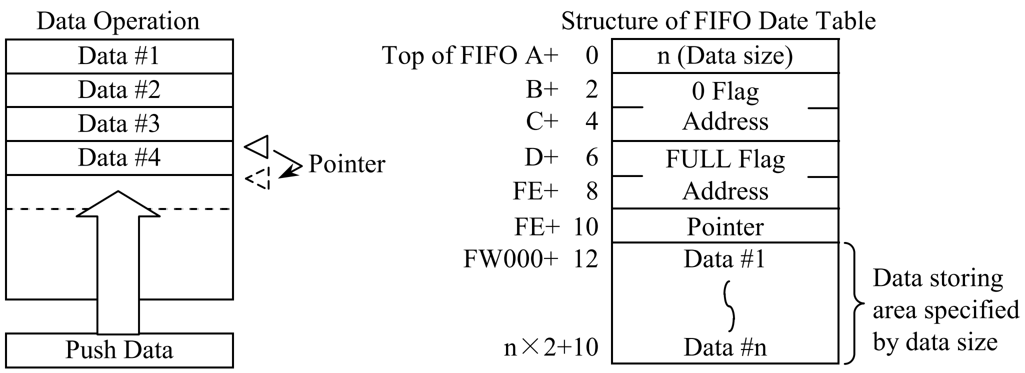


If the input condition, R000, is ON, the contents of FL000 are exchanged for the contents of FL002.



Note: Be careful of address overlap, when specifying a long word.

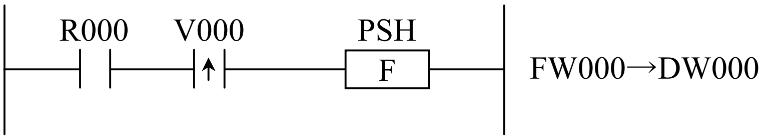
PSH: FIFO PUSH (FIFO Write)

<table>
<tr><td>Function</td><td colspan="10">PSH pushes the contents of the source into the FIFO table.</td></tr>
<tr><td rowspan="3">Parameter
Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>na</td><td>na</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>PSH
F</td><td colspan="7">SW→TBW</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td colspan="8"></td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
TB: Head address of the FIFO table</td></tr>
<tr><td>Contents of Process</td><td colspan="10">
</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) P S H ⌴Parameter, Parameter Enter</td></tr>
<tr><td>Notes</td><td colspan="10">• The 0 flag is turned OFF.
• When the pointer is n prior to the push processing, the push operation is not performed with the FULL flag turned ON. If the pointer is incremented to n after the push operation, the FULL flag is also turned ON. The FULL flag is turned OFF, otherwise.
• Push operation is not performed, if data size, n≤0 or 256<n.
• Push operation is not performed, if the pointer<0 or n<pointer.
• All flags (X, E, P, N, Z, and V) remain intact.</td></tr>
</table>

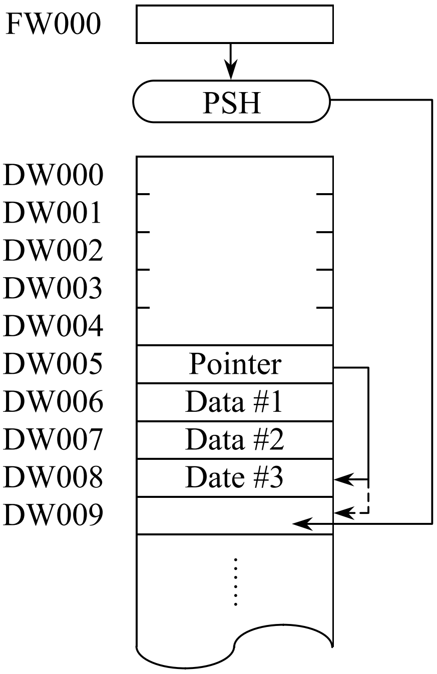
√: Applicable

na: Not applicable

[Word]

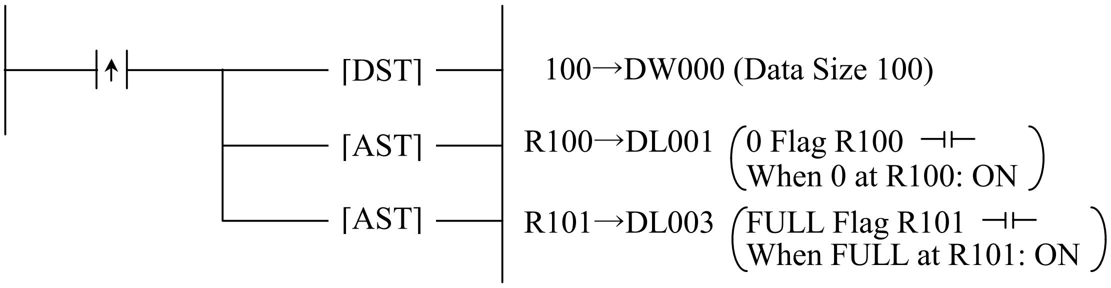


The contents of FW000 are set into the FIFO data table close to DW000 only once when the input condition, R000, changes from OFF to ON.



- Data Size, n, is set by Data Set, DST, etc.
- The addresses of 0 and FULL flags are set by Address Set, AST.

Example:

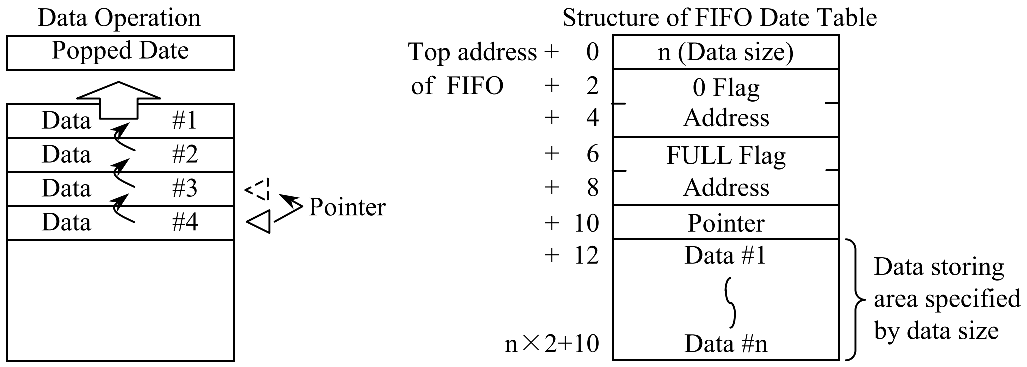


POP: FIFO POP (FIFO Read)

Function	POP stores the popped data in the destination through the pop processing of the FIFO table.

Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	na	na	na	-	-	-	-	-	-

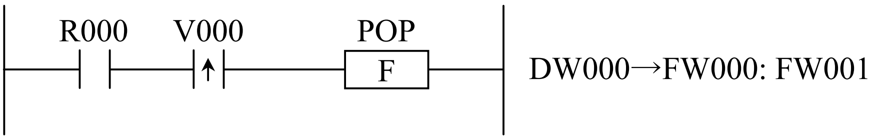
Parameter				
	Word	Registers	POP -[F]-	TBW→DW
		Registers with immediate data		
	Long Word	Registers		
		Registers with immediate data		
	TB: Head address of the FIFO table W: Word D: Address storing the destination			

Contents of Process	 Data Operation Popped Date Data #1 Data #2 Data #3 Data #4 Pointer Structure of FIFO Date Table Top address of FIFO + 0 + 2 + 4 + 6 + 8 + 10 + 12 n×2+10 n (Data size) 0 Flag Address FULL Flag Address Pointer Data #1 Data #n Data storing area specified by data size
Input Procedure	(Shift + F) P O P ␣Parameter, Parameter Enter
Notes	• The FULL flag is turned OFF. • When the pointer is 0 prior to the pop operation, the pop operation is not performed with the 0 flag turned ON. If the pointer is decremented to 0 after the pop operation, the 0 flag is also turned ON. The 0 flag is turned OFF, otherwise. • Pop operation is not performed if data size, n≤0 or 256<n. • Pop operation is not performed if the pointer<0 or n<pointer. • All the flags (X, E, P, N, Z, and V) remain intact.

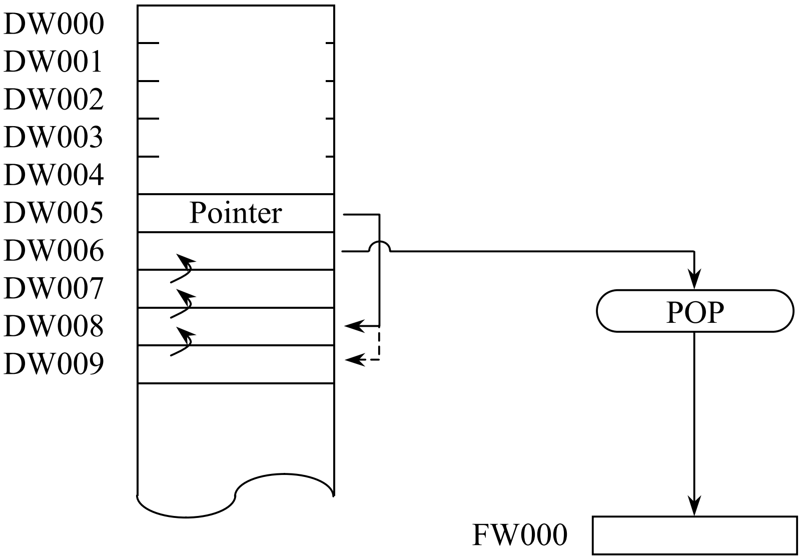
√: Applicable

na: Not applicable

[Word]



Data is read-in from the FIFO data table close to DW000 only once when the input condition, R000, changes from OFF to ON.



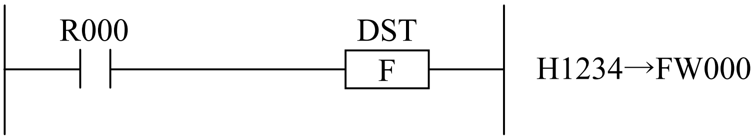
DST: DATA SET

Function	DST stores immediate data in the destination.									
Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	na	√	na	√	-	-	-	-	-	-
Parameter	Word	Registers								
		Registers with immediate data	DST ─┤F├─	nW→DW						
	Long Word	Registers								
		Registers with immediate data	DST ─┤F├─	nL→DL						
	n: Immediate data D: Address storing the destination				W: Word L: Long Word					
Contents of Process	Word	nW→(D)W								
	Long Word	nL→(D)L								
Input Procedure	(Shift + F) D S T ␣Parameter, Parameter Enter									
Notes	All flags remain intact.									

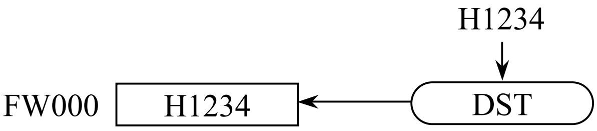
√: Applicable

na: Not applicable

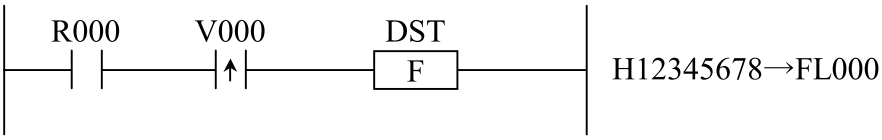
[Word Constant]



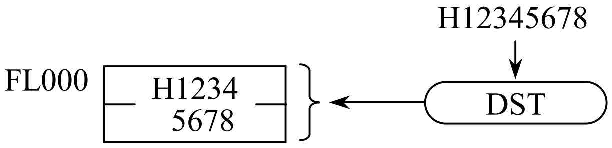
If the input condition, R000, is ON, immediate data, H1234, is stored in FW000.



[Long Word Constant]



Immediate data, H12345678 is stored in FL000 only once when the input condition, R000, changes from OFF to ON.



- When entering an immediate data in decimal notation, the data range will be -999999999≤ n≤+999999999.

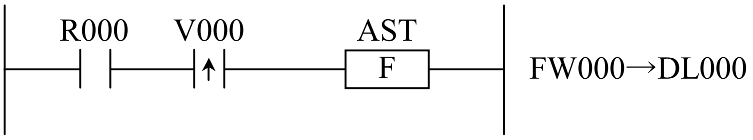
AST: ADDRESS SET

<table>
<tr><td>Function</td><td colspan="10">AST stores the address data in the destination.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>na</td><td>na</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td colspan="8"></td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>AST
─┤F├─</td><td colspan="7">SL: nW→DL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
n: Immediate data L: Long Word
D: Address storing the destination</td></tr>
<tr><td>Contents of Process</td><td colspan="10">AST SW (Address Data) → (D) Word Constant</td></tr>
<tr><td>Input Procedure</td><td colspan="10">([Shift] + [F]) [A] [S] [T] ␣Parameter, Parameter [Enter]</td></tr>
<tr><td>Notes</td><td colspan="10">All flags remain intact.</td></tr>
</table>

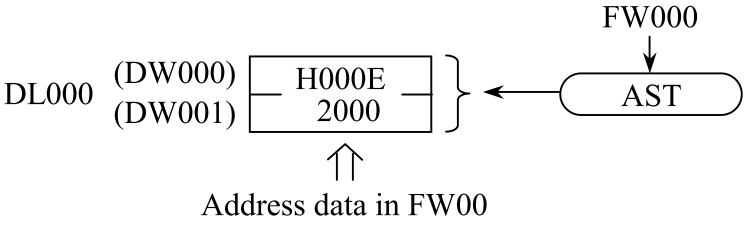
√: Applicable

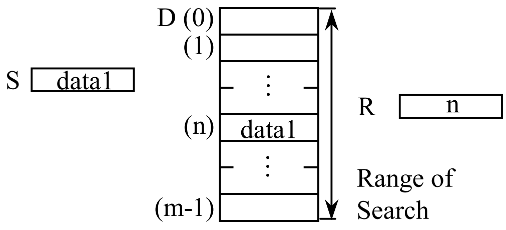
na: Not applicable

[Long Word]

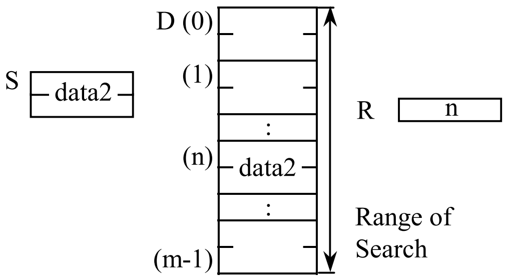


The address data of FW000 is stored in DL000 only once when the input condition, R000, changes from OFF to ON.



<table>
<tr><td>Function</td><td colspan="10">SCH searches a specified area at the destination (number of search steps = m) for the specified contents of the source and stores the number of passed steps from the top of the destination before finding the source.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>na</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td colspan="8"></td></tr>
<tr><td>Registers with immediate data</td><td>SCH
─[F]─</td><td colspan="7">SW: DW: nW→RW</td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td colspan="8"></td></tr>
<tr><td>Registers with immediate data</td><td>SCH
─[F]─</td><td colspan="7">SL: DL: nW→RW</td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
D: Address storing the destination L: Long Word
n: Immediate data
R: Address storing the result (result of operation)</td></tr>
<tr><td rowspan="2">Contents of Process</td><td colspan="5">Word</td><td colspan="5">Long Word</td></tr>
<tr><td colspan="5">

S
data1
D (0)
(1)
⋮
(n)
data1
⋮
(m-1)
R
n
Range of Search

</td><td colspan="5">

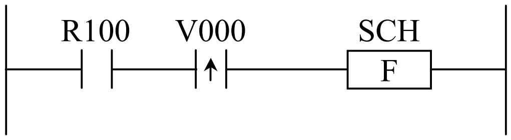
S
data2
D (0)
(1)
:
(n)
data2
:
(m-1)
R
n
Range of Search

</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) S C H
⌴ Parameter, Parameter, Parameter, Parameter Enter</td></tr>
<tr><td>Notes</td><td colspan="10">• All flags remain intact.
• Only the searched data found first is effective when the search starts from the top of the area.
• If no search data is found in the search area, -1 (HFFFF) is set in Result.
• No operation, if m (the number of search steps)≤0 or 256&lt;m</td></tr>
</table>

√: Applicable

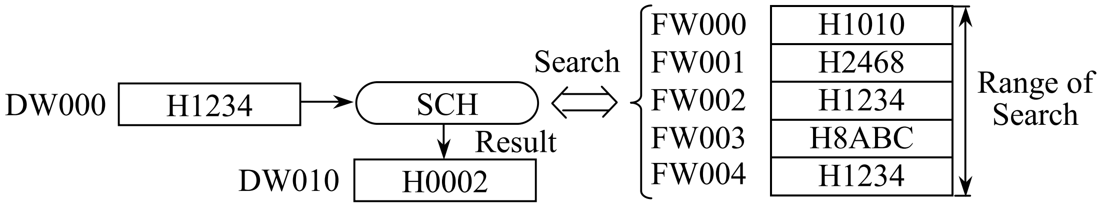
na: Not applicable

[Word]

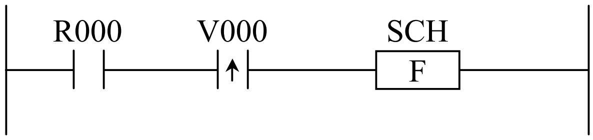


DW000: FW000: +0005→DW010

The 5 steps beginning with FW000 are searched for the same data as the contents of DW000 only once when the input condition, R000, changes from OFF to ON. The result is stored in DW010.

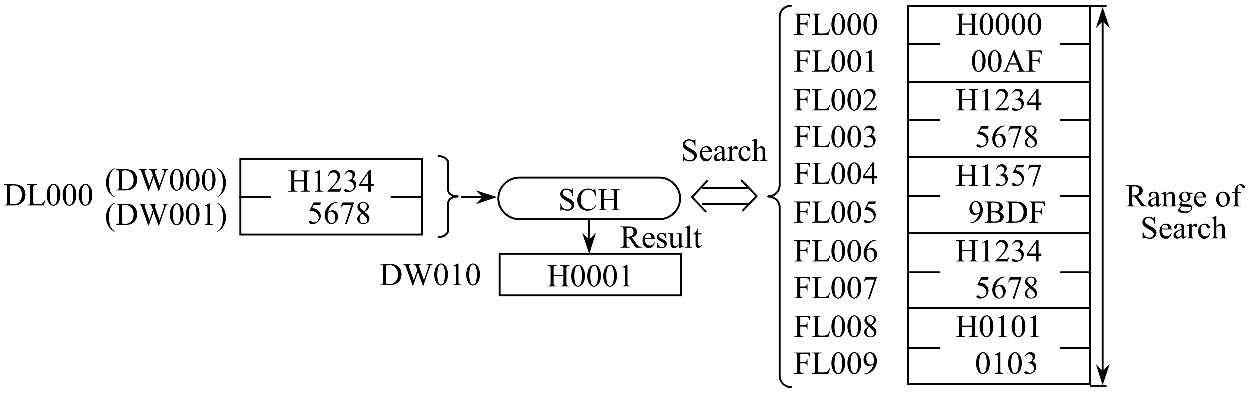


[Long Word]



DW000: FW000: +0005→DW010

The 5 steps beginning with FL000 are searched for the same data as the contents of DL000 only once when the input condition, R000, changes from OFF to ON. The result is stored in DW010.

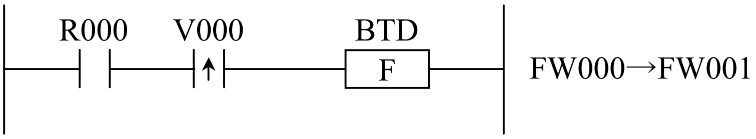


<table>
<tr><td>Function</td><td colspan="10">BTD converts the contents of the source from BIN (Binary) to BCD, and stores the result in Result.
BCD: Binary Coded Decimal number</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>na</td><td>√</td><td>na</td><td>-</td><td>↕</td><td>-</td><td>-</td><td>-</td><td>↕</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>BTD
F</td><td colspan="7">SW→RW</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>BTD
F</td><td colspan="7">SL→DL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
R: Address storing the result L: Long Word</td></tr>
<tr><td rowspan="2">Contents of Process</td><td>Word</td><td colspan="9">(S)W $\xrightarrow{\text{BIN→BCD}}$ (R)W</td></tr>
<tr><td>Long Word</td><td colspan="9">(S)L $\xrightarrow{\text{BIN→BCD}}$ (R)L</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) B T D ⌴Parameter, Parameter Enter</td></tr>
<tr><td>Flag Setting</td><td colspan="10">E: 1, if (S)<0 0, otherwise
V: Word ; 1, if (S)>9999 0, otherwise
Long word ; 1, if (S)>99999999 0, otherwise
Others: Not affected</td></tr>
<tr><td>Notes</td><td colspan="10">• When (S)<0, no process is made with the Error flag (E) turned ON and the Overflow flag (V), OFF. (The result is not affected.)
• The following full scale values are set in result, if an overflow occurs.
<table><tr><td>Word</td><td>Long Word</td></tr><tr><td>H9999</td><td>H99999999</td></tr></table></td></tr>
</table>

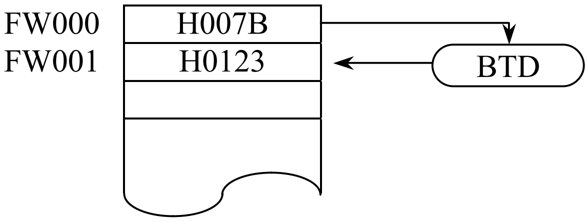
√: Applicable

na: Not applicable

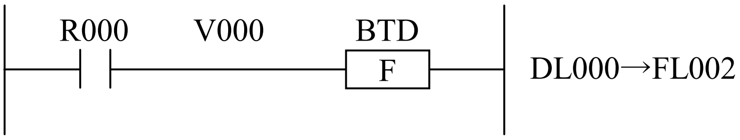
[Word]



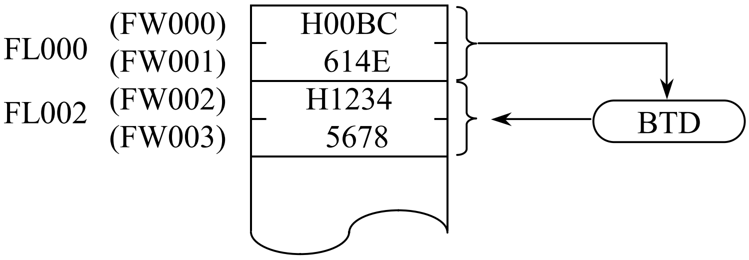
The contents of FW000 are converted from BIN to BCD only once when the input condition, R000, changes from OFF to ON, and the result is stored in FW001.



[Long Word]

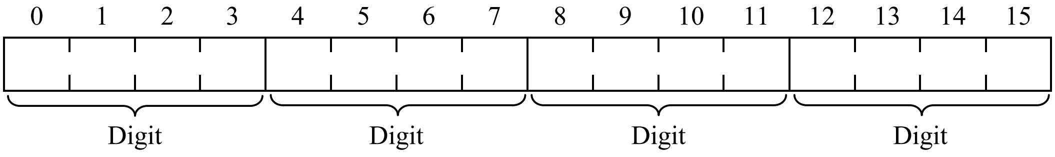


If the input condition, R000, is ON, the contents of FL000 are converted from BIN to BCD and the result is stored in FL002.



Function	DTB converts the contents of the source from BCD to BIN (Binary), storing the result in Result.									
Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	na	√	na	-	↕	-	-	-	-

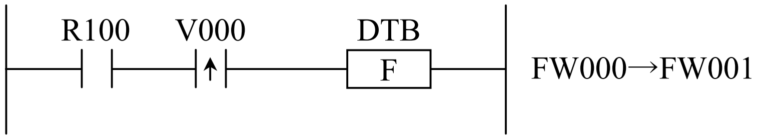
Parameter				
Parameter	Word	Registers	DTB ─[F]─	SW→RW
		Registers with immediate data		
	Long Word	Registers	DTB ─[F]─	SL→DL
		Registers with immediate data		
	S: Address storing the source D: Address storing the result		W: Word L: Long Word	

Contents of Process	Word	$(S)W \xrightarrow{\text{BCD}\rightarrow\text{BIN}} (R)W$
	Long Word	$(S)L \xrightarrow{\text{BCD}\rightarrow\text{BIN}} (R)L$
Input Procedure	(Shift + F) D T B ␣Parameter, Parameter Enter	
Flag Setting	E: When a certain digit (4 bits) in the source is HA to HF: 1 Otherwise: 0 Others: Not affected	
Notes	No operation if Error Flag turns ON. The result is not affected. <Digit>  0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 Digit Digit Digit Digit	

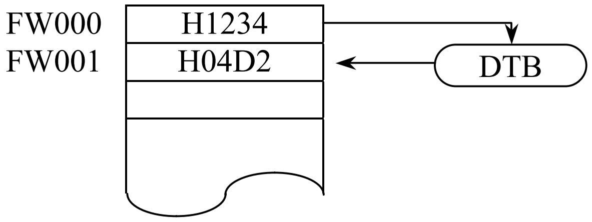
√: Applicable

na: Not applicable

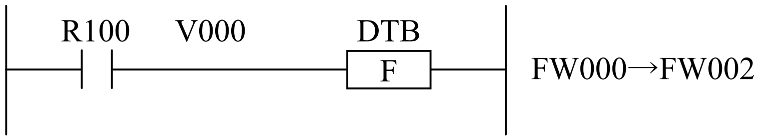
[Word]



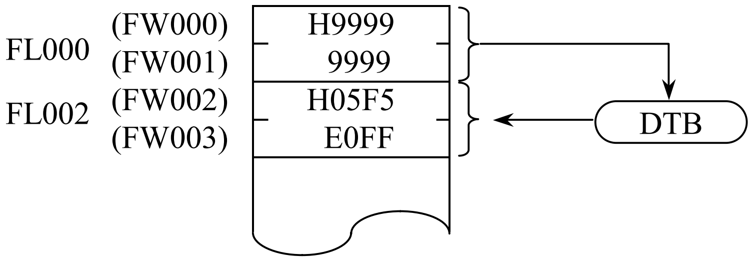
The contents of FW000 are converted from BCD to BIN only once when the input condition, R000, changes from OFF to ON, and the result is stored in FW001.



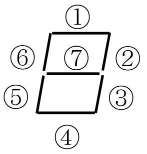
[Long Word]



If the input condition, R000, is ON, the contents of FL000 are converted from BCD to BIN and the result is stored in FL002.



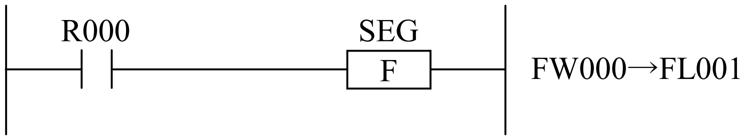
SEG: BINARY→SEGMENT CONVERSION

<table>
<tr><td>Function</td><td colspan="10">SEG converts the contents of the source or immediate data from BIN (Binary) to 7-segment data, and stores the result in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>√</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td colspan="2">Registers</td><td>SEG
F</td><td colspan="6">SW→RW</td></tr>
<tr><td colspan="2">Registers with immediate data</td><td>SEG
F</td><td colspan="6">nW→RW</td></tr>
<tr><td rowspan="2">Long Word</td><td colspan="2">Registers</td><td>SEG
F</td><td colspan="6">SL→RL</td></tr>
<tr><td colspan="2">Registers with immediate data</td><td colspan="7"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
n: Immediate data L: Long Word
R: Address storing the result</td></tr>
<tr><td rowspan="3">Contents of Process</td><td colspan="2">Word</td><td>(S)W</td><td colspan="5">$\xrightarrow{\text{BIN→7 Segment}}$</td><td colspan="2">(R, R+1)W</td></tr>
<tr><td colspan="2">Word Constant</td><td>nW</td><td colspan="5">$\xrightarrow{\text{BIN→7 Segment}}$</td><td colspan="2">(R, R+1)W</td></tr>
<tr><td colspan="2">Long Word</td><td>(S)L</td><td colspan="5">$\xrightarrow{\text{BIN→7 Segment}}$</td><td colspan="2">{(R)L, (R+2)L}</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) S E G ␣Parameter, Parameter Enter</td></tr>
<tr><td>Structure of Segment Data</td><td colspan="10">(MSB)
<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>0</td><td>①</td><td>②</td><td>③</td><td>④</td><td>⑤</td><td>⑥</td><td>⑦</td></tr></table> ⟹ 
• Displayed when the associated bit is ON.
• 7 Segment Correspondence Table
<table><tr><td>No.</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td></tr><tr><td>Display</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>A</td><td>b</td><td>C</td><td>d</td><td>E</td><td>F</td></tr><tr><td>Data</td><td>H7E</td><td>H30</td><td>H6D</td><td>H79</td><td>H33</td><td>H5B</td><td>H5F</td><td>H70</td><td>H7F</td><td>H7B</td><td>H77</td><td>H1F</td><td>H4E</td><td>H3D</td><td>H4F</td><td>H47</td></tr></table></td></tr>
<tr><td>Notes</td><td colspan="10">All flags remain intact.</td></tr>
</table>

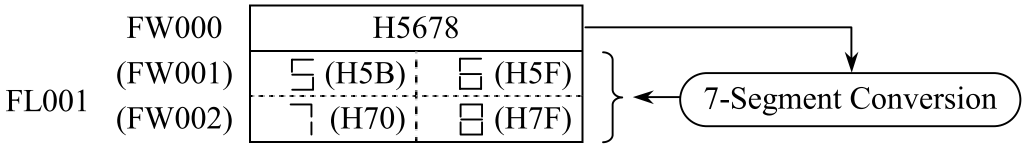
√: Applicable

na: Not applicable

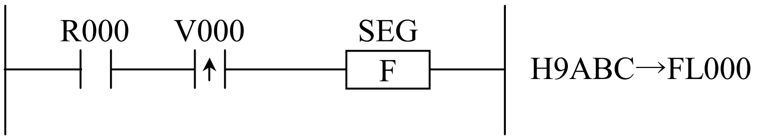
[Word]



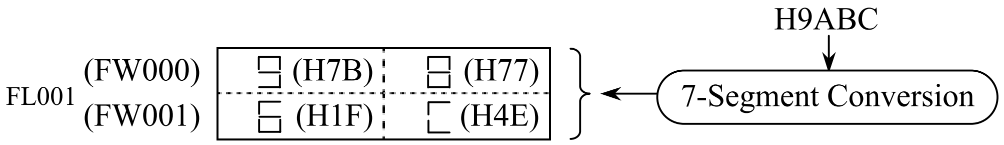
If the input condition, R000, is ON, the contents of FW000 are converted from binary data to 7-segment data of 4 characters. The result is stored in FL001.



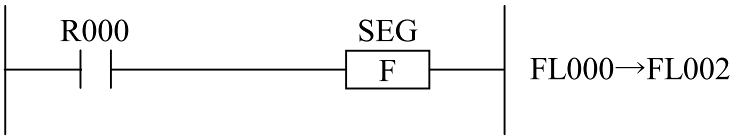
[Word Constant]



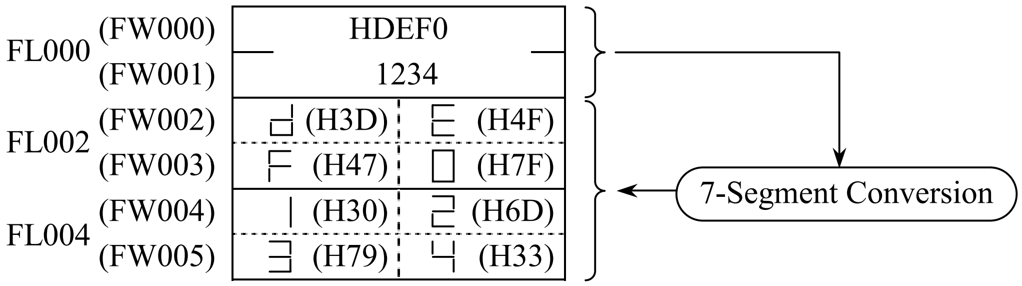
A binary data, H9ABC, is converted to 7-segment data of 4 characters only once when the input condition, R000, changes from OFF to ON. The result is stored in FL000.



[Long Word]



If the input condition, R000, is ON, the contents of FL000 are converted from binary data to 7-segment data of 8 characters. The result is stored in FL002.



ASP: BINARY→ASCII CONVERSION (PACKED MODE)

Function	ASP converts the contents of the source from a binary data to an ASCII data, and stores the result in packed mode in Result. ASCII: American Standard Code for Information Interchange

Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	na	na	na	-	-	-	-	-	-

Parameter				
	Word	Registers	ASP ─[F]─	SW→RW
		Registers with immediate data		
	Long Word	Registers		
		Registers with immediate data		
	S: Address storing the source R: Address storing the result		W: Word L: Long Word	

Contents of Process	Word	$(S)W \xrightarrow[\text{Packed Mode}]{\text{BIN}\rightarrow\text{ASCII}} (R, R+1)W$
Input Procedure	(Shift + F) A S P ␣Parameter, Parameter Enter	

ASCII Correspondence Table

Binary	0	1	2	3	4	5	6	7	8	9
ASCII	H30	H31	H32	H33	H34	H35	H36	H37	H38	H39

A	B	C	D	E	F
H41	H42	H43	H44	H45	H46

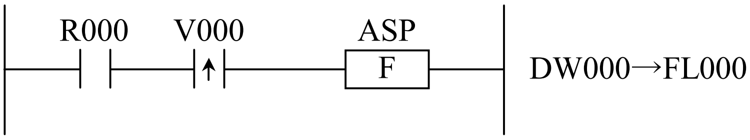
Notes

- All flags remain intact.
- The result is stored, 2 bytes in R (high-order) and another 2 bytes in R+1 (low-order).

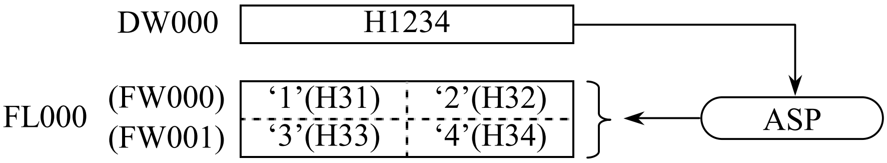
√: Applicable

na: Not applicable

[Word]



The contents of DW000 are converted from binary data to ASCII data only once when the input condition, R000, changes from OFF to ON. The result is stored in FL000.



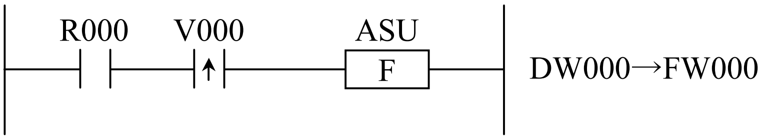
ASU: BINARY→ASCII CONVERSION (UNPACKED MODE)

<table>
<tr><td>Function</td><td colspan="10">ASU converts the contents of the source from a binary data to an ASCII data, and stores the result in unpacked mode in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>na</td><td>na</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>ASU
F</td><td colspan="7">SW→RW</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td colspan="8"></td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
R: Address storing the result L: Long Word</td></tr>
<tr><td>Contents of Process</td><td>Word</td><td colspan="9">(S)W $\xrightarrow[\text{Unpacked Mode}]{\text{BIN→ASCII}}$ (R, R+1, R+2, R+3)W</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) A S U ␣Parameter, Parameter Enter</td></tr>
<tr><td>ASCII Correspondence Table</td><td colspan="10">
<table>
<tr><td>Binary</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
<tr><td>ASCII</td><td>H30</td><td>H31</td><td>H32</td><td>H33</td><td>H34</td><td>H35</td><td>H36</td><td>H37</td><td>H38</td><td>H39</td></tr>
</table>
<table>
<tr><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td></tr>
<tr><td>H41</td><td>H42</td><td>H43</td><td>H44</td><td>H45</td><td>H46</td></tr>
</table>
</td></tr>
<tr><td>Notes</td><td colspan="10">• All flags remain intact.
• The result is stored in the low-order bytes of R, R+1, R+2, and R+3, byte by byte, starting at the high-order data. The high-order bytes of R to R+3 are filled with ASCII ‘0’ (H30).</td></tr>
</table>

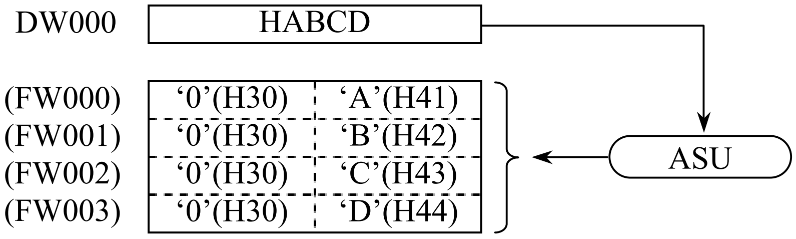
√: Applicable

na: Not applicable

[Word]



The contents of DW000 are converted from binary data to ASCII data only once when the input condition, R000, changes from OFF to ON. The result is stored in FL000 in unpacked mode.



(FW000)	‘0’(H30)	‘A’(H41)
(FW001)	‘0’(H30)	‘B’(H42)
(FW002)	‘0’(H30)	‘C’(H43)
(FW003)	‘0’(H30)	‘D’(H44)

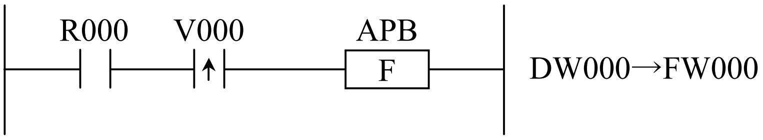
APB: ASCII→BINARY CONVERSION (PACKED MODE)

<table>
<tr><td>Function</td><td colspan="10">APB converts the contents of the source from an ASCII data (packed mode) to a binary data, and stores the result in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>na</td><td>na</td><td>na</td><td>-</td><td>↕</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>APB
−[F]−</td><td colspan="7">SW→RW</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td colspan="8"></td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source   W: Word
R: Address storing the result   L: Long Word</td></tr>
<tr><td>Contents of Process</td><td>Word</td><td colspan="9">$(S, S+1)W \xrightarrow[\text{Packed Mode}]{\text{ASCII→BIN}} (R)W$</td></tr>
<tr><td>Input Procedure</td><td colspan="10">([Shift] + [F]) [A] [P] [B] ␣Parameter, Parameter [Enter]</td></tr>
<tr><td>ASCII Correspondence Table</td><td colspan="10">
<table>
<tr><td>Binary</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
<tr><td>ASCII</td><td>H30</td><td>H31</td><td>H32</td><td>H33</td><td>H34</td><td>H35</td><td>H36</td><td>H37</td><td>H38</td><td>H39</td></tr>
</table>
<table>
<tr><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td></tr>
<tr><td>H41</td><td>H42</td><td>H43</td><td>H44</td><td>H45</td><td>H46</td></tr>
</table>
</td></tr>
<tr><td>Flag Setting</td><td colspan="10">E: When data other than H30 to H39 and H41 to H46 is detected: 1   Otherwise: 0
Others: Not affected</td></tr>
<tr><td>Notes</td><td colspan="10">No operation, if Error Flag (E) is turned ON.   The result is not affected.</td></tr>
</table>

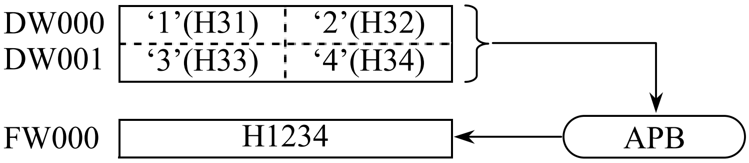
√: Applicable

na: Not applicable

[Word]



The contents of DW000 are converted from ASCII data (packed mode) to binary data only once when the input condition, R000, changes from OFF to ON.   The result is stored in Result.



AUB: ASCII→BINARY CONVERSION (UNPACKED MODE)

Function	AUB converts the contents of the source from an ANSII data (unpacked mode) to a binary data, and stores the result in Result.

Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	na	na	na	-	↕	-	-	-	-

Parameter				
	Word	Registers	AUB F	SW→RW
		Registers with immediate data		
	Long Word	Registers		
		Registers with immediate data		
	S: Address storing the source R: Address storing the result		W: Word L: Long Word	

Contents of Process	Word	$(S, S+1, S+2, S+3)W \xrightarrow[\text{Unpacked Mode}]{\text{ASCII→BIN}} (R)W$
Input Procedure	(Shift + F) A U B ␣Parameter, Parameter Enter	

ASCII Correspondence Table

Binary	0	1	2	3	4	5	6	7	8	9
ASCII	H30	H31	H32	H33	H34	H35	H36	H37	H38	H39

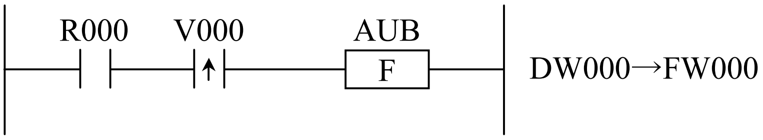
A	B	C	D	E	F
H41	H42	H43	H44	H45	H46

Flag Setting	E: When data other than H30 to H39 and H41 to H46 is detected: 1 Otherwise: 0 Others: Not affected
Notes	• No operation, if Error Flag (E) is turned ON. The result is not affected. • The high-order bytes of S to S+3 may contain optional values.

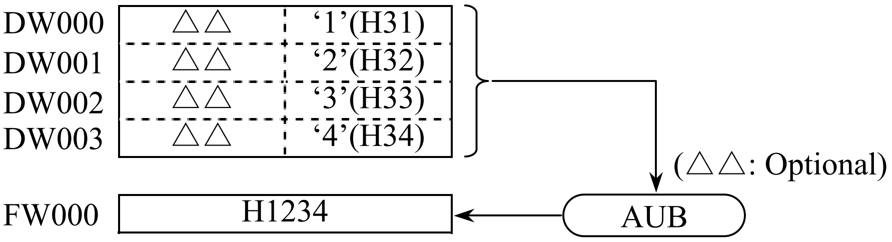
√: Applicable

na: Not applicable

[Word]



The contents of DW000 are converted from ASCII data (unpacked mode) to binary data only once when the input condition, R000, changes from OFF to ON. The result is stored in FW000.



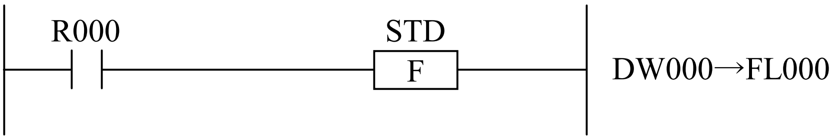
STD: SINGLE→DOUBLE CONVERSION

Function	STD extends the contents of the source with a sign extension from word data to long word data, and stores the result in Result.									
Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	na	na	na	-	-	-	-	-	-
Parameter	Word	Registers	STD ─[F]─	SW→RL						
		Registers with immediate data								
	Long Word	Registers								
		Registers with immediate data								
	S: Address storing the source R: Address storing the result			W: Word L: Long Word						
Contents of Process	Word	(S)W→(R)W								
Input Procedure	([Shift] + [F]) [S] [T] [D] ␣Parameter, Parameter [Enter]									
Notes	All flags remain intact.									

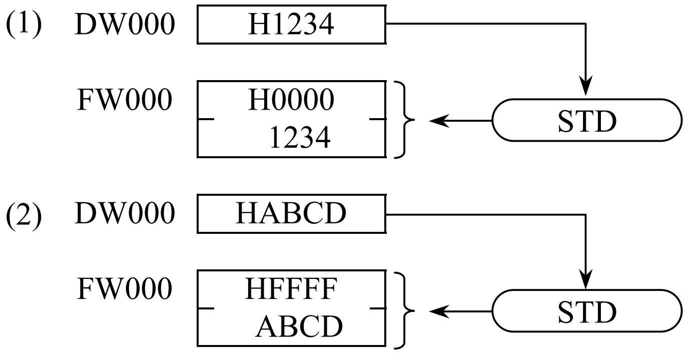
√: Applicable

na: Not applicable

[Word]



If the input condition, R000, is ON, the contents of DW000 are extended with sign extension from word data to long word data. The result is stored in FL000.



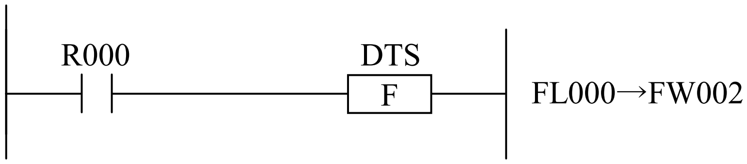
DTS: DOUBLE→SINGLE CONVERSION

<table>
<tr><td>Function</td><td colspan="10">DTS stores the contents of the source in the long word data format into Result in the word data format.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>na</td><td>na</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>↕</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td colspan="8"></td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>DTS
-[F]-</td><td colspan="7">SL→RW</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
R: Address storing the result L: Long Word</td></tr>
<tr><td>Contents of Process</td><td>Long Word</td><td colspan="9">(S)L→(R)W</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) D T S ⌴Parameter, Parameter Enter</td></tr>
<tr><td>Flag Setting</td><td colspan="10">V: 1, if (S)<-32768 or 32767<(S) 0, otherwise
Others: Not affected</td></tr>
<tr><td>Notes</td><td colspan="10">The following full-scale values are set in Result if an overflow occurs.<table><tr><td>When (S) L > 32767:</td><td>H7FFF</td></tr><tr><td>When (S) L < 32768:</td><td>H8000</td></tr></table></td></tr>
</table>

√: Applicable

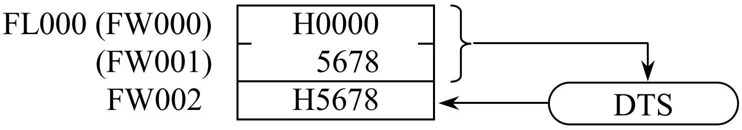
na: Not applicable

[Word]

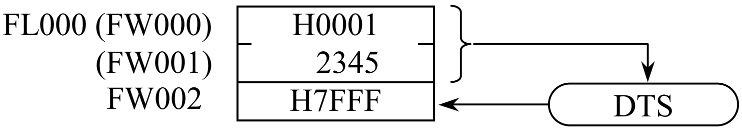


If the input condition, R000, is ON, the contents of DW000 in the long word format are stored in FW002 in the word data format.

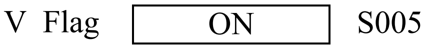
(1)



(2)



The V flag is turned ON.



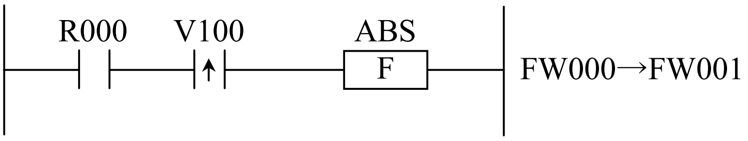
ABS: ABSOLUTE VALUE

<table>
<tr><td>Function</td><td colspan="10">ABS stores the absolute value of the contents of the source in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>na</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>↕</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td colspan="8"></td></tr>
<tr><td>Registers with immediate data</td><td>ABS
─[F]─</td><td colspan="7">SW→RW</td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td colspan="8"></td></tr>
<tr><td>Registers with immediate data</td><td>ABS
─[F]─</td><td colspan="7">SL→RL</td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
R: Address storing the result L: Long Word</td></tr>
<tr><td rowspan="2">Contents of Process</td><td colspan="2">Word</td><td colspan="8">| (S)W |→(R)W</td></tr>
<tr><td colspan="2">Long Word</td><td colspan="8">| (S)L |→(R)L</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) A B S ␣Parameter, Parameter Enter</td></tr>
<tr><td>Flag Setting</td><td colspan="10">V: (Word) 1, if (S)=-32768 0, otherwise
(Long Word) 1, if (S)=-2147483648 0, otherwise
Others: Not affected</td></tr>
<tr><td>Notes</td><td colspan="10">The following full-scale values are set in Result if an overflow occurs.<table><tr><th>Word</th><th>Long Word</th></tr><tr><td>H7FFF</td><td>H7FFFFFFF</td></tr></table></td></tr>
</table>

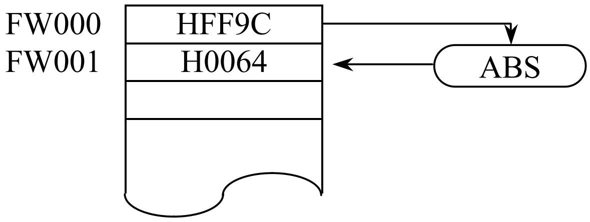
√: Applicable

na: Not applicable

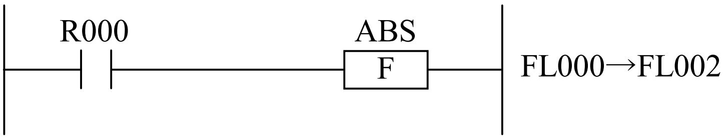
[Word]



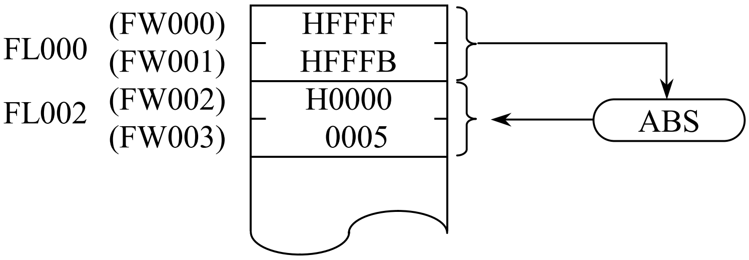
The absolute value of the contents of FW000 is stored in FW001 only once when the input condition, R000, changes from OFF to ON.



[Long Word]



If the input condition, R000, is ON, the absolute value of the contents of FL000 is stored in FL002.



NEG: SIGN CHANGE

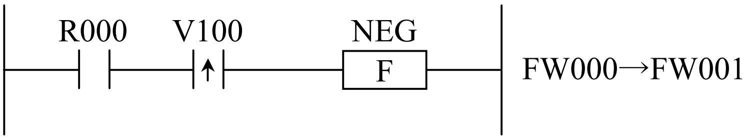
Function	NEG changes the sign of the contents of the source, and stores the result in Result.									
Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	na	√	na	-	-	-	-	-	↕
Parameter	Word	Registers	NEG F	SW→RW						
		Registers with immediate data								
	Long Word	Registers	NEG F	SL→RL						
		Registers with immediate data								
	S: Address storing the source R: Address storing the result			W: Word L: Long Word						
Contents of Process	Word	-(S)W→(R)W								
	Long Word	-(S)L→(R)L								
Input Procedure	(Shift + F) N E G ␣Parameter, Parameter Enter									
Flag Setting	V: (Word) 1, if (S)=-32768 0, otherwise (Long Word) 1, if (S)=-2147483648 0, otherwise Others: Not affected									
Notes	The following full-scale values are set in Result if an overflow occurs.									

Word	Long Word
H7FFF	H7FFFFFFF

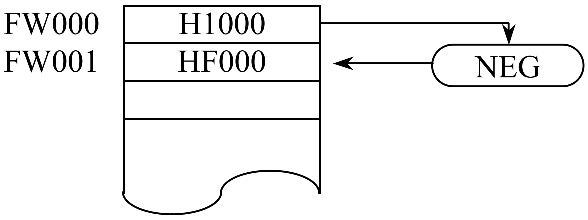
√: Applicable

na: Not applicable

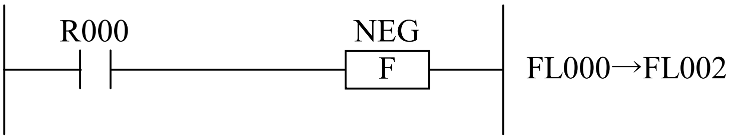
[Word]



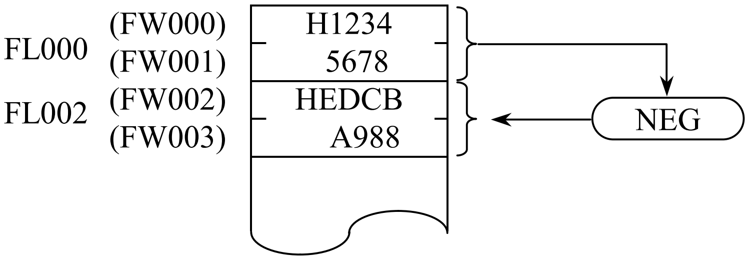
The sign of the contents of FW000 is changed only once when the input condition, R000, changes from OFF to ON, and the result is stored in FW001.



[Long Word]



If the input condition, R000, is ON, the sign of the contents of FL000 is changed and the result is stored in FL002.



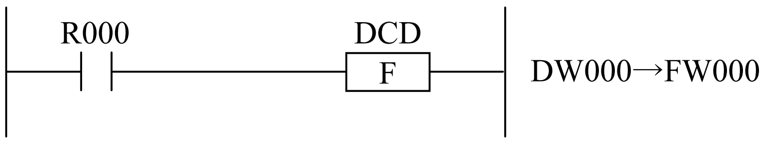
DCD: DECODE

<table>
<tr><td>Function</td><td colspan="10">DCD decodes the contents of the source, and stores the result in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>na</td><td>na</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td colspan="8"></td></tr>
<tr><td>Registers with immediate data</td><td>NEG
─[F]─</td><td colspan="7">SW→RW</td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td colspan="8"></td></tr>
<tr><td>Registers with immediate data</td><td>NEG
─[F]─</td><td colspan="7">SW→RL</td></tr>
<tr><td colspan="10">S: Address storing the source   W: Word
R: Address storing the result   L: Long Word</td></tr>
<tr><td rowspan="2">Contents of Process</td><td colspan="2">Word</td><td colspan="8">12 15   0   n   15(LSB)
SW [| n] → RW [0 to 0 | 1 | 0 to 0]</td></tr>
<tr><td colspan="2">Long Word</td><td colspan="8">11 15   0   n   31(LSB)
SW [| n] → RL [0 to 0 | 1 | 0 to 0]</td></tr>
<tr><td>Input Procedure</td><td colspan="10">([Shift] + [F]) [D] [C] [D] ␣Parameter, Parameter [Enter]</td></tr>
<tr><td>Notes</td><td colspan="10">• All flags remain intact.
• If the source data given in word, only the low-order 4 bits are significant, while low-order 5 bits are significant if given in long word.</td></tr>
</table>

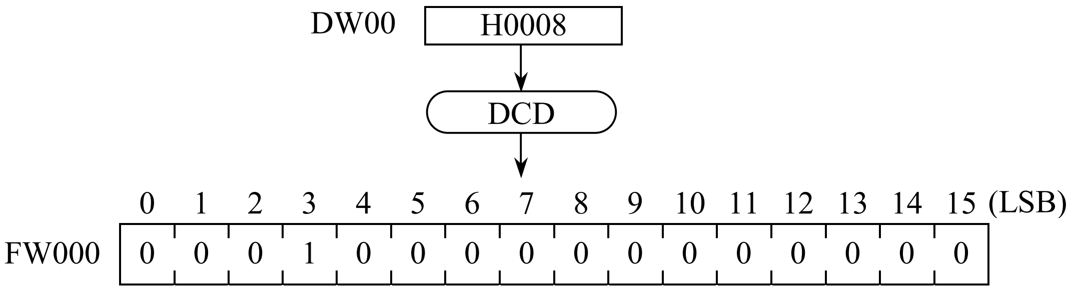
√: Applicable

na: Not applicable

[Word]

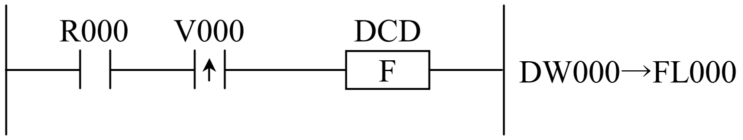


If the input condition, R000, is ON, the contents of DW000 are decoded and the result is stored in FW002.

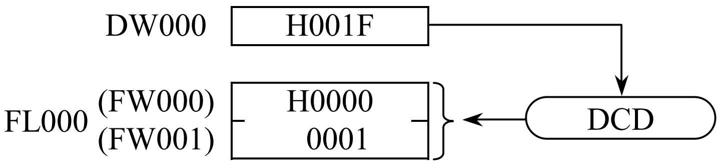


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 (LSB)
FW000	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

[Long Word]



The contents of DW000 are decoded only once when the input condition, R000, changes from OFF to ON, and the result is stored in FL000.



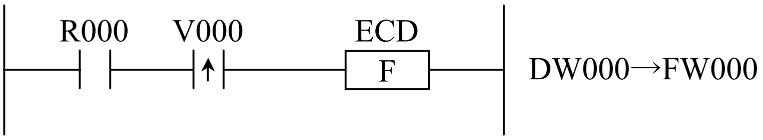
ECD: ENCODE

Function	ECD encodes the contents of the source, and stores the result in Result.									
Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	na	√	na	-	↕	-	-	-	-
Parameter	Word	Registers	ECD ─F─	SW→RW						
		Registers with immediate data								
	Long Word	Registers	ECD ─F─	SL→RW						
		Registers with immediate data								
	S: Address storing the source R: Address storing the result	W: Word L: Long Word								
Contents of Process	Word	SW	0 0 to 0	n 1	15(LSB) △ to △	→ RW	12	15 n		
	Long Word	SL=	0 0 to 0	n 1	31(LSB) △ to △	→ RW	11	15 n		
Input Procedure	(Shift + F) E C D ␣Parameter, Parameter Enter									
Flag Setting	E: When (S)=0: 1 Otherwise: 0 Others: Not affected									
Notes	• No operation if (S)=0. The result is not affected. • The bit being encoded has effect only on ‘1’ that is found first after MSB of the source.									

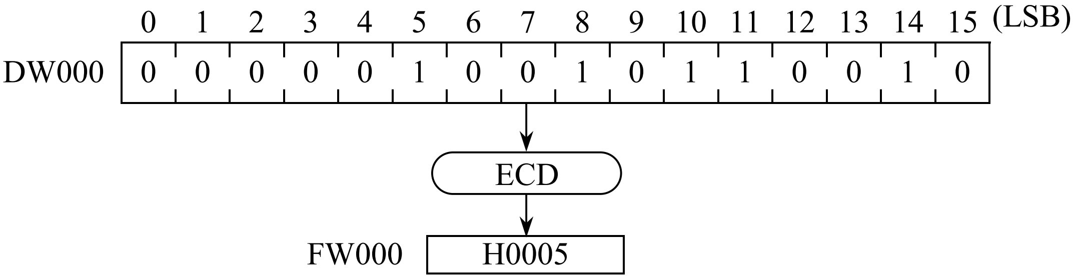
√: Applicable

na: Not applicable

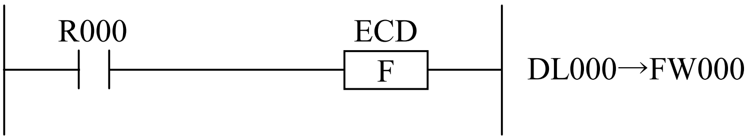
[Word]



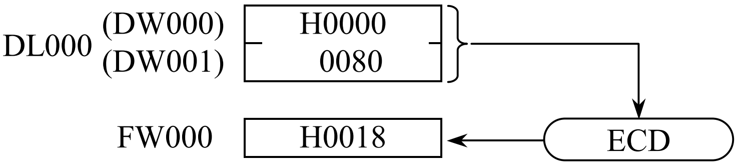
The contents of DW000 are encoded only once when the input condition, R000, changes from OFF to ON, and the result is stored in FW000.



[Long Word]



If the input condition, R000, is ON, the contents of DL000 are encoded and the result is stored in FW000.

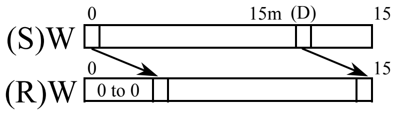
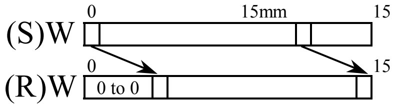
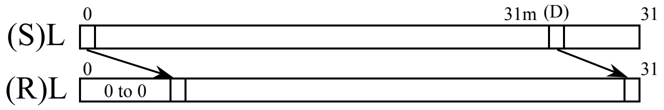
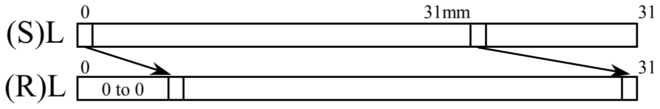


LSR: LOGICAL SHIFT RIGHT

Function	LSR shifts the contents of the source right for the number of bits specified by the contents of the destination or immediate data, and stores the result in Result.

Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	√	√	√	-	-	-	-	-	-

Parameter				
	Word	Registers	LSR F	SW: DW→RW
		Registers with immediate data	LSR F	SW: nW→RW
	Long Word	Registers	LSR F	SL: DW→RL
		Registers with immediate data	LSR F	SL: nW→RL
	S: Address storing the source D: Address storing the destination n: Immediate data R: Address storing the result			W: Word L: Long Word

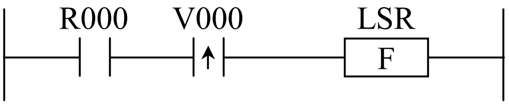
Contents of Process		
	Word	
	Word Constant	
	Long Word	
	Long Constant	

Input Procedure	(Shift + F) L S R ␣Parameter, Parameter, Parameter Enter
Notes	• All flags remain intact. • The bits being shifted are low-order 4 bits in word, or low-order 5 bits in long word.

√: Applicable

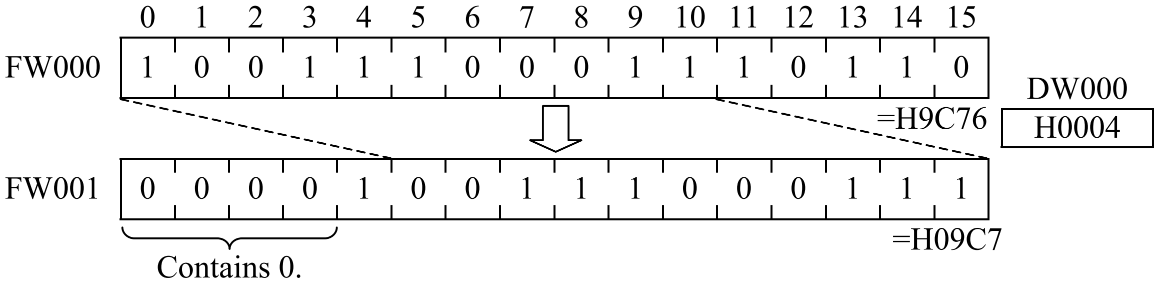
na: Not applicable

[Word]

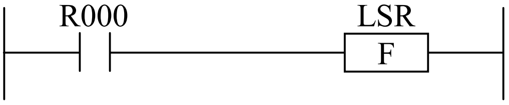


FW000: DW000→FW001

The contents of FW000 are shifted right for the number of bits specified by DW000 only once when the input condition, R000, changes from OFF to ON, and the result is stored in FW001.

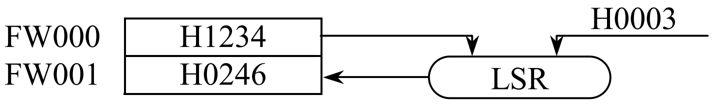


[Word Constant]

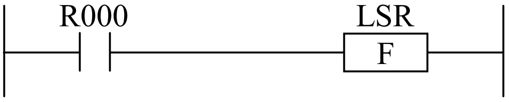


FW000: H0003→FW001

If the input condition, R000, is ON, the contents of FW000 are shifted 3 bits right and the result is stored in FW001.

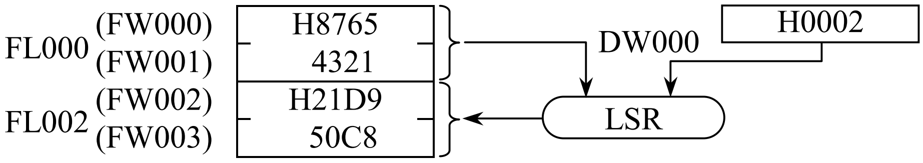


[Long Word]

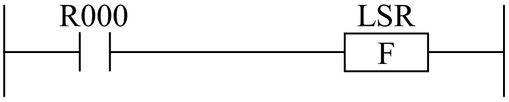


FL000: DW000→FL002

The contents of FL000 are shifted right for the number of bits specified by DW000 only once when the input condition, R000, changes from OFF to ON, and the result is stored in FL002.

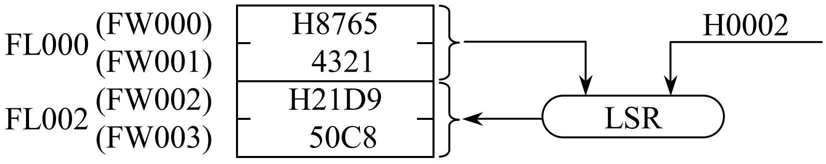


[Long Constant]



FL000: H0002→FL002

If the input condition, R000, is ON, the contents of FL000 are shifted 2 bits right and the result is stored in FL002.

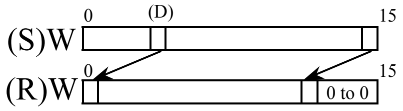
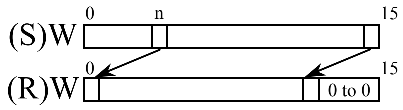
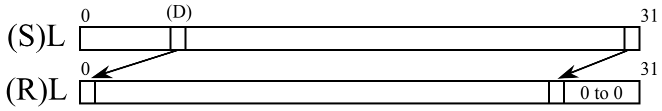
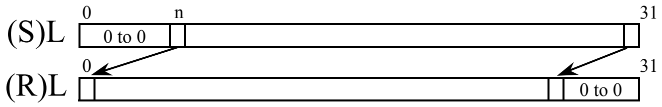


LSL: LOGICAL SHIFT LEFT

Function	LSL shifts the contents of the source left for the number of bits specified by the contents of the destination or immediate data, and stores the result in Result.

Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	√	√	√	-	-	-	-	-	-

Parameter				
	Word	Registers	LSL F	SW: DW→RW
		Registers with immediate data	LSL F	SW: nW→RW
	Long Word	Registers	LSL F	SL: DW→RL
		Registers with immediate data	LSL F	SL: nW→RL
	S: Address storing the source D: Address storing the destination n: Immediate data R: Address storing the result			W: Word L: Long Word

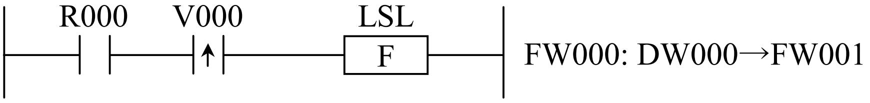
Contents of Process		
	Word	
	Word Constant	
	Long Word	
	Long Constant	

Input Procedure	(Shift + F) L S L ␣Parameter, Parameter, Parameter Enter
Notes	• All flags remain intact. • The bits being shifted are the low-order 4 bits in word, or low-order 5 bits in long word.

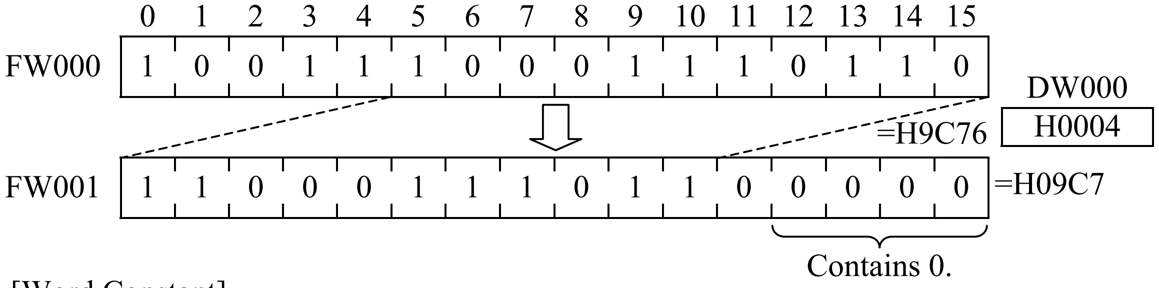
√: Applicable

na : Not applicable

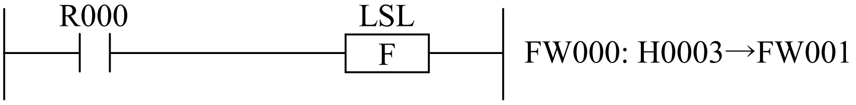
[Word]



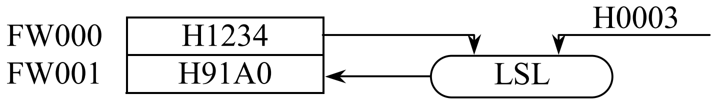
The contents of FW000 are shifted left for the number of bits specified by DW000 only once when the input condition, R000, changes from OFF to ON, and the result is stored in FW001.



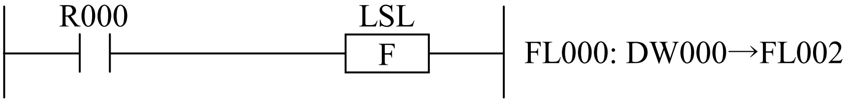
[Word Constant]



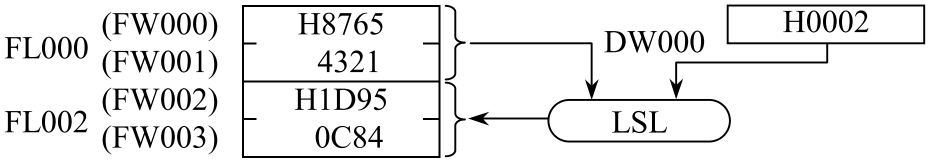
If the input condition, R000, is ON, the contents of FW000 are shifted 3 bits left and the result is stored in FW001.



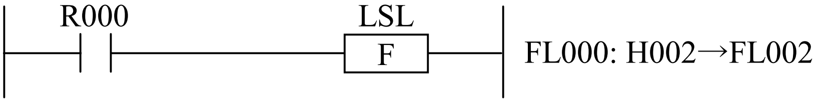
[Long Word]



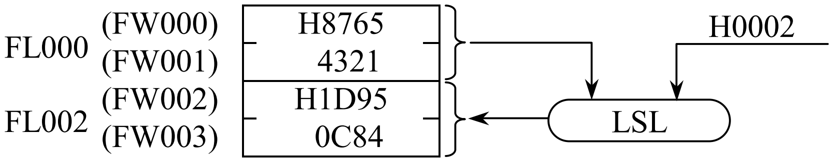
The contents of FL000 are shifted left for the number of bits specified by DW000 only once when the input condition, R000, is ON, and the result is stored in FL002.



[Long Constant]



If the input condition, R000, is ON, the contents of FL000 are shifted 2 bits left and the result is stored in FL002.

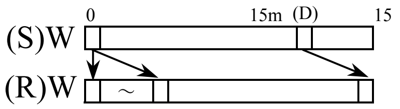


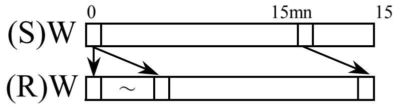
ASR: ARITHMETIC SHIFT RIGHT

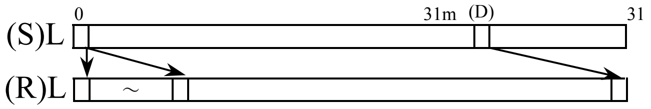
Function	ASR shifts the contents of the source right (with sign bit) for the number of bits specified by the contents of the destination or immediate data, and stores the result in Result.

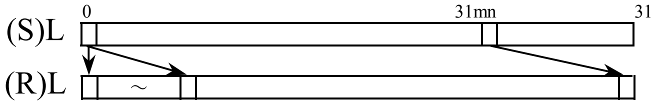
Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	√	√	√	-	-	-	-	-	-

Parameter	Word	Registers	ASR F	SW: DW→RW
		Registers with immediate data	ASR F	SW: nW→RW
	Long Word	Registers	ASR F	SL: DW→RL
		Registers with immediate data	ASR F	SL: nW→RL
	S: Address storing the source D: Address storing the destination n: Immediate data R: Address storing the result		W: Word L: Long Word	

<table>
<tr><td rowspan="4">Contents of Process</td><td>Word</td><td>


</td></tr>
<tr><td>Word Constant</td><td>


</td></tr>
<tr><td>Long Word</td><td>


</td></tr>
<tr><td>Long Constant</td><td>


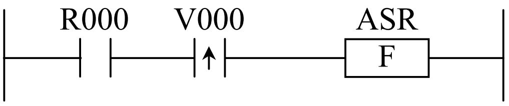
</td></tr>
</table>

Input Procedure	(Shift + F) A S R ␣Parameter, Parameter, Parameter Enter
Notes	• All flags remain intact. • The bits being shifted are the low-order 4 bits in word, or low-order 5 bits in long word.

√: Applicable

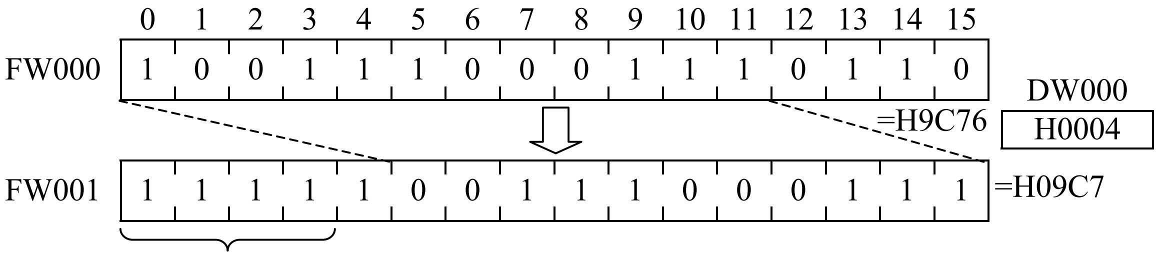
na: Not applicable

[Word]



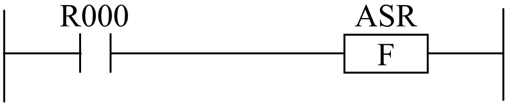
FW000: DW000→FW001

The contents of FW000 are shifted right for the number of bits specified by DW000 only once when the input condition, R000, changes from OFF to ON, and the result is stored in FW001.



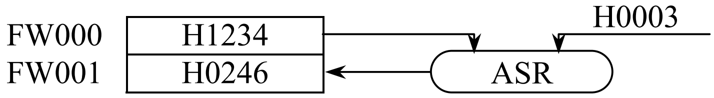
Contains the contents of Most Significant Bit.

[Word Constant]

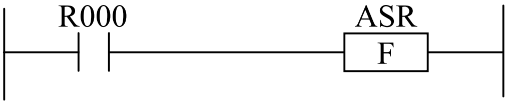


FW000: H0003→FW001

If the input condition, R000, is ON, the contents of FW000 are shifted 3 bits right and the result is stored in FW001.

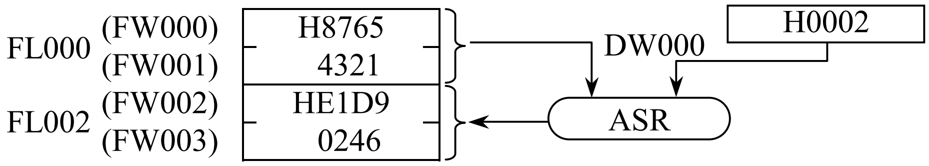


[Long Word]

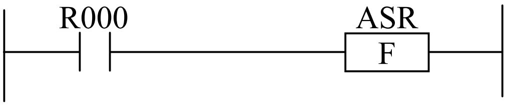


FL000: DW000→FL002

If the input condition, R000, is ON, the contents of FL000 are shifted right for the number of bits specified by DW000 and the result is stored in FL002.

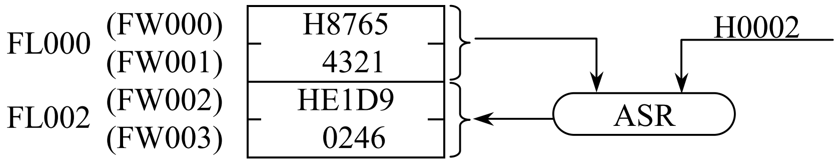


[Long Constant]



FL000: H002→FL002

If the input condition, R000, is ON, the contents of FL000 are shifted 2 bits right and the result is stored in FL002.



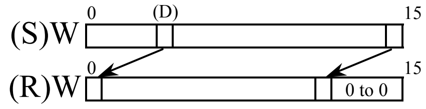
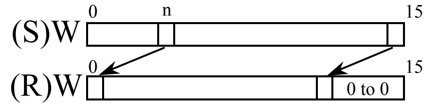
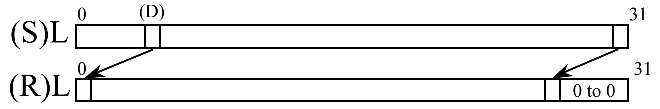
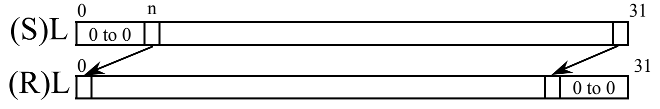
ASL: ARITHMETIC SHIFT LEFT

Function	ASL shifts the contents of the source left for the number of bits specified by the contents of the destination or immediate data, and stores the result in Result.

	Word		Long Word		Flag					
Parameter Format	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	√	√	√	-	-	-	-	-	↕

Parameter	Word	Registers	ASL -[F]-	SW: DW→RW
		Registers with immediate data	ASL -[F]-	SW: nW→RW
	Long Word	Registers	ASL -[F]-	SL: DW→RL
		Registers with immediate data	ASL -[F]-	SL: nW→RL

S: Address storing the source W: Word
D: Address storing the destination L: Long Word
n: Immediate data
R: Address storing the result

Contents of Process	Word	
	Word Constant	
	Long Word	
	Long Constant	

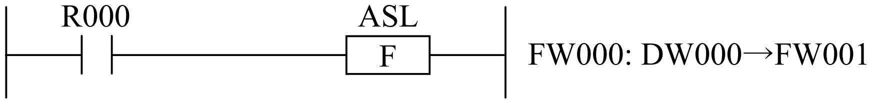
Input Procedure	(Shift + F) A S L ␣Parameter, Parameter, Parameter Enter
Flag Setting	V: Zero (0) is set if the sign bit changes for once during the shift operation. Others: Not affected
Notes	The following full-scale values are set if an overflow occurs.

	Word	Long Word
When (S) > 0:	H7FFF	H7FFFFFFF
When (S) < 0:	H8000	H80000000

√: Applicable

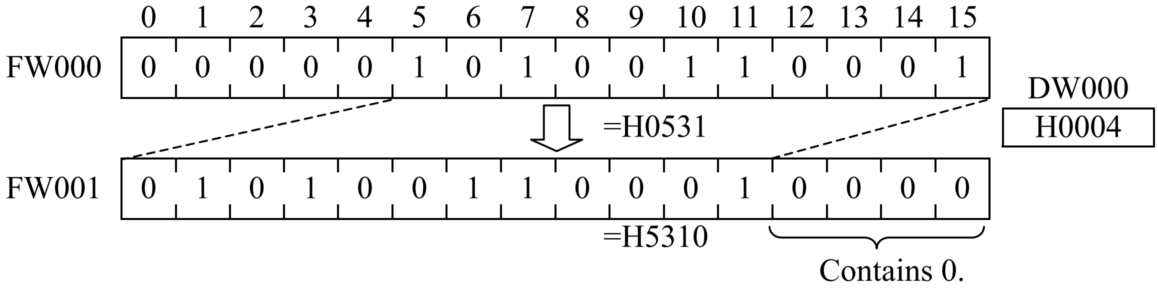
na: Not applicable

[Word]

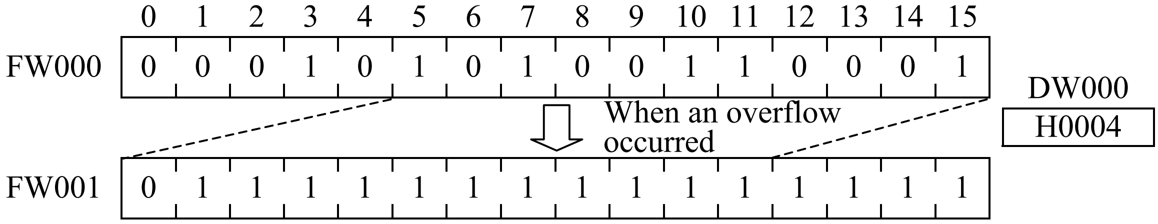


If the input condition, R000, is ON, the contents of FW000 are shifted left for the number of bits specified by DW000 and the result is stored in FW001.

(1)


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
FW000 0 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1
=H0531
DW000
H0004
FW001 0 1 0 1 0 0 1 1 0 0 0 1 0 0 0 0
=H5310
Contains 0.

(2)


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
FW000 0 0 0 1 0 1 0 1 0 0 1 1 0 0 0 1
When an overflow occurred
DW000
H0004
FW001 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

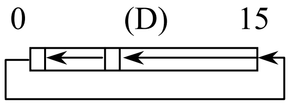
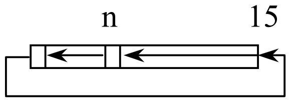
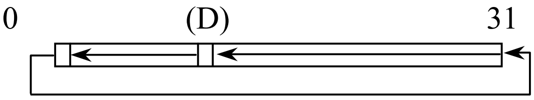
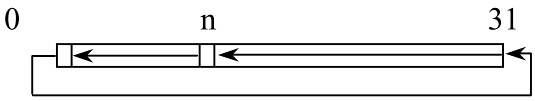
V flag is turned ON.

V Flag ON S005

ROL: ROTATE LEFT

Function	ROL rotates the contents of the source left for the number of bits specified by the contents of the destination or immediate data, and stores the result in Result.									
Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	√	√	√	-	-	-	-	-	-

Parameter				
	Word	Registers	ROL F	SW: DW→RW
		Registers with immediate data	ROL F	SW: nW→RW
	Long Word	Registers	ROL F	SL: DW→RL
		Registers with immediate data	ROL F	SL: nW→RL
	S: Address storing the source D: Address storing the destination n: Immediate data R: Address storing the result		W: Word L: Long Word	

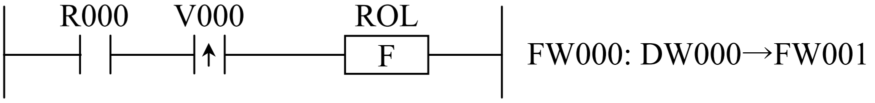
Contents of Process			
	Word		Left Rotation (S) → (R)
	Word Constant		Left Rotation (S) → (R)
	Long Word		Left Rotation (S)L → (R)L
	Long Constant		Left Rotation (S)L → (R)L

Input Procedure	(Shift + F) R O L ␣Parameter, Parameter, Parameter Enter
Notes	• All flags remain intact. • The bits being rotated are low-order 4 bits in word, or low-order 5 bits in long word.

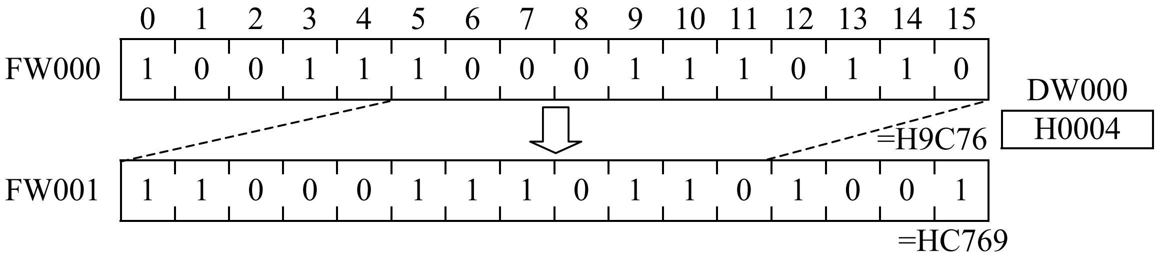
√: Applicable

na: Not applicable

[Word]

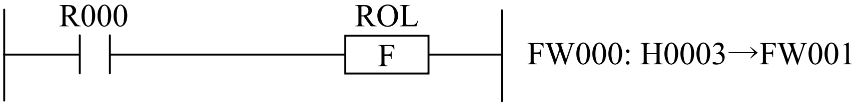


The contents of FW000 are rotated left for the number of bits specified by DW000 only once when the input condition, R000, changes from OFF to ON, and the result is stored in FW001.

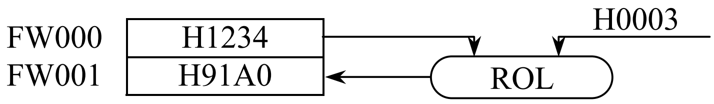


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FW000	1	0	0	1	1	1	0	0	0	1	1	1	0	1	1	0
FW001	1	1	0	0	0	1	1	1	0	1	1	0	1	0	0	1

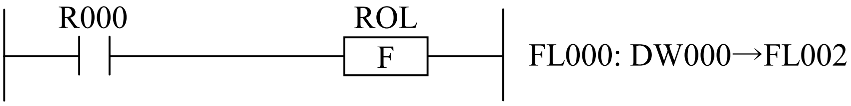
[Word Constant]



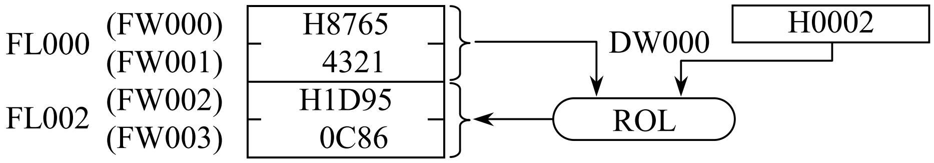
If the input condition, R000, is ON, the contents of FW000 are rotated 3 bits left and the result is stored in FW001.



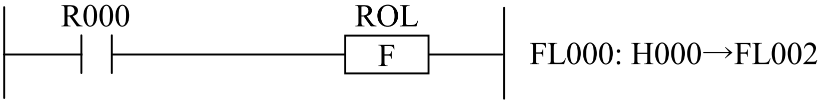
[Long Word]



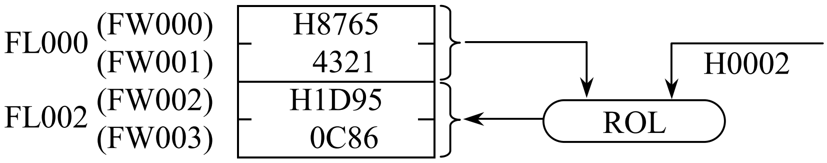
If the input condition, R000, is ON, the contents of FL000 are rotated left for the number of bits specified by DW000 and the result is stored in FL002.



[Long Constant]



If the input condition, R000, is ON, the contents of FL000 are rotated 2 bits left, and the result is stored in FL002.

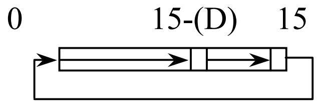
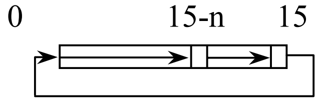
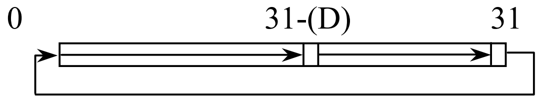
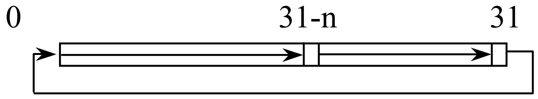


ROR: ROTATE RIGHT

Function	ROR rotates the contents of the source right for the number of bits specified by the contents of the destination or immediate data, and stores the result in Result.

Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	√	√	√	-	-	-	-	-	-

Parameter				
	Word	Registers	ROR F	SW: DW→RW
		Registers with immediate data	ROR F	SW: nW→RW
	Long Word	Registers	ROR F	SL: DW→RL
		Registers with immediate data	ROR F	SL: nW→RL
	S: Address storing the source D: Address storing the destination n: Immediate data R: Address storing the result			W: Word L: Long Word

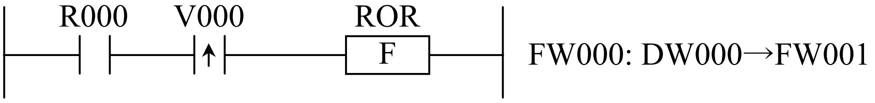
Contents of Process			
	Word		Right Rotation (S)W → (R)W
	Word Constant		Right Rotation (S)W → (R)W
	Long Word		Right Rotation (S)L → (R)L
	Long Constant		Right Rotation (S)L → (R)L

Input Procedure	(Shift + F) R O R ␣Parameter, Parameter, Parameter Enter
Notes	• All flags remain intact. • The bits being rotated are low-order 4 bits in word, or low-order 5 bits in long word.

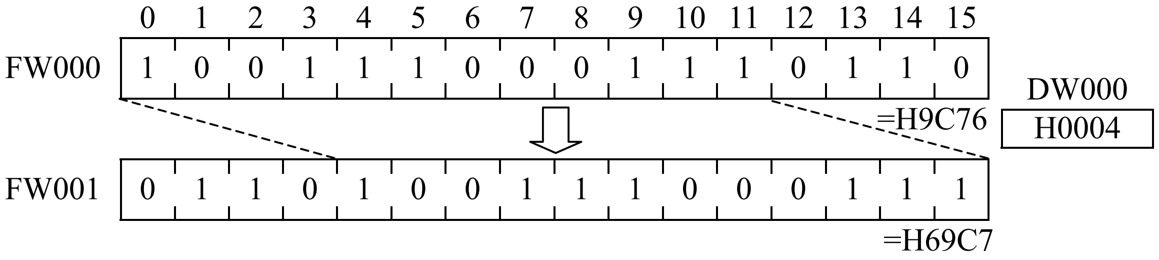
√: Applicable

na: Not applicable

[Word]

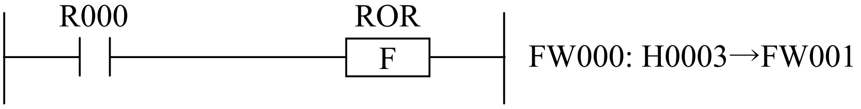


The contents of FW000 are rotated right for the number of bits specified by DW000 only once when the input condition, R000, changes from OFF to ON, and the result is stored in FW001.

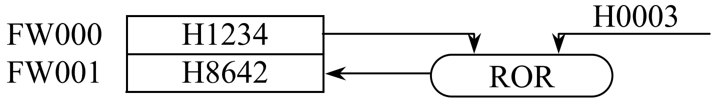


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
FW000	1	0	0	1	1	1	0	0	0	1	1	1	0	1	1	0	=H9C76
FW001	0	1	1	0	1	0	0	1	1	1	0	0	0	1	1	1	=H69C7

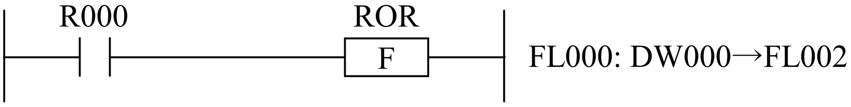
[Word Constant]



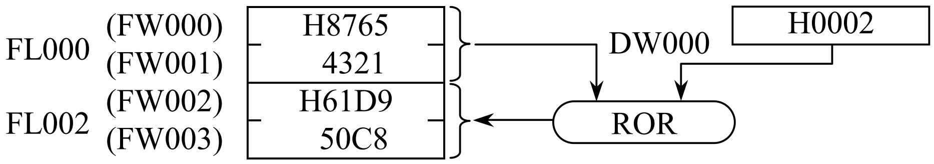
If the input condition, R000, is ON, the contents of FW000 are rotated 3 bits right if the input condition, R000, is ON, and the result is stored in FW001.



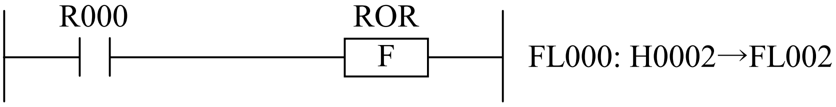
[Long Word]



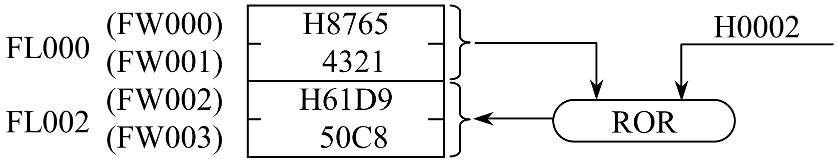
The contents of FL000 are rotated right for the number of bits specified by DW000 only once when the input condition, R000, is ON, and the result is stored in FL002.



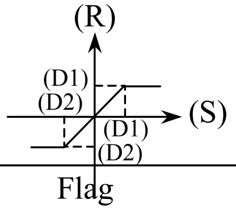
[Long Constant]



If the input condition, R000, is ON, the contents of FL000 are rotated 2 bits right if the input condition, R000, is ON, and the result is stored in FL002.



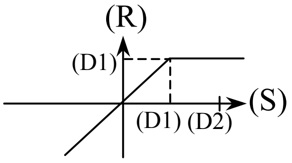
LIM: LIMITER

Function	LIM compares the contents of the source with the boundary value (the contents of destinations, D1 and D2), and stores the result in Result. (See the figure at right.) 

Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	√	√	na	-	↕	-	-	-	-

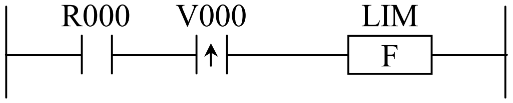
Parameter	Word	Registers	LIM ─[F]─	SW: D1W: D2W→RW
		Registers with immediate data	LIM ─[F]─	SW: n1W: n2W→RW
	Long Word	Registers	LIM ─[F]─	SL: D1L: D2L→RL
		Registers with immediate data		
	S: Address storing the source D1 and D2: Address storing the destination n1 and n2: Immediate data R: Address storing the result			W: Word L: Long Word

Contents of Process	Word	• When (D1)<(S): (D1)W→(R)W • When (D2)≤(S)≤(D1): (S)W→(R)W • When (S)<(D2): (D2)W→(R)W
	Word Constant	• When n1<(S): n1W→(R)W • When n2≤(S)≤n1: (S)W→(R)W • When (S)<n2: n2W→(R)W
	Long Word	• When (D1)L<(S)L: (D1)L→(R)L • When (D2)L≤(S)L≤(D1)L: (S)L→(R)L • When (S)L<(D2)L: (D2)L→(R)L

Input Procedure	(Shift + F) L I M ⌴Parameter, Parameter, Parameter, Parameter Enter
Flag Setting	E: 1, if (D1)<(D2) or n1<n2 0, otherwise Others: Not affected
Notes	If the error flag is ON, comparison between the contents of the source and (D2) or n2 is not performed. (See the figure at right.) 

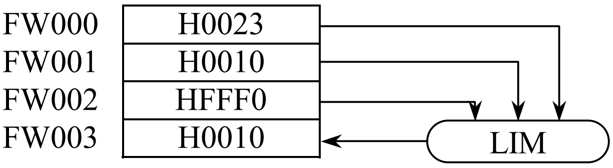
√: Applicable
na: Not applicable

[Word]

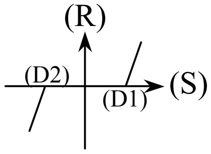


FW000: FW001: FW002→FW003

The contents of FW000 are compared with the contents of FW001 and FW002 only once when the input condition, R000, changes from OFF to ON. The result is stored in FW003.

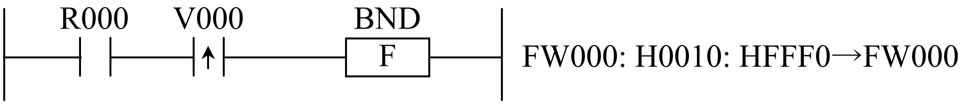


BND: DEAD BAND

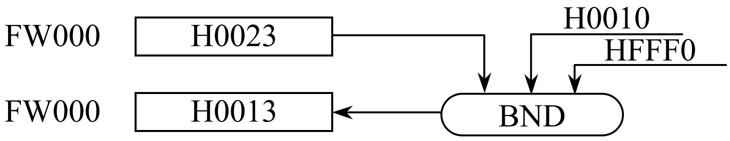
<table>
<tr><td>Function</td><td colspan="10">BND compares the contents of the source with the boundary value (the contents of destinations, D1 and D2, or immediate data, n1 and n2), and stores the extent of the boundary in Result as Data 0 (dead area). (See the figure at right.)

 </td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>√</td><td>√</td><td>na</td><td>-</td><td>↕</td><td>-</td><td>-</td><td>-</td><td>↕</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>BND
─[F]─</td><td colspan="7">SW: D1W: D2W→RW</td></tr>
<tr><td>Registers with immediate data</td><td>BND
─[F]─</td><td colspan="7">SW: n1W: n2W→RW</td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>BND
─[F]─</td><td colspan="7">SL: D1L: D2L→RL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source   W: Word
D1 and D2: Address storing the destination   L: Long Word
n1 and n2: Immediate data
R: Address storing the result</td></tr>
<tr><td rowspan="3">Contents of Process</td><td colspan="2">Word</td><td colspan="8">• When (D1)<(S): (S)W-(D1)W→(R)W
• When (D2)≤(S)≤(D1): 0→(R)W
• When (S)<(D2): (S)W-(D2)W→(R)W</td></tr>
<tr><td colspan="2">Word Constant</td><td colspan="8">• When n1<(S): (S)-n1W →(R)W
• When n2≤(S)≤n1: 0→(R)W
• When (S)<n2: (S)W-n2W→(R)W</td></tr>
<tr><td colspan="2">Long Word</td><td colspan="8">• When (D1)L<(S)L: (S)L-(D1)L→(R)L
• When (D2)L≤(S)L≤(D1)L: 0→(R)L
• When (S)L<(D2)L: (S)L-(D2)L→(R)L</td></tr>
<tr><td>Input Procedure</td><td colspan="10">([Shift] + [F]) [B] [N] [D]
⌴ Parameter, Parameter, Parameter, Parameter [Enter]</td></tr>
<tr><td>Flag Setting</td><td colspan="10">E: 1, if (D1)<(D2) or n1<n2   0, otherwise
V: (Word)   1, if (R)<-32768 or 32767<R   0, otherwise
(Long Word) 1, if (R)<-2147483648 or 2147483647<(R)   0, otherwise
Others: Not affected</td></tr>
<tr><td>Notes</td><td colspan="10">• If (D1)<(D2) or n1 <n2, the error flag (E) turns ON, the overflow flag (V) turns OFF, and no process is made. (The result is not affected.)
• The following full-scale values are set if an overflow occurred:
<table><tr><td></td><td>Word</td><td>Long Word</td></tr><tr><td>Positive Overflow</td><td>H7FFF</td><td>H7FFFFFFF</td></tr><tr><td>Negative Overflow</td><td>H8000</td><td>H80000000</td></tr></table></td></tr>
</table>

√: Applicable
na: Not applicable

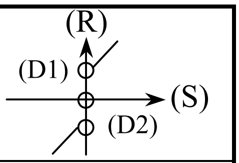
[Word]



The contents of FW000 are compared with immediate data, H001 and HFFF0, only once when the input condition, R000, changes from OFF to ON. The result is stored in FW000.



ZON: DEAD ZONE

Function	ZON adds a bias (the contents of destinations, D1 and D2, or immediate data, n1 and n2) according to the sign condition, and stores the result in Result. (See the figure at right.) 

Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	√	√	√	-	↕	-	-	-	↕

Parameter				
	Word	Registers	ZON F	SW: D1W: D2W→RW
		Registers with immediate data	ZON F	SW: n1W: n2W→RW
	Long Word	Registers	ZON F	SL: D1L: D2L→RL
		Registers with immediate data		

S: Address storing the source W: Word
D: Address storing the destination L: Long Word
n: Immediate data
R: Address storing the result

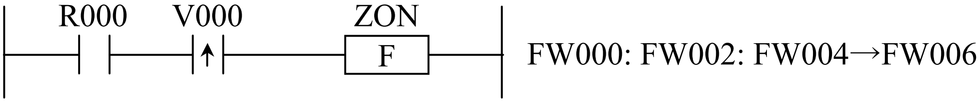
Contents of Process		
	Word	• When (S) >0: (S)W+(D1)W→(R)W • When (S) =0: 0→(R)W • When (S) <0: (S)W+(D2)W→(R)W
	Word Constant	• When (S) >0: (S)W+n1W→(R)W • When (S) =0: 0W→(R)W • When (S) <0: (S)W+n2W→(R)W
	Long Word	• When (S)L>0: (S)L+(D1)L→(R)L • When (S)L=0: 0→(R)L • When (S)L<0: (S)L+(D2)L→(R)L

Input Procedure	(Shift + F) Z O N ␣ Parameter, Parameter, Parameter, Parameter Enter
Flag Setting	E: 1, if (D1)<(D2) or n1<n2 0, otherwise V: (Word) 1, if (R)<-32768 or 32767<R 0, otherwise (Long Word) 1, if (R)<-2147483648 or 2147483647<(R) 0, otherwise Others: Not affected
Notes	• If (D1)<(D2) or n1 <n2, the error flag (E) turns ON, the overflow flag (V) turns OFF, and no operation is made. (The result is not affected.) • The following full-scale values are set if an overflow occurred:

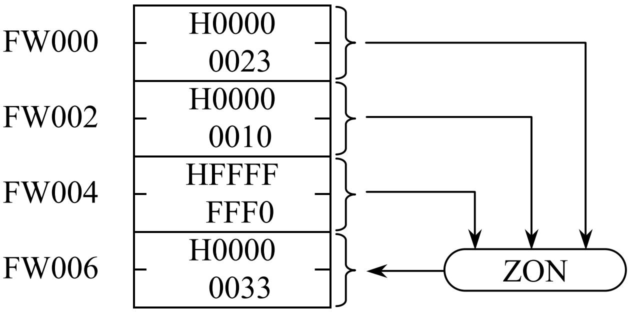
	Word	Long Word
Positive Overflow	H7FFF	H7FFFFFFF
Negative Overflow	H8000	H80000000

√: Applicable
na: Not applicable

[Word]



The contents of FW002 or FW004 are added according to the sign condition of the contents of FW000 only once when the input condition, R000, changes from OFF to ON. The result is stored in FW006.



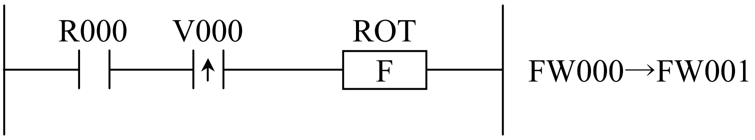
ROT: ROOT

<table>
<tr><td>Function</td><td colspan="10">ROT stores the root of the contents of the source (integer part, only) in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>na</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>ROT
F</td><td colspan="7">SW→DW</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>ROT
F</td><td colspan="7">SL→DL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
D: Address storing the destination L: Long Word</td></tr>
<tr><td rowspan="2">Contents of Process</td><td colspan="2">Word</td><td colspan="8">When (S)W≥0: $\sqrt{(S)W}$→(R)W
When (S)W<0: 0W→(R)W</td></tr>
<tr><td colspan="2">Long Word</td><td colspan="8">When (S)L≥0: $\sqrt{(S)L}$→(R)L
When (S)L<0: 0→(R)L</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) R O T ␣ Parameter, Parameter Enter</td></tr>
<tr><td>Notes</td><td colspan="10">• All flags remain intact.
• If the contents of the source is negative, 0 is set in Result.</td></tr>
</table>

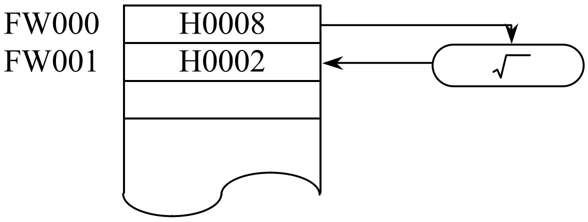
√: Applicable

na: Not applicable

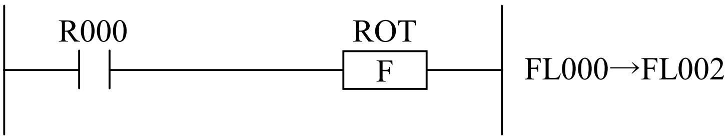
[Word]



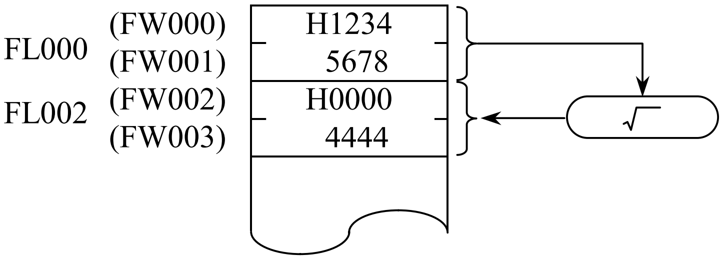
The root of the contents of FW000 is obtained only once when the input condition, R000, changes from OFF to ON, and the result is stored in FW001.



[Long Word]



If the input condition, R000, is ON, the root of the contents of FL000 is obtained and the result is stored in FL002.



MAX: MAXIMUM

Function	MAX compares the sizes of the contents of the source and the destination or immediate data, and stores larger value in Result.

Parameter Format	Word		Long Word		Flag					
	Registers	Registers with immediate data	Registers	Registers with immediate data	X	E	P	N	Z	V
	√	√	√	na	-	-	-	-	-	-

Parameter	Word	Registers	MAX ─[F]─	SW: DW→RW
		Registers with immediate data	MAX ─[F]─	SW: nW→RW
	Long Word	Registers	MAX ─[F]─	SL: DL→RL
		Registers with immediate data		
	S: Address storing the source D: Address storing the destination n: Immediate data R: Address storing the result			W: Word L: Long Word

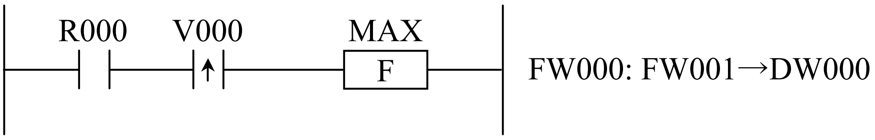
Contents of Process	Word	When (S)W≥(D)W: (S)W→(R)W When (S)W<(D)W: (D)W→(R)W
	Word Constant	When (S)W≥nW: (S)W→(R)W When (S)W<nW: nW→(R)W
	Long Word	When (S)L≥ (D)L: (S)L→(R)L When (S)L<(D)L: (D)L→(R)L

Input Procedure	([Shift] + [F]) [M] [A] [X] ⌴Parameter, Parameter, Parameter [Enter]
Notes	All flags remain intact.

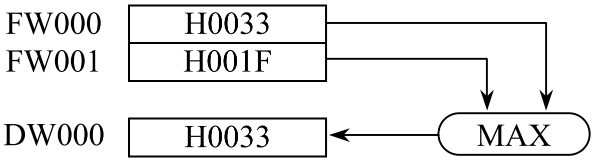
√: Applicable

na: Not applicable

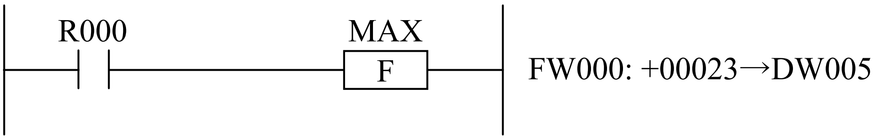
[Word]



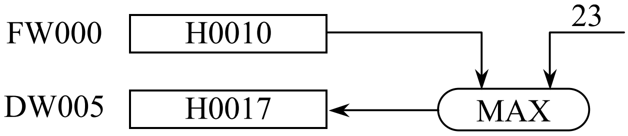
The contents of FW000 and FW001 are compared only once when the input condition, R000, changes from OFF to ON, and larger value is stored in DW001.



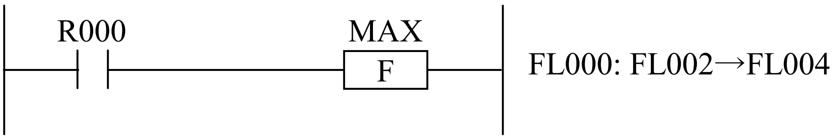
[Word Constant]



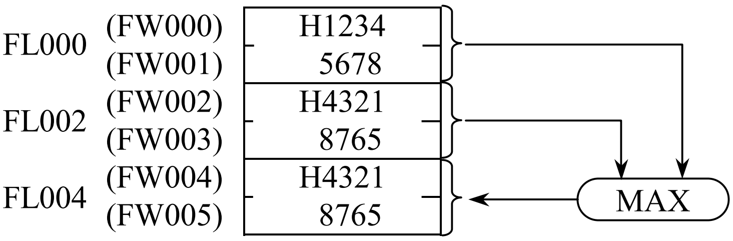
If the input condition, R000, is ON, the contents of FW000 and immediate data, 23, are compared and larger value is stored in DW005.



[Long Word]



If the input condition, R000, is ON, the contents of FL000 and FL002 are compared and larger value is stored in FL004.



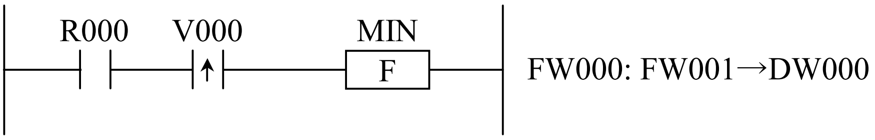
MIN: MINIMUM

<table>
<tr><td>Function</td><td colspan="10">MIN compares the sizes of the contents of the source and the destination or immediate data, and stores smaller value in Result.</td></tr>
<tr><td rowspan="3">Parameter Format</td><td colspan="2">Word</td><td colspan="2">Long Word</td><td colspan="6">Flag</td></tr>
<tr><td>Registers</td><td>Registers with immediate data</td><td>Registers</td><td>Registers with immediate data</td><td>X</td><td>E</td><td>P</td><td>N</td><td>Z</td><td>V</td></tr>
<tr><td>√</td><td>√</td><td>√</td><td>na</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="2">Word</td><td>Registers</td><td>MIN
F</td><td colspan="7">SW: DW→RW</td></tr>
<tr><td>Registers with immediate data</td><td>MIN
F</td><td colspan="7">SW: nW→RW</td></tr>
<tr><td rowspan="2">Long Word</td><td>Registers</td><td>MIN
F</td><td colspan="7">SL: DL→RL</td></tr>
<tr><td>Registers with immediate data</td><td colspan="8"></td></tr>
<tr><td colspan="10">S: Address storing the source W: Word
D: Address storing the destination L: Long Word
n: Immediate data
R: Address storing the result</td></tr>
<tr><td rowspan="3">Contents of Process</td><td>Word</td><td colspan="9">When (S)W≤(D)W: (S)W→(R)W
When (S)W>(D)W: (D)W→(R)W</td></tr>
<tr><td>Word Constant</td><td colspan="9">When (S)W≤nW: (S)W→(R)W
When (S)W>nW: nW→(R)W</td></tr>
<tr><td>Long Word</td><td colspan="9">When (S)L≤ (D)L: (S)L→(R)L
When (S)L>(D)L: (D)L→(R)L</td></tr>
<tr><td>Input Procedure</td><td colspan="10">(Shift + F) M I N ⌴ Parameter, Parameter, Parameter Enter</td></tr>
<tr><td>Notes</td><td colspan="10">All flags remain intact.</td></tr>
</table>

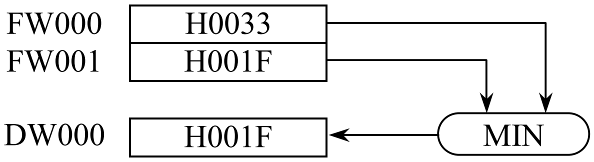
√: Applicable

na: Not applicable

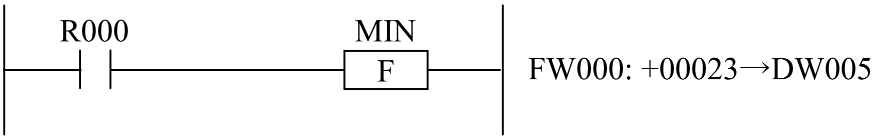
[Word]



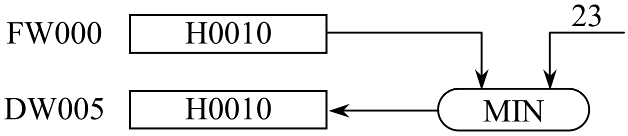
The contents of FW000 and FW001 are compared only once when the input condition, R000, changes from OFF to ON, and smaller value is stored in DW000.



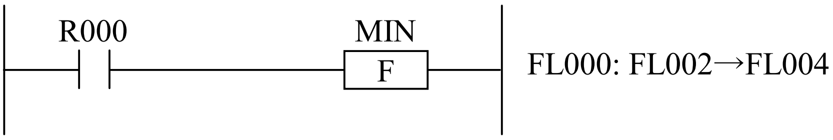
[Word Constant]



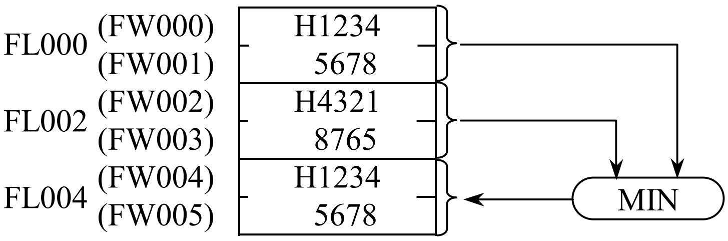
If the input condition, R000, is ON, the contents of FW000 and immediate data, 23, are compared and smaller value is stored in DW005.



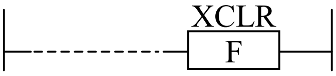
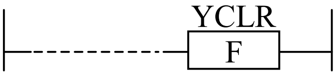
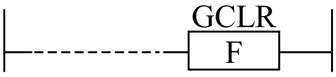
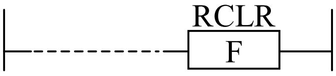
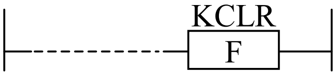
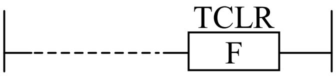
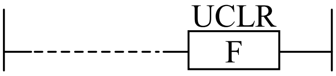
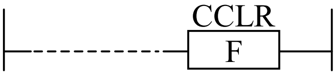
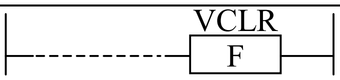
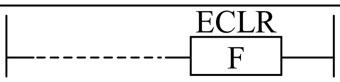
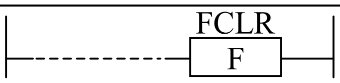
[Long Word]

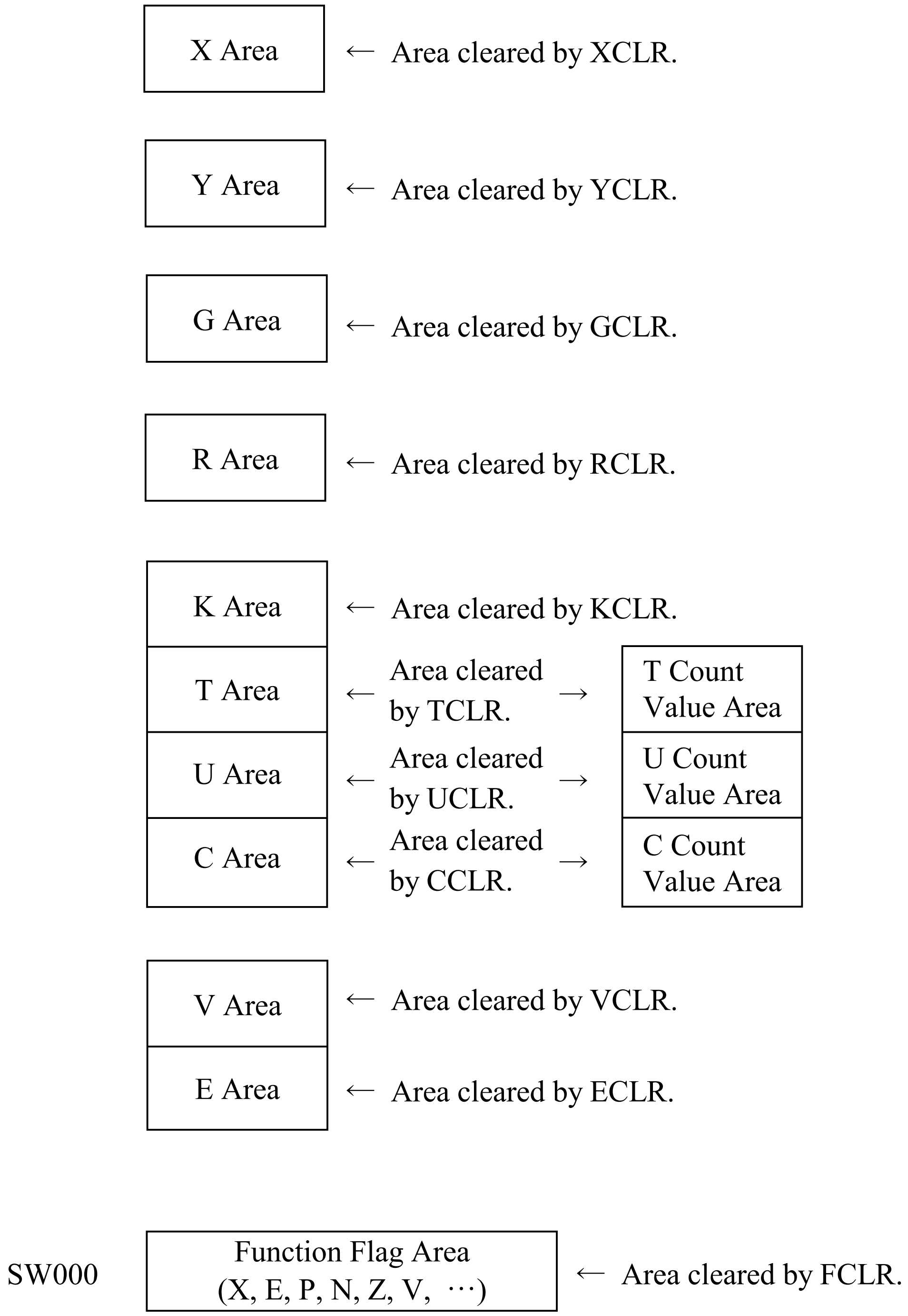


If the input condition, R000, is ON, the contents of FL000 and FL002 are compared and smaller value is stored in FL004.



CLR: CLEAR

Function	CLR clears a specified I/O area. TCLR, UCLR, and CCLR also clear the respective count value areas.							
Parameter Format			Flag					
			X	E	P	N	Z	V
Symbol	XCLR		-	-	-	-	-	-
	YCLR		-	-	-	-	-	-
	GCLR		-	-	-	-	-	-
	RCLR		-	-	-	-	-	-
	KCLR		-	-	-	-	-	-
	TCLR		-	-	-	-	-	-
	UCLR		-	-	-	-	-	-
	CCLR		-	-	-	-	-	-
	VCLR		-	-	-	-	-	-
	ECLR		-	-	-	-	-	-
	FCLR		0	0	0	0	0	0
Symbol Input Procedure	(Shift + F) □ C L R Enter Enter the code (X, Y, G, R, …) of an area to be cleared. No parameters are required.							

Cleaer Area	

X Area
← Area cleared by XCLR.
Y Area
← Area cleared by YCLR.
G Area
← Area cleared by GCLR.
R Area
← Area cleared by RCLR.
K Area
← Area cleared by KCLR.
T Area
← Area cleared by TCLR. →
T Count Value Area
U Area
← Area cleared by UCLR. →
U Count Value Area
C Area
← Area cleared by CCLR. →
C Count Value Area
V Area
← Area cleared by VCLR.
E Area
← Area cleared by ECLR.
SW000
Function Flag Area (X, E, P, N, Z, V, ⋯)
← Area cleared by FCLR.

THIS PAGE INTENTIONALLY LEFT BLANK.

3 PROCESSING TIME

3.1 Processing Time for 2α

The processing time of a ladder circuit program is a total of the execution time of the ladder program and the OS processing time including the arithmetic instruction, process register, nesting coil, and so on.

(1) Processing Time of Ladder Circuit Instruction

Symbol	Processing Time μs/Instruction		
	2α	2αE	2αH (f)
┤├, ┤/├, │, ┤, ┐	0.33		0.075
┤↑├, ┤↓├, -○-, -Ⓢ-, -Ⓡ-	0.66		0.15
END Instruction (Automatically set at programming)	300	180	

(2) Processing Time of Arithmetic Function
Figure 3-1 shows the outline of the arithmetic processing by OS.

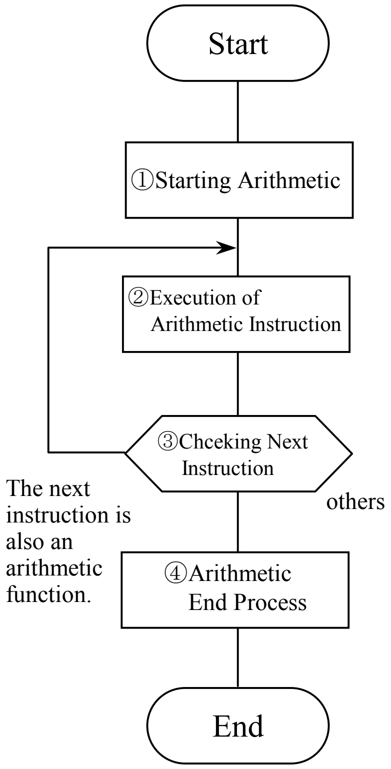

Start
①Starting Arithmetic
②Execution of Arithmetic Instruction
③Chceking Next Instruction
The next instruction is also an arithmetic function.
others
④Arithmetic End Process
End

Figure 3-1 Outline of the Arithmetic Processing by OS

The time required for ② is given in the Arithmetic Function List. (See section 2.4.)
The OS processing time for ①, ③, and ④ is shown in the following table:

No.	Processing Time [ms]	
	2α	2αE, 2αH (f)
①	0.12	0.072
③	When the next is an arithmetic instruction: 0.04 Otherwise: 0.01	When the next is an arithmetic instruction: 0.024 Otherwise: 0.006
④	0.12	0.072

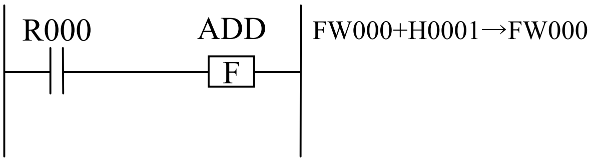
[Brief Calculation]

- Single arithmetic function instruction:
  0.3 ms/instruction (2α)
  0.18 ms/instruction (2αE, 2αH (f))
- Sequence of more than 1 arithmetic function instructions:
  0.15 ms/instruction (2α)
  0.09 ms/instruction (2αE, 2αH(f))

The approximate process time required for an arithmetic function is obtained by multiplying the unit time in the above by the number of the arithmetic function instructions being used.

The method of calculation for the following circuits is shown below using 2α as an example.

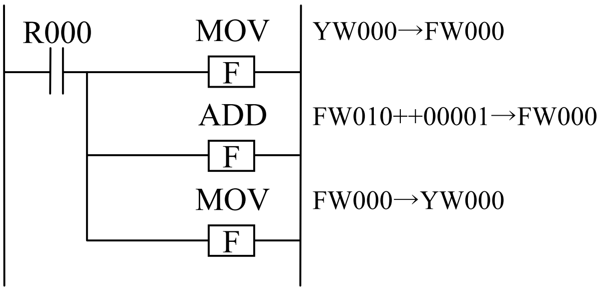
(a)



Processing Time
=①+②+③+④

①	=0.12
②	=0.06
③	=0.01
④	=0.12
Total	0.31 [ms]

(b)

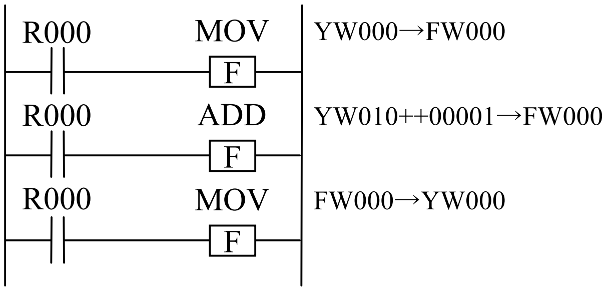


Processing Time
=①+3×③+④

①		=0.12
3× ③	=2×0.04+1×0.01	=0.09
④		=0.12
2×②MOV	=2×0.04	=0.08
②ADD#		=0.06
Total		=0.47 [ms]

A circuit, (b), in the above may be formed as (c), though the processing time becomes longer.

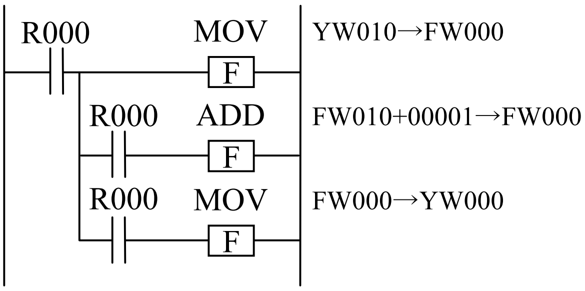
(c)



Processing Time
=[①+②MOV×③+④]
+[①+②ADD#+③+④]
+[①+②MOV+③+④]
=3×①+3×③+3×④
+2×②MOV+②ADD#

3×①=3×0.12	=0.36
3×③=3×0.01	=0.03
3×④=3×0.12	=0.36
2×②MOV=2×0.04	=0.08
②ADD#	=0.06
Total	=0.89 [ms]

The processing time of the following circuit can be obtained by the same calculation as in (c). If at least one contact, etc. appears after the arithmetic function, use ‘Others’ for ③.



- If the arithmetic function is not ON, the processing time is a total 0.06μs/instruction - the same as -○- of the ladder circuit instruction and the parameter execution time (0.33μs/1 parameter).

(3) Processing Time of Special Coil

The process register (P) and nesting coil (N), if ON, are processed by OS like the arithmetic function. The processing time is shown below:

Item	Processing Time [ms]	
	2α	2αE, 2αH (f)
P -○-	0.5	0.3
N -○-	0.5	0.3

- Values given in the above may become larger if there are many CPU queues.
- When each coil is not ON, it takes 0.66μs/instruction.

(4) Processing Time of Up-Down Counter

The up-down counter is processed in the same way as the arithmetic instruction. In this case, however, set the following value in ② of the arithmetic instruction calculation expression before computation.

Counter Processing Time ≒ 70μs/instrution (2α)
≒ 42μs/insttuction (2αE, 2αH (f))

(5) Processing Time of Timer (T) and One-Shot (U)

Processing of the timer and one-shot is made for the number of edition points at 100 ms intervals without regard to the existence of a program.

This 100 ms timing is generated by the interrupts caused at 10 ms intervals within the OS, as shown in Figure 3.2.

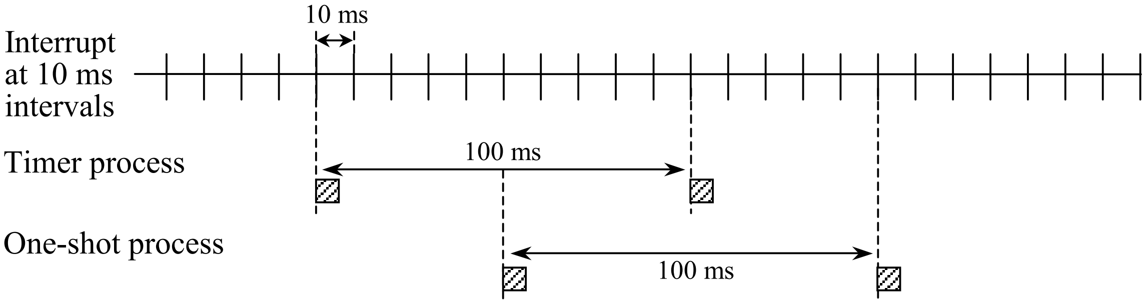


Figure 3-2 OS Process Timing

The timer and one-shot are processed in units of 16 points as shown below:

Processing Time of Timer 16 Points: 250μs/16 points

Processing Time of One-Shot 16 Points: 300μs/16 points

The OS basic processing time at every 10 ms: 250μs, approx.

Thus, the processing time appearing at every 100 ms is:

$$\text{Timer} = 250+250\times(\frac{\text{Timer Points}}{16})$$

$$\text{One-Shot} = 250+300\times(\frac{\text{One-Shot Points}}{16})$$

When the 10 ms timer is used, the following timer processing time is required at every 10 ms:

250+250 = 500μs/16 Points

(6) Remote I/O Support Time

Upon completion of the remote I/O transfer of 1 cycle, the OS restarts the remote I/O after saving its error information. Figure 3-3 shows the relationship between the remote I/O processing time and OS processing at this time.

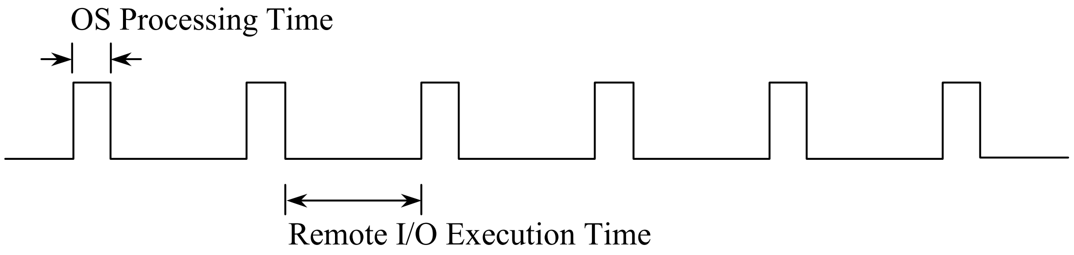


Figure 3-3 Remote I/O and OS Process

Although the remote I/O points may be changed by the edition setting, the relation between the number of points, the OS processing time, and the remote I/O execution time differs as follows:

Number of Remote I/O Points	OS Processing Time [ms]	Remote I/O Execution Time [ms]
512	0.2+0.2=0.4	5.0
1024	0.2+0.4=0.6	10.0
1536	0.2+0.6=0.8	15.0
2048	0.2+0.8=1.0	20.0

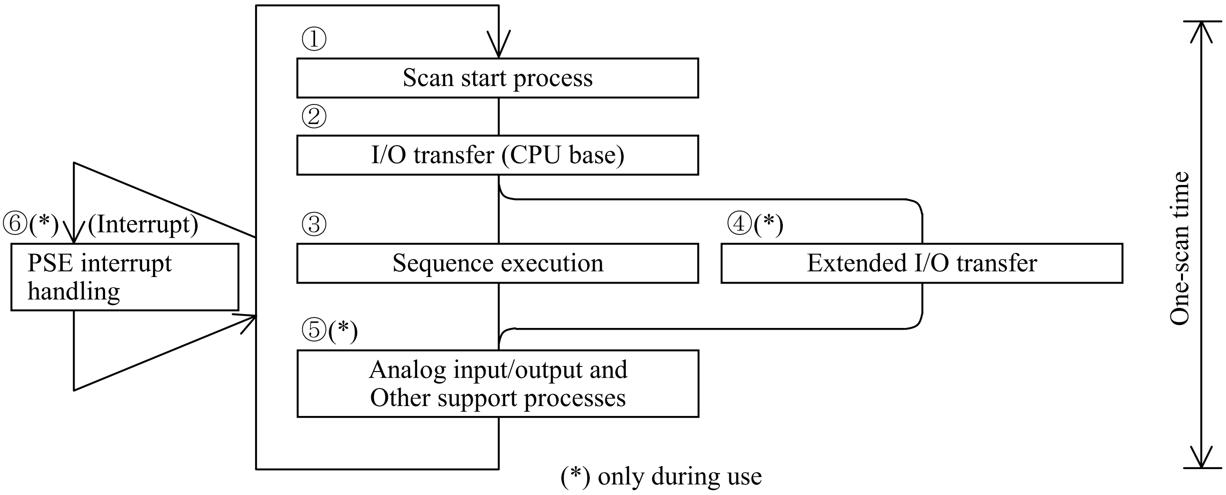
- To use the analog/pulse counter, etc., it is necessary to add the processing time of each support OS to the values shown in the above table.
  For the support OS processing time, refer to the manuals for analog/pulse counter, etc.

3.2 Processing Time for 4α

3.2.1 Scan time

During a sequence run, the PCs execute an I/O transfer process (data transfer between I/O memory and input/output modules), sequence program, and other processes rapidly and repeatedly.

As shown in Figure 3-4, the time required for a single execution, one-scan time (T), can be determined by the equation shown below.



(*) only during use

Figure 3-4 PCs Processing Flow

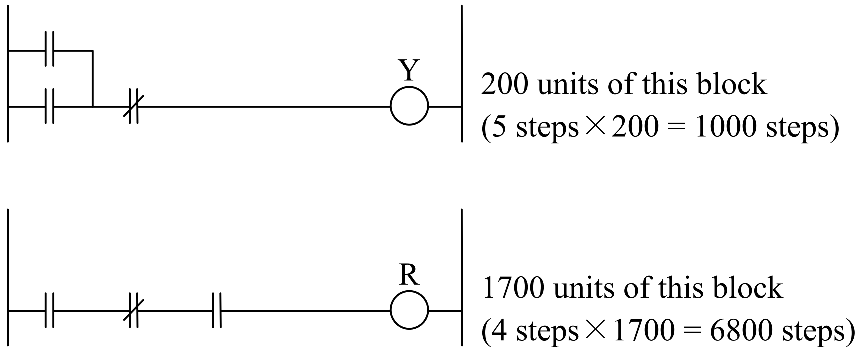
T = ①+②+max (③, ④) +⑤+⑥

→ The value ③ or ④ is used, whichever greater.

Table 3-1 shows the time required for processes ① through ⑥.

3.2.2 Scan time calculation example

Figure 3-5 shows a scan time calculation example. (In this example, it is assumed that the CPU base has eight basic slots and special I/O module (analog input/output module etc.), does not use a 10-ms time setting, and is not provided with a PS monitor.)



Scan time (T) = 2.0 ms(①)+2.1 ms (②)
+{2μs×5×200+2μs×4×1700} (③)
= 19.7 ms

Figure 3-5 Scan Time Calculation Example

Table 3-1 Processing Time List

No.	Item			Processing time (maximun)	Remarks
①	Scan start distance			2.0 ms	
②	I/O transfer (CPU base)		4-slot	1.1 ms	DI/O and I/O transfer within CPU base
			8-slot	2.1 ms	
③	Sequence execution	─┤├─ , ─┤/├─		2μs	Associated processing times are added in accordance with the number of employed instruction words.
		─┤↑├─ , ─┤↓├─		2μs	
		─(S)─ , ─(R)─		2μs	
		─( )─	T	(ON state) 210μs (OFF state) 80μs	
			U	(ON state) 300μs (OFF state) 180μs	
			CU CD CR	(ON→OFF state) 240μs (Other) 4μs	
			Other	2μs	
		─[F]─		(ON state)	See “2.4 Arithmetic Function List. ”
				(OFF state) 8μs	
		│ , ─┤ , ┐		2μs	
④	Extended I/O transfer		4-slot CPU base	11.6 ms	Only when an extended I/O module (LWE800 or LWE805) is installed in the CPU base with extension slots.
			8-slot CPU base	8.0 ms	
⑤	Analog input/output and other support processes			1.2 ms	Only when the analog input/output and preset counter modules are used.
⑥	PSE interrupt process			The scan time may be increased by several milliseconds to several hundred milliseconds at the beginning of PSE remote connection or during a read or monitoring process.	
⑦	Other • 10-ms timer • Host computer			T = T×1.1 Approx. 5 ms	(Only when a 10-ms timer is used) (Only when the host computer is connected)

3.3 Display of Average Scan Time

The average program scan time of systems can be seen through PSE by adding a program shown in Figure 3-6. The average scan time may be displayed at every 8 seconds in the user program RUN state.

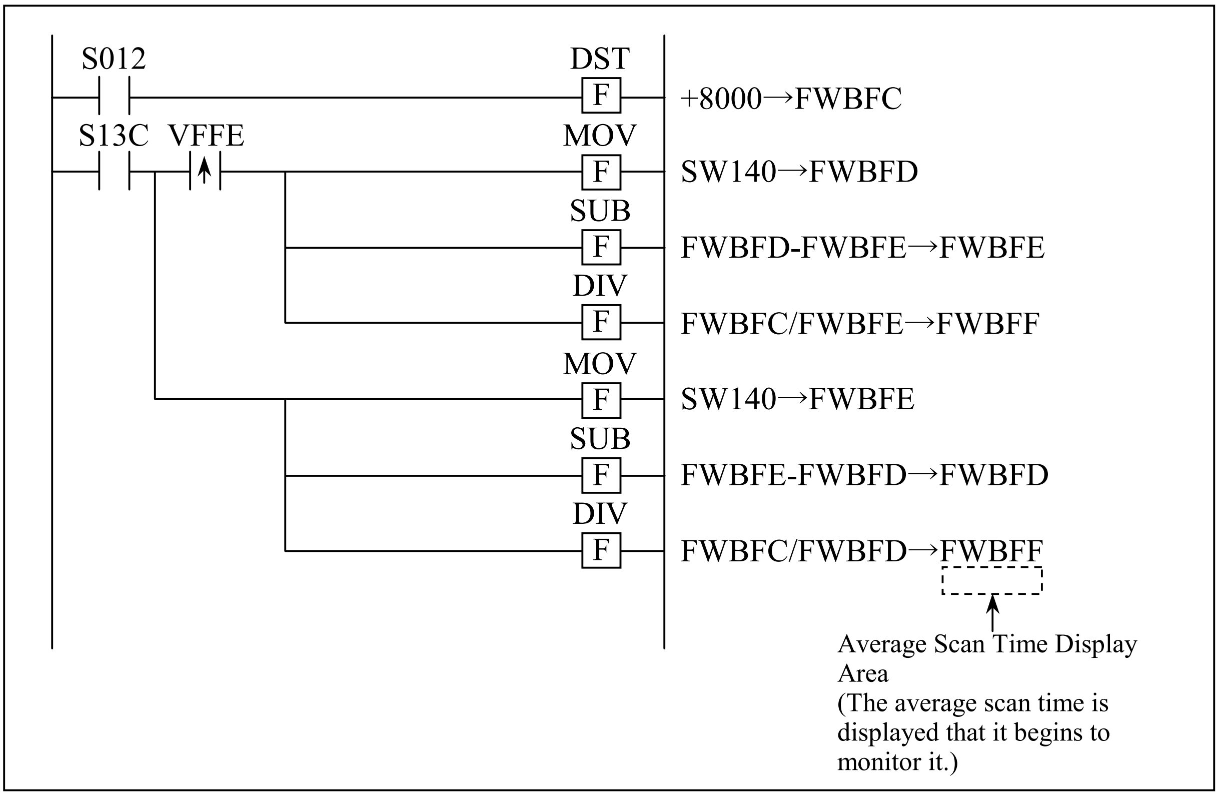


Figure 3-6 Scan Time Display Program

INDEX

INDEX