User's
Manual

FA-M3 VITESSE™

Sequence CPU Instruction Manual – Instructions

IM 34M06P12-03E

vigilantplant.®

YOKOGAWA
Yokogawa Electric Corporation

IM 34M06P12-03E
5th Edition

Blank Page

Applicable Product

● Range-free Multi-controller FA-M3

- Model Name: F3SP05, F3SP08, F3SP21, F3SP25, F3SP35,
  F3SP22, F3SP28, F3SP38, F3SP53, F3SP58, F3SP59
- Name: Sequence CPU Modules

- Model Name: F3SP66, F3SP67, F3SP71, F3SP76
- Name: Sequence CPU Modules (with network functions)

The document number and document model code for this manual are given below. Refer to the document number in all communications, including when purchasing additional copies of this manual.

Document No.	:	IM 34M06P12-03E
Document Model Code	:	DOCIM

Media No. IM 34M06P12-03E (CD) 5th Edition : Jan. 31, 2012 (YHQ)

Important

■ About This Manual

- This Manual should be passed on to the end user.
- Before using the controller, read this manual thoroughly to have a clear understanding of the controller.
- This manual explains the functions of this product, but there is no guarantee that they will suit the particular purpose of the user.
- Under absolutely no circumstances may the contents of this manual be transcribed or copied, in part or in whole, without permission.
- The contents of this manual are subject to change without prior notice.
- Every effort has been made to ensure accuracy in the preparation of this manual. However, should any errors or omissions come to the attention of the user, please contact the nearest Yokogawa Electric representative or sales office.

■ Safety Precautions when Using/Maintaining the Product

- The following safety symbols are used on the product as well as in this manual.

⚠

Danger. This symbol on the product indicates that the operator must follow the instructions laid out in this user’s manual to avoid the risk of personnel injuries, fatalities, or damage to the instrument. Where indicated by this symbol, the manual describes what special care the operator must exercise to prevent electrical shock or other dangers that may result in injury or the loss of life.

⏚

Protective Ground Terminal. Before using the instrument, be sure to ground this terminal.

⏚

Function Ground Terminal. Before using the instrument, be sure to ground this terminal.

∼

Alternating current. Indicates alternating current.

⎓

Direct current. Indicates direct current.

The following symbols are used only in the user’s manual.

WARNING

Indicates a “Warning”.

Draws attention to information essential to prevent hardware damage, software damage or system failure.

CAUTION

Indicates a “Caution”

Draws attention to information essential to the understanding of operation and functions.

TIP

Indicates a “TIP”

Gives information that complements the present topic.

SEE ALSO

Indicates a “SEE ALSO” reference.

Identifies a source to which to refer.

- For the protection and safe use of the product and the system controlled by it, be sure to follow the instructions and precautions on safety stated in this manual whenever handling the product. Take special note that if you handle the product in a manner other than prescribed in these instructions, the protection feature of the product may be damaged or impaired. In such cases, Yokogawa cannot guarantee the quality, performance, function and safety of the product.
- When installing protection and/or safety circuits such as lightning protection devices and equipment for the product and control system as well as designing or installing separate protection and/or safety circuits for fool-proof design and fail-safe design of processes and lines using the product and the system controlled by it, the user should implement it using devices and equipment, additional to this product.
- If component parts or consumable are to be replaced, be sure to use parts specified by the company.
- This product is not designed or manufactured to be used in critical applications which directly affect or threaten human lives and safety — such as nuclear power equipment, devices using radioactivity, railway facilities, aviation equipment, shipboard equipment, aviation facilities or medical equipment. If so used, it is the user’s responsibility to include in the system additional equipment and devices that ensure personnel safety.
- Do not attempt to modify the product.

■ Exemption from Responsibility

- Yokogawa Electric Corporation (hereinafter simply referred to as Yokogawa Electric) makes no warranties regarding the product except those stated in the WARRANTY that is provided separately.
- Yokogawa Electric assumes no liability to any party for any loss or damage, direct or indirect, caused by the use or any unpredictable defect of the product.

■ Software Supplied by the Company

- Yokogawa Electric makes no other warranties expressed or implied except as provided in its warranty clause for software supplied by the company.
- Use the software with one computer only. You must purchase another copy of the software for use with each additional computer.
- Copying the software for any purposes other than backup is strictly prohibited.
- Store the original media that contain the software in a safe place.
- Reverse engineering, such as decompiling of the software, is strictly prohibited.
- Under absolutely no circumstances may the software supplied by Yokogawa Electric be transferred, exchanged, or sublet or leased, in part or as a whole, for use by any third party without prior permission by Yokogawa Electric.

■ General Requirements for Using the FA-M3 Controller

● Avoid installing the FA-M3 controller in the following locations:

- Where the instrument will be exposed to direct sunlight, or where the operating temperature exceeds the range 0°C to 55°C (32°F to 131°F).
- Where the relative humidity is outside the range 10 to 90%, or where sudden temperature changes may occur and cause condensation.
- Where corrosive or flammable gases are present.
- Where the instrument will be exposed to direct mechanical vibration or shock.
- Where the instrument may be exposed to extreme levels of radioactivity.

● Use the correct types of wire for external wiring:

- Use copper wire with temperature ratings greater than 75°C.

● Securely tighten screws:

- Securely tighten module mounting screws and terminal screws to avoid problems such as faulty operation.
- Tighten terminal block screws with the correct tightening torque as given in this manual.

● Securely lock connecting cables:

- Securely lock the connectors of cables, and check them thoroughly before turning on the power.

● Interlock with emergency-stop circuitry using external relays:

- Equipment incorporating the FA-M3 controller must be furnished with emergency-stop circuitry that uses external relays. This circuitry should be set up to interlock correctly with controller status (stop/run).

● Ground for low impedance:

- For safety reasons, connect the [FG] grounding terminal to a Japanese Industrial Standards (JIS) Class D Ground*1 (Japanese Industrial Standards (JIS) Class 3 Ground). For compliance to CE Marking, use braided or other wires that can ensure low impedance even at high frequencies for grounding.

*1 Japanese Industrial Standard (JIS) Class D Ground means grounding resistance of 100 Ω max.

● Configure and route cables with noise control considerations:

- Perform installation and wiring that segregates system parts that may likely become noise sources and system parts that are susceptible to noise. Segregation can be achieved by measures such as segregating by distance, installing a filter or segregating the grounding system.

● Configure for CE Marking Conformance:

- For compliance to CE Marking, perform installation and cable routing according to the description on compliance to CE Marking in the “Hardware Manual” (IM 34M06C11-01E).

● Keep spare parts on hand:

- We recommend that you stock up on maintenance parts including spare modules.
- Preventive maintenance (replacement of the module or its battery) is required for using the module beyond 10 years. For enquiries on battery replacement service (for purchase), contact your nearest Yokogawa Electric representative or sales office. (The module has a built-in lithium battery. Lithium batteries may exhibit decreased voltage, and in rare cases, leakage problems after 10 years.)

● Discharge static electricity before operating the system:

- Because static charge can accumulate in dry conditions, first touch grounded metal to discharge any static electricity before touching the system.

● Never use solvents such as paint thinner for cleaning:

- Gently clean the surfaces of the FA-M3 controller with a cloth that has been soaked in water or a neutral detergent and wringed.
- Do not use volatile solvents such as benzine or paint thinner or chemicals for cleaning, as they may cause deformity, discoloration, or malfunctioning.

● Avoid storing the FA-M3 controller in places with high temperature or humidity:

- Since the CPU module has a built-in battery, avoid storage in places with high temperature or humidity.
- Since the service life of the battery is drastically reduced by exposure to high temperatures, take special care (storage temperature should be from –20°C to 75°C).
- There is a built-in lithium battery in a CPU module and temperature control module which serves as backup power supply for programs, device information and configuration information. The service life of this battery is more than 10 years in standby mode at room temperature. Take note that the service life of the battery may be shortened when installed or stored at locations of extreme low or high temperatures. Therefore, we recommend that modules with built-in batteries be stored at room temperature.

● Always turn off the power before installing or removing modules:

- Failing to turn off the power supply when installing or removing modules, may result in damage.

● Do not touch components in the module:

- In some modules you can remove the right-side cover and install ROM packs or change switch settings. While doing this, do not touch any components on the printed-circuit board, otherwise components may be damaged and modules may fail to work.

● Do not use unused terminals:

- Do not connect wires to unused terminals on a terminal block or in a connector. Doing so may adversely affect the functions of the module.

■ Waste Electrical and Electronic Equipment

Waste Electrical and Electronic Equipment (WEEE), Directive 2002/96/EC
(This directive is only valid in the EU.)

This product complies with the WEEE Directive (2002/96/EC) marking requirement. The following marking indicates that you must not discard this electrical/electronic product in domestic household waste.

Product Category
With reference to the equipment types in the WEEE directive Annex 1, this product is classified as a “Monitoring and Control instrumentation” product.
Do not dispose in domestic household waste.
When disposing products in the EU, contact your local Yokogawa Europe B. V. office.

■ How to Discard Batteries

The following description on DIRECTIVE 2006/66/EC (hereinafter referred to as the EU new directive on batteries) is valid only in the European Union.

Some models of this product contain batteries that cannot be removed by the user. Make sure to dispose of the batteries along with the product.

Do not dispose in domestic household waste.
When disposing products in the EU, contact your local Yokogawa Europe B. V. office.

Battery type: Lithium battery

Note: The symbol above means that the battery must be collected separately as specified in Annex II of the EU new directive on batteries.

Introduction

■ Overview of the Manual

This manual describes the instructions, which can be used in writing programs for the sequence CPU modules (F3SP05, F3SP08, F3SP21, F3SP22-0S, F3SP25, F3SP28-3N, F3SP28-3S, F3SP35, F3SP38-6N, F3SP38-6S, F3SP53-4H, F3SP53-4S, F3SP58-6H, F3SP58-6S, F3SP59-7S) and sequence CPU modules (with network functions) (F3SP66-4S, F3SP67-6S, F3SP71-4N, F3SP76-7N, F3SP71-4S, F3SP76-7S) designed for use with the Range-free Multi-controller FA-M3.

■ How to Read the Manual

First read the “Sequence CPU – Functions User's Manual” and then proceed to Chapter 1 of this manual. You may read relevant parts of Chapters 2 and 3 as and when required.

■ Other User’s Manuals

For individual sequence CPU modules, please refer to the relevant user's manuals.

F3SP71
F3SP76

● For information on functions, refer to:

- Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S) (IM 34M06P15-01E)
- Sequence CPU – Network Functions (for F3SP71-4N/4S, F3SP76-7N/7S) (IM 34M06P15-02E)

F3SP66
F3SP67

● For information on functions, refer to:

- Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S) (IM 34M06P14-01E)
- Sequence CPU – Network Functions (for F3SP66-4S, F3SP67-6S) (IM 34M06P14-02E)

F3SP22 F3SP53
F3SP28 F3SP58
F3SP38 F3SP59

● For information on functions, refer to:

- Sequence CPU Instruction Manual – Functions (for F3SP22-0S, F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S) (IM 34M06P13-01E)

F3SP05 F3SP21 F3SP25 F3SP35

● For information on functions, refer to:

- Sequence CPU – Functions (for F3SP21, F3SP25, F3SP35) (IM 34M06P12-02E)

All types of Sequence CPU Modules

● Specifications and Layout*1 of the FA-M3, Mounting and Wiring, Testing, Maintenance and Inspection, and System-wide Restrictions for Mounting Modules

*1: See specific manuals for products other than the power module, base module, I/O module, cables, and terminal block units.

- Hardware Manual (IM 34M06C11-01E)

● For information on the commands and responses of personal computer link functions, refer to:

- Personal Computer Link Commands (IM 34M06P41-01E)

● For information on creating ladder programs, refer to:

- FA-M3 Programming Tool WideField3 (IM 34M06Q16-01E, 02E, 03E, 04E)

● For information on the functions of the fiber-optic FA-bus modules, refer to:

- Fiber-optic FA-bus Module and Fiber-optic FA-bus Type 2 Module, FA-bus Type 2 Module (IM 34M06H45-01E)

● For information on the functions of the FA link H and fiber-optic FA link H modules, refer to:

- FA Link H Module, Fiber-optic FA Link H Module (IM 34M06H43-01E)

■ Notational Conventions

● Symbols Used

The following symbols are used in this manual:

F3SP25 F3SP35	: Available for the F3SP25 and F3SP35 sequence CPU modules.
F3SP22 F3SP28 F3SP38	: Available for the F3SP22-0S, F3SP28-3N, F3SP28-3S, F3SP38-6N and F3SP38-6S sequence CPU modules.
F3SP53 F3SP58 F3SP59	: Available for the F3SP53-4H, F3SP53-4S, F3SP58-6H, F3SP58-6S and F3SP59-7S sequence CPU modules.
F3SP22-0S F3SP53-4S F3SP28-3S F3SP58-6S F3SP38-6S F3SP59-7S	: Available for the F3SP22-0S, F3SP28-3S, F3SP38-6S, F3SP53-4S, F3SP58-6S and F3SP59-7S sequence CPU modules.
F3SP66 F3SP67	: Available for the F3SP66-4S and F3SP67-6S sequence CPU modules (with network functions).
F3SP71 F3SP76	: Available for the F3SP71-4N, F3SP76-7N, F3SP71-4S and F3SP76-7S sequence CPU modules (with network functions).
F3SP71-4S F3SP76-7S	Available for the F3SP71-4S and F3SP76-7S sequence CPU modules (with network functions).

No mark : Available for all sequence CPU modules (F3SP05, F3SP08, F3SP21, F3SP22, F3SP25, F3SP28, F3SP35, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66, F3SP67, F3SP71 and F3SP76).

Copyrights and Trademarks

■ Copyrights

■ Trademarks

Blank Page

FA-M3
Sequence CPU Instruction Manual- Instructions

IM 34M06P12-03E 5th Edition

CONTENTS

1. General Description

This chapter provides an outline of the instructions for the sequence CPU modules. Please refer to Chapter 2 and Chapter 3 for detailed descriptions of the instructions.

1.1 Instruction and Program Size

The maximum program capacity is

F3SP05	5K steps (5,120 steps)
F3SP08, F3SP21, F3SP22	10K steps (10,240 steps)
F3SP25	20K steps (20,480 steps)
F3SP35	100K steps (102,400 steps)
F3SP28	30K steps (30,720 steps)
F3SP53, F3SP66	56K steps (57,344 steps)
F3SP71	60K steps (61,440 steps)
F3SP38, F3SP58, F3SP67	120K steps (122,880 steps)
F3SP59	254K steps (260,096 steps)
F3SP76	260K steps (266,240 steps)

An instruction consists of one to seven steps. Consequently, the number of instructions that can be contained in a program varies according to the type of instructions used.

SEE ALSO

See Chapter 2, "Basic Instructions," and Chapter 3, "Application Instructions," for the relationship between the number of steps and instructions.

1.2 Bit Manipulation

Bit manipulation is performed when a basic instruction specifying a bit device (X, Y, I, E, T, C, L, or M) is executed. Bit manipulation is executed on a bit basis.

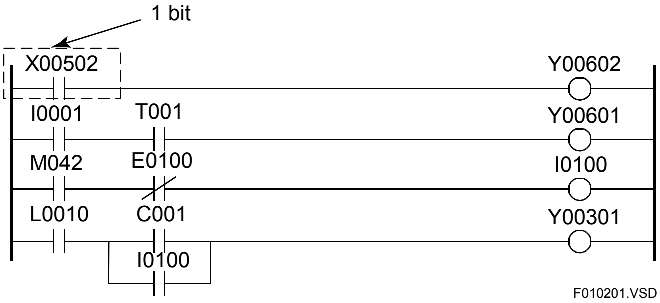


Figure 1.2.1 Outline of a Bit Manipulation

1.3 Word Manipulation (16 bits)

Word manipulation is a process of manipulating target devices on a 16-bit basis. An instruction processes points in 16-point units if bit devices (X, Y, I, E, L, and M) are specified in the instruction, and in single-point units if a word device (T, C, D, B, F, W, R, V, and Z) is specified in the instruction. A word-processing instruction can handle numbers from

-32768 to 32767 (in decimal) or $8000 to $7FFF (in signed hexadecimal).

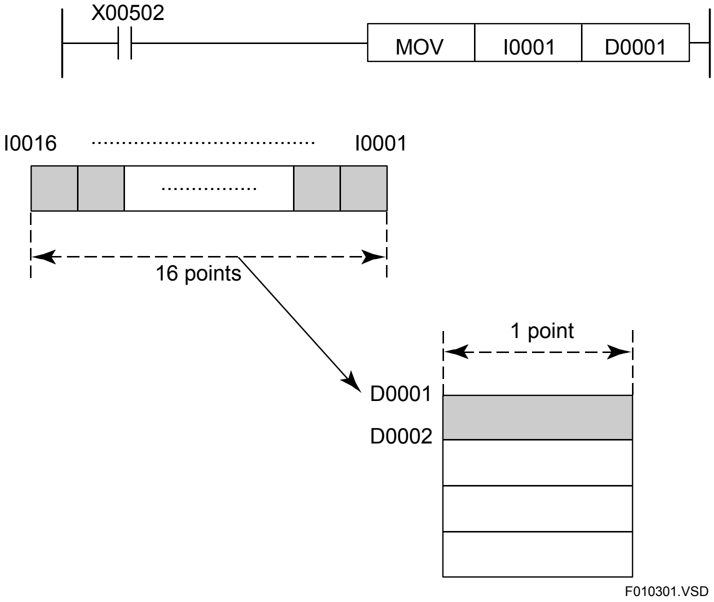


Figure 1.3.1 Outline of a Word Manipulation

■ Input/output Relay (Word Manipulation on X/Y)

If the number of input/output relay points, which starts at the specified relay number, is less than 16, the value of the bits corresponding to the empty bit positions is unpredictable (0 or 1).

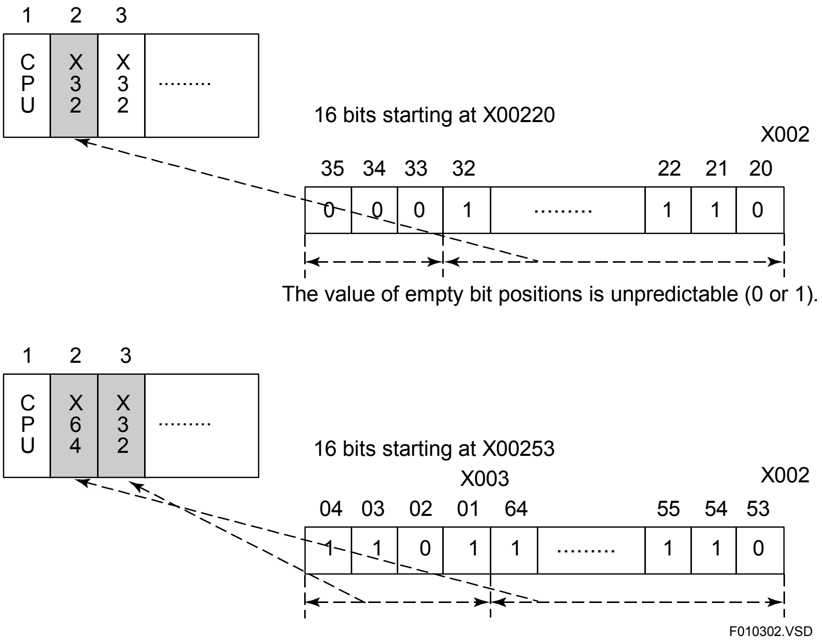


Figure 1.3.2 Input/output Relays

SEE ALSO

For details on the input/output relays, see Section 4.1 of "Sequence CPU Instruction Manual – Functions (for F3SP22-0S, F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S)" (IM 34M06P13-01E), Section A4.1 of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E), or Section A4.1 of "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

CAUTION

If bit positions whose state is unpredictable (0 or 1) are likely to cause problems in an application, they should be masked off as required.

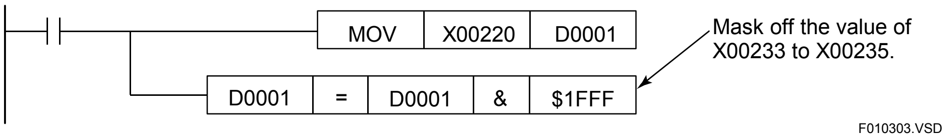


Figure 1.3.3 Masking

1.4 Long Word Manipulation (32 bits)

Long word manipulation refers to a process of processing target devices on a 32-bit basis. An instruction processes points in 32-point units if bit devices (X, Y, I, E, L, and M) are specified in the instruction and in 2-point units if word devices (D, B, F, W, R, V, and Z) are specified in the instruction. A long-word-processing instruction can handle numbers from -2147483648 to 2147483647 (in decimal) or $80000000 to $7FFFFFFF (in signed hexadecimal).

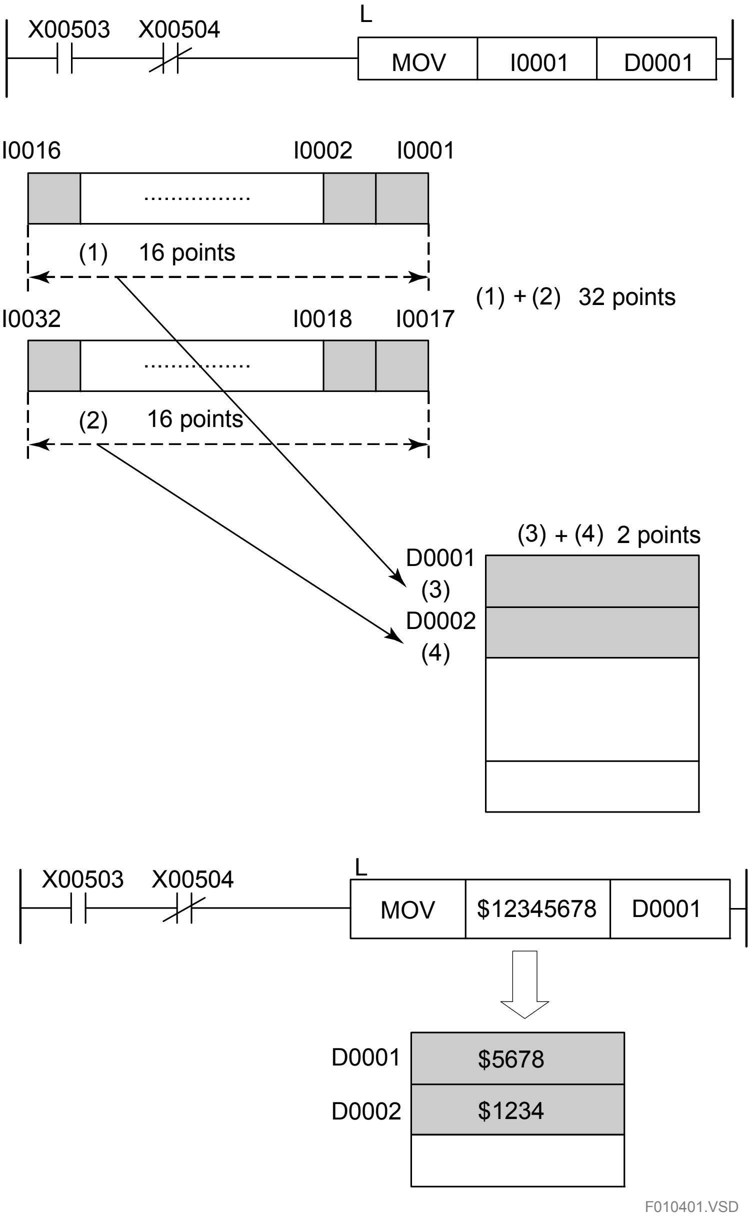


Figure 1.4.1 Outline of a Long Word Manipulation

■ Input/output Relay (Long Word Manipulation on X/Y)

If the number of input/output relay points, which starts at the specified relay number, is less than 32, the value of the bits corresponding to the empty bit positions is unpredictable (0 or 1).

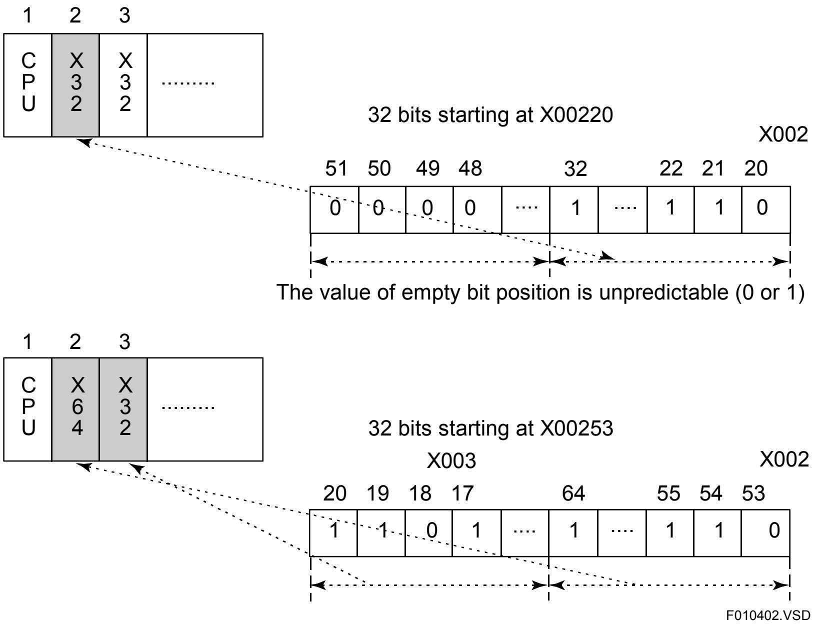


Figure 1.4.2 Input/output Relays

CAUTION

If bit positions whose state is unpredictable (0 or 1) are likely to cause problems in an application, they should be masked off as required.

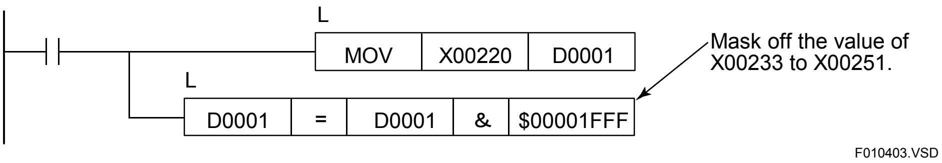
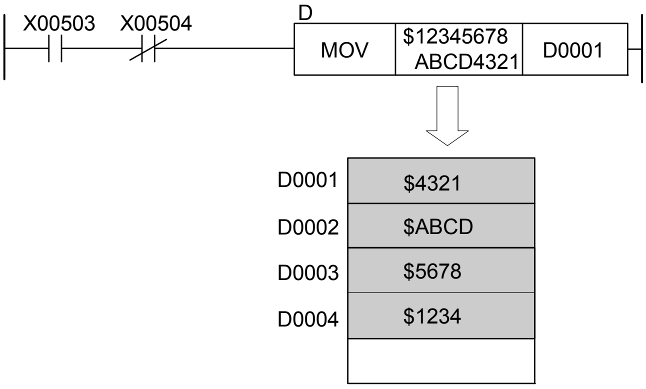


Figure 1.4.3 Masking

1.5 Double Long Word Manipulation (64 bits)

F3SP71
F3SP76

Double long word manipulation refers to a process of processing target devices on a 64-bit basis. This instruction processes points in the register device (D, B, F, W, and R) in 4-point units. A double long-word-processing instruction can handle numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 (in decimal) or $8000000000000000 to $7FFFFFFFFFFFFFFF (in signed hexadecimal). In double long-word processing, any bit devices (X, Y, I, E, L, and M) cannot be used as a specified device.



F010401_1.VSD

Figure 1.5.1 Outline of a Double Long Word Manipulation

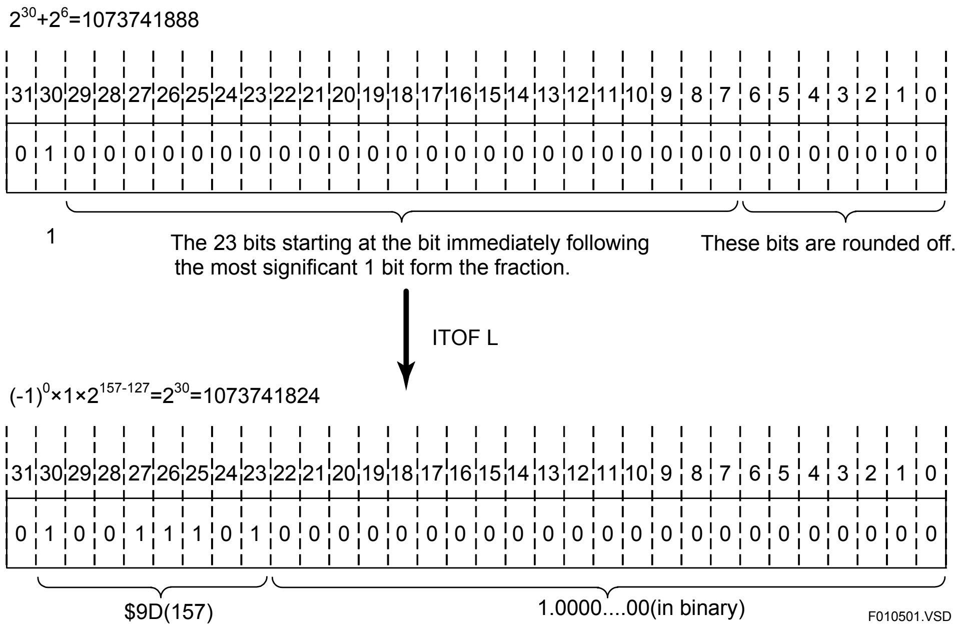
1.6 Floating-point Processing

F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

■ Value Range of Floating-point Numbers

-2^{128} to $+2^{128}$ (Approx. -3.4×10^{38} to $+3.4 \times 10^{38}$)

Since the fraction is represented by 23 bits, the number of decimal significant digits of floating-point numbers is approximately 6 to 7 digits. Long-word integers are rounded as shown below when they are converted to floating-point numbers.



$2^{30}+2^{6}$=1073741888

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1
The 23 bits starting at the bit immediately following the most significant 1 bit form the fraction.
These bits are rounded off.
ITOF L

$(-1)^{0}\times1\times2^{157-127}=2^{30}$=1073741824

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$9D(157)
1.0000....00(in binary)
F010501.VSD

Figure 1.6.1 Value Range of Floating-point Numbers

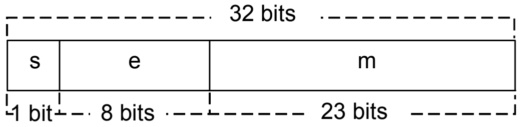
When a floating-point number is converted to an integer or long-word integer, its fraction is rounded off.

■ Floating-point Arithmetic Instructions

A floating-point arithmetic instruction can contain neither integer nor long-word integer. Any integer or long-word integer to be specified in a floating-point arithmetic instruction must be converted to a floating-point number with the ITOF instruction before being specified in the floating-point arithmetic instruction. A rounding error always results from a floating-point operation. Consequently, the programmer should do programming while taking rounding errors into consideration. The result of a floating-point operation whose value is smaller than 2^{-127} is rounded to 0. An instruction error is raised if the result of a floating-point operation exceeds the valid value range of floating-point numbers, that is, -2^{128} to $+2^{128}$.

■ Internal Representation of Floating-point Numbers

Floating-point data is represented in the IEEE single-precision format as shown below.



s : Represents the sign (1 bit).
0 : +
1 : -
e : Represents the exponent (8 bits).
m : Represents the fraction (23 bits).

F010502.VSD

Figure 1.6.2 Internal Representation of Floating-point Numbers

(1) If $e \neq 0$, single-precision data type = $(-1)^{s} \times 1.m \times 2^{e-127}$

(2) If $e = 0$, single-precision data type = 0 if $m = 0$ (all bits being zeros represent the number 0).

1.7 Double-precision Floating-point Processing

F3SP71
F3SP76

■ Value Range of Double-precision Floating-point Numbers

-2^{1023} to $+2^{1023}$ (Approx. -1.79×10^{308} to $+1.79 \times 10^{308}$)

Since the fraction is represented by 52 bits, the number of decimal significant digits of floating-point numbers is approximately 15 to 16 digits. Double long-word integers are rounded as shown below when they are converted to double-precision floating-point numbers.

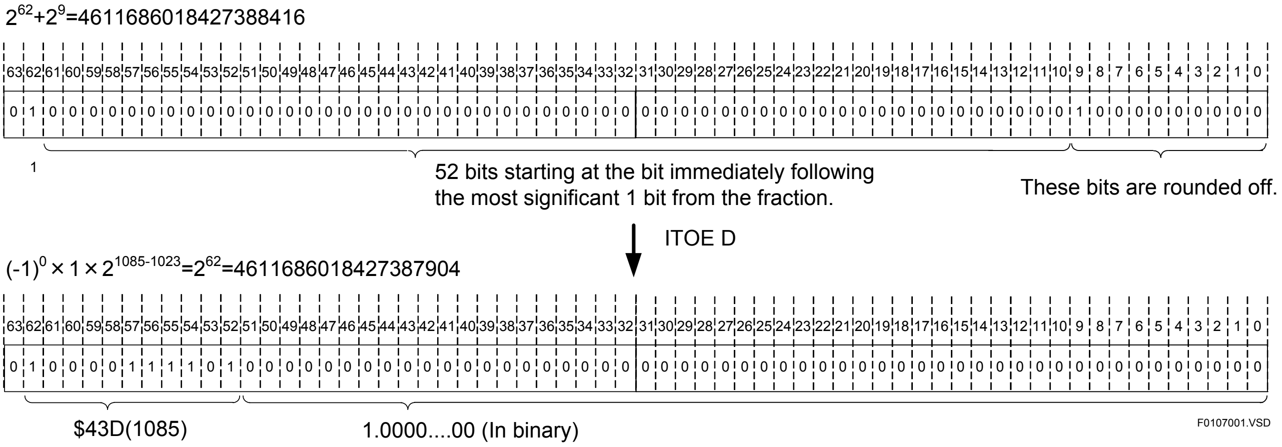


Figure 1.7.1 Value Range of Double-precision Floating-point Numbers

When a double-precision floating-point number is converted to a long-word integer or double long-word integer, its fraction is rounded off.

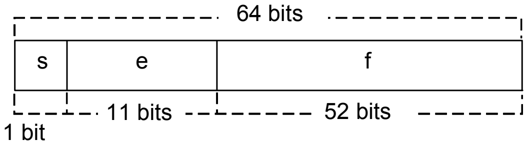
■ Double-precision Floating-point Arithmetic Instructions

A double-precision floating-point arithmetic instruction can contain neither integer, long-word integer, nor double long-word integer. Any long-word or double long-word integer to be specified in a double-precision floating-point arithmetic instruction must be converted to a double-precision floating-point number with the ITOE instruction before being specified in the floating-point arithmetic instruction.

A rounding error always results from a double-precision floating-point operation. Consequently, the programmer should do programming while taking rounding errors into consideration. The result of a double-precision floating-point operation whose absolute value is smaller than 2^{-1023} is rounded to 0.

■ Internal Representation of Double-precision Floating-point Numbers

Double-precision floating-point data is represented in the IEEE double-precision format (IEEE754) as shown below.



s : Represents the sign (1 bit).
0 : +
1 : -
e : Represents the exponent (11 bits).
f : Represents the fraction (52 bits).

F010504.VSD

Figure 1.7.2 Internal Representation of Double-precision Floating-point Numbers

(1) If e ≠ 0, double-precision data type = $(-1)^s$ x 1.f x 2^{e-1023}

(2) If e = 0, double-precision data type = 0 if f = 0 (all bits being zeros represent the number 0).

1.8 String Manipulation

F3SP25 F3SP35	F3SP22 F3SP28 F3SP38	F3SP53 F3SP58 F3SP59	F3SP66 F3SP67	F3SP71 F3SP76

■ String Handling

A string is manipulated on a byte (8 bits) basis. A string is terminated by a $00. Since the maximum string length is 2,047 characters, any string longer than 2,047 characters may not be manipulated properly.

A string literal of 1 to 4 bytes may be specified as the destination of a SMOV instruction. String literals of 1 to 2 bytes may be specified in string manipulation instructions other than SMOV.

Any string literal (e.g., "ABC") appearing in a string manipulation instruction is left justified. A string literal specified in an instruction other than the string manipulation instructions is right justified as it is handled as ASCII-coded numeric data.

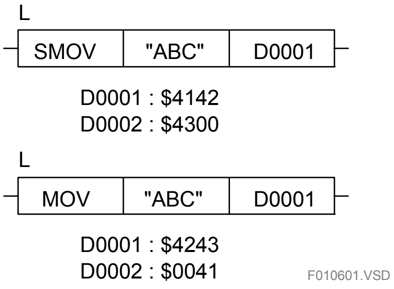


Figure 1.8.1 String Handling

1.9 High-speed Processing of Application Instructions

1.9.1 When Using the F3SP05, F3SP08, F3SP21, F3SP25 or F3SP35

High-speed processing application instructions are high-speed versions of application instructions whose execution speed is augmented by imposing conditions on the devices to be specified in the instruction. Such instructions include the MOV, CAL, CMP, and 16-bit logical instructions.

Devices must be specified as explained below in high-speed application instructions.

(1) When Par1, Par2, and Par3 in Figure 1.9.1 are bit devices (X, Y, I, E, L, H, and M)

- Do not use index modification.
- Use device numbers 1, 17, 33, 49, and so forth.
- When using X and Y relays, set the data code type to BIN.

(2) When Par1, Par2, and Par3 in Figure 1.9.1 below are word devices (D, R, W, Z, T, C, and A) excluding file registers (B)

- Do not use index modification.
- When using link registers (W), use W1 to W1024.

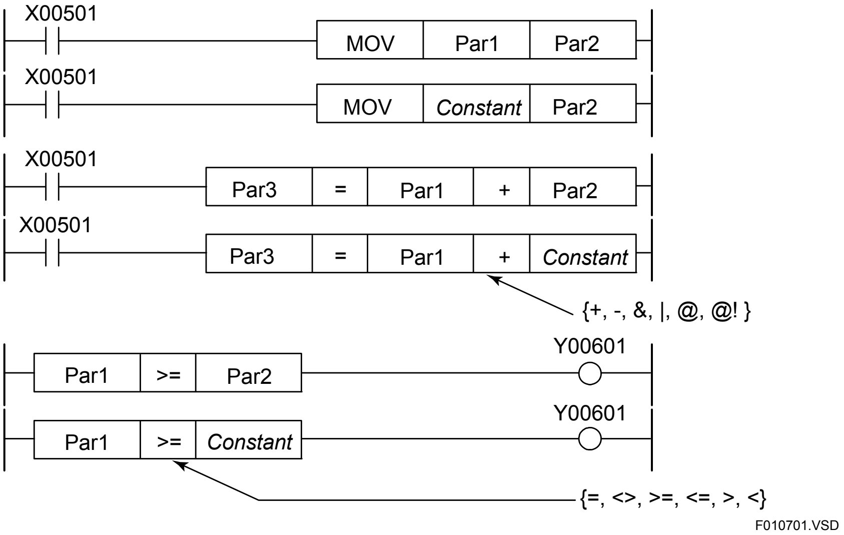
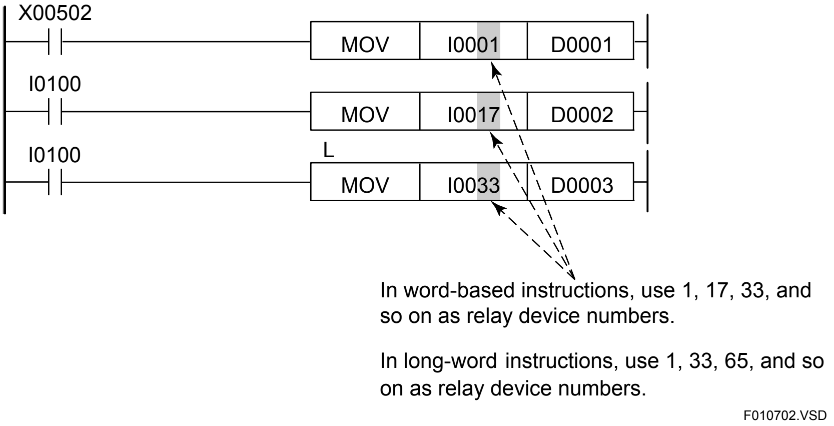


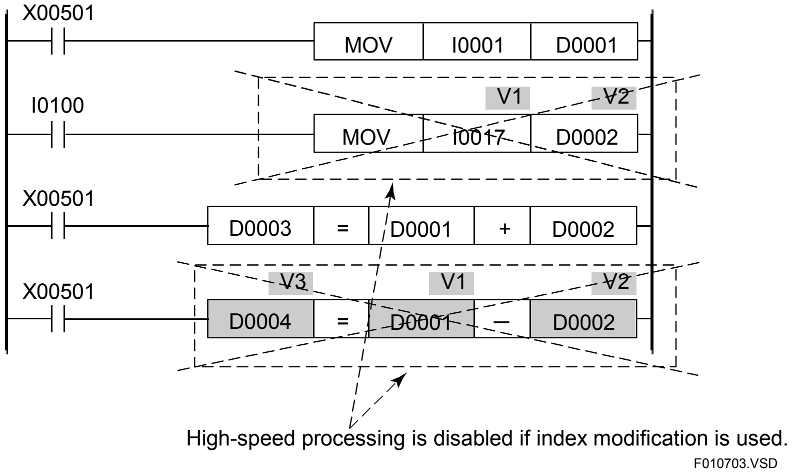
Figure 1.9.1 Parameter Specification Conditions for High-speed Application Instructions



In word-based instructions, use 1, 17, 33, and so on as relay device numbers.

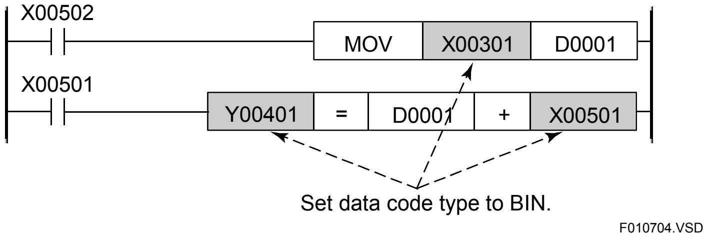
In long-word instructions, use 1, 33, 65, and so on as relay device numbers.

Figure 1.9.2 Example of High-speed Processing of Application Instructions (1)



High-speed processing is disabled if index modification is used.

Figure 1.9.3 Example of High-speed Processing of Application Instructions (2)



Set data code type to BIN.

Figure 1.9.4 Example of High-speed Processing of Application Instructions (3)

1.9.2 When Using the F3SP22, F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 or F3SP67

■ Applicable Application Instructions

High-speed processing application instructions are high-speed versions of application instructions whose execution speed is augmented by imposing conditions on the devices to be specified in the instruction.

The table below lists the applicable application instructions.

Table 1.9.1 Applicable Application Instructions

Classification	FUNC NO.	Instruction	Processing Unit	Mnemonic
Comparison	10	Compare	16 bit	CMP
		Compare long-word data	32 bit	CMP L
Arithmetic operation	20/20P	Add	16 bit	CAL
		Subtract	16 bit	
		Multiply	16 bit	
		Divide	16 bit	
		Add long-word data	32 bit	CAL L
		Subtract long-word data	32 bit	
		Multiply long-word data	32 bit	
		Divide long-word data	32 bit	
	120/120P	Increment	16 bit	INC
	121/121P	Decrement	16 bit	DEC
Logical operation	20/20P	AND	16 bit	CAL
		OR	16 bit	
		XOR	16 bit	
		NXOR	16 bit	
		AND long-word data	32 bit	CAL L
Shift	32/32P	Shift right	16 bit	RSFT
	33/33P	Shift left	16 bit	LSFT
Data transfer	40/40P	Move	16 bit	MOV
		Move long-word data	32 bit	MOV L
Special module	81/81P	Read special module	16 bit	READ
		Read special module in long-word units	32 bit	READ L
	82/82P	Write special module	16 bit	WRITE
		Write special module in long-word units	32 bit	WRITE L

■ Device Specification

Devices must be specified as explained below in high-speed application instructions.

(1) Using Instructions in the table below

- Do not use index modification.
- When using file registers with the F3SP38, F3SP58, F3SP59 or F3SP67, use registers from B1 to B131072.
- When using bit devices (X, Y, I, E, L, H and M), use device numbers 1, 17, 33, 49, and so on.
- When using X and Y relays, set the data code type to BIN.

Table 1.9.2 Device Specification (1)

Classification	FUNC NO.	Instruction	Processing Unit	Mnemonic
Comparison	10	Compare	16 bit	CMP
		Compare long-word data	32 bit	CMP L
Arithmetic operation	20/20P	Add	16 bit	CAL
		Subtract	16 bit	
		Multiply	16 bit	
		Divide	16 bit	
		Add long-word data	32 bit	CAL L
		Multiply long-word data	32 bit	
	120/120P	Increment	16 bit	INC
	121/121P	Decrement	16 bit	DEC
Logical operation	20/20P	AND	16 bit	CAL
		OR	16 bit	
		XOR	16 bit	
		AND long-word data	32 bit	CAL L
Data transfer	40/40P	Move	16 bit	MOV
		Move long-word data	32 bit	MOV L

(2) Using Instructions in the table below

- Includes the conditions in (1).
- When using constants, their positions must satisfy the following conditions.

Table 1.9.3 Device Specification (2)

Classification	FUNC NO.	Instruction	Processing Unit	Example of Conditions for Literal Position

Arithmetic operation	20/20P	Subtract long-word data	32 bit	D1=D3-1 ✓
				D1=1-D3 ×

		Divide long-word data	32 bit	D1=D5/2 ✓
				D1=2/D5 ×

Logical operation	20/20P	XNOR	16 bit	When using literals high-speed processing of application instructions is not available.

(3) Using Instructions in the table below

- High-speed processing of application instructions is available only when the conditions in the following table are satisfied. D1 must satisfy all conditions mentioned in (1).

Table 1.9.4 Device Specification (3)

Classification	FUNC NO.	Instruction	Processing Unit	Example of Conditions for Literal Position
Shift	32/32P	Shift right	16 bit	RSFT D1 2 ✓ RSFT D1 D2 ×
	33/33P	Shift left	16 bit	LSFT D1 2 ✓ LSFT D1 D2 ×
Special module	81/81P	Read special module	16 bit	READ 2 1 D1 1
		Read special module in long-word units	32 bit	READ L 2 1 D1 1
	82/82P	Write special module	16 bit	WRITE D1 2 1 1
		Write special module in long-word units	32 bit	WRITE L D1 2 1 1

1.9.3 When Using the F3SP71 or F3SP76

■ Applicable Application Instructions

High-speed processing application instructions are high-speed versions of application instructions and part of basic instructions whose execution speed is augmented by imposing conditions on the devices to be specified in the instruction.

The table below lists the applicable application instructions.

Table 1.9.5 Applicable Application Instructions (1/2)

Classification	FUNC NO.	Instruction	Processing Unit	Mnemonic
Basic	311	Load specified bit	16 bit	LDW
		Load specified long-word bit	32 bit	LDW L
	312	Out specified bit	16 bit	OUTW
		Out specified long-word bit	32 bit	OUTW L
	313	Set specified bit	16 bit	SETW
		Set specified long-word bit	32 bit	SETW L
	314	Reset specified bit	16 bit	RSTW
		Reset specified long-word bit	32 bit	RSTW L
Comparison	10	Compare	16 bit	CMP
		Compare long-word data	32 bit	CMP L
		Compare double long-word data	64 bit	CMP D
	904	Compare float	32 bit	FCMP
		Compare double-precision float	64 bit	FCMP E
Arithmetic operation	20/20P	Add	16 bit	CAL
		Subtract	16 bit	
		Multiply	16 bit	
		Divide	16 bit	
		Add long-word data	32 bit	CAL L
		Subtract long-word data	32 bit	
		Multiply long-word data	32 bit	
		Divide long-word data	32 bit	
		Add double long-word data	64 bit	CAL D
		Subtract double long-word data	64 bit	
	903/903P	Add float	32 bit	FCAL
		Subtract float	32 bit	
		Multiply float	32 bit	
		Divide float	32 bit	
		Add double-precision float	64 bit	FCAL E
		Subtract double-precision float	64 bit	
		Multiply double-precision float	64 bit	
		Divide double-precision float	64 bit	
	120/120P	Increment	16 bit	INC
		Increment long-word data	32 bit	INC L
	121/121P	Decrement	16 bit	DEC
		Decrement long-word data	32 bit	DEC L

Table 1.9.5 Applicable Application Instructions (2/2)

Classification	FUNC NO.	Instruction	Processing Unit	Mnemonic
Logical operation	20/20P	AND	16 bit	CAL
		OR	16 bit	
		XOR	16 bit	
		NXOR	16 bit	
		AND long-word data	32 bit	CAL L
		OR long-word data	32 bit	
		XOR long-word data	32 bit	
		NXOR long-word data	32 bit	
	21/21P	Two's complement	16 bit	NEG
		Two's complement long-word	32 bit	NEG L
	22/22P	Not	16 bit	NOT
		Not long-word	32 bit	NOT L
Rotate	30/30P	Right rotate	16 bit	RROT
		Right rotate long-word	32 bit	RROT L
	31/31P	Left rotate	16 bit	LROT
		Left rotate long-word	32 bit	LROT L
Shift	32/32P	Right shift	16 bit	RSFT
		Right shift long-word	32 bit	RSFT L
	33/33P	Left shift	16 bit	LSFT
		Left shift long-word	32 bit	LSFT L
Data Processing	52/52P	Binary conversion	16 bit	BIN
		Long-word binary conversion	32 bit	BIN L
	53/53P	BCD conversion	16 bit	BCD
		Long-word BCD conversion	32 bit	BCD L
	901/901P	Integer to float	16 bit	ITOF
		Long-word integer to float	32 bit	ITOF L
	920/920P	Long-word integer to double-precision float	32 bit	ITOE
		Double long-word integer to double-precision float	64 bit	ITOE D
	902/902P	Float to integer	16 bit	FTOI
		Float to long-word integer	32 bit	FTOI L
	922/922P	Double-precision float to long-word integer	32 bit	ETOI L
		Double-precision float to double long-word integer	64 bit	ETOI D
	925/925P	Float to double-precision float	32 bit	FTOE
	926/926P	Double-precision float to float	64 bit	ETOF
Data transfer	40/40P	Move	16 bit	MOV
		Move long-word data	32 bit	MOV L
		Move double long-word data	64 bit	MOV D
	42/42P	Block move	n word	BMOV
	43/43P	Block set	n word	BSET
Special module	81/81P	Read special module	16 bit	READ
		Read special module in long-word units	32 bit	READ L
	82/82P	Write special module	16 bit	WRITE
		Write special module in long-word units	32 bit	WRITE L

■ Device Specification

Devices must be specified as explained below in high-speed application instructions.

(1) Using All Instructions in Table 1.9.5

+ If parameters for word (16 bits) processing are specified:
  - When using bit devices (X, Y, I, E, L, H and M), use device numbers 1, 17, 33, 49, and so on (16 x n + 1).
  - When using X and Y relays, set the data code type to BIN.

+ If parameters for long-word (32 bits) processing or floating-point (32 bits) processing are specified:
  - When using bit devices (X, Y, I, E, L, H and M), use device numbers 1, 33, 65, 97, and so on (32 x n + 1).
  - When using register devices (D, B, F, W, R, V, A, U, and Z), use odd numbers 1, 3, 5, 7, and so on (2 x n + 1).
  - When using X and Y relays, set the data code type to BIN.

+ If parameters for double long-word (64 bits) processing or double-precision floating-point (64 bits) processing are specified:
  - When using register devices (D, B, F, W, R, and A), use odd numbers 1, 3, 5, 7, and so on (2 x n + 1).

CAUTION

If your instructions are not included in Table 1.9.5, they operate faster if they meet condition (1).

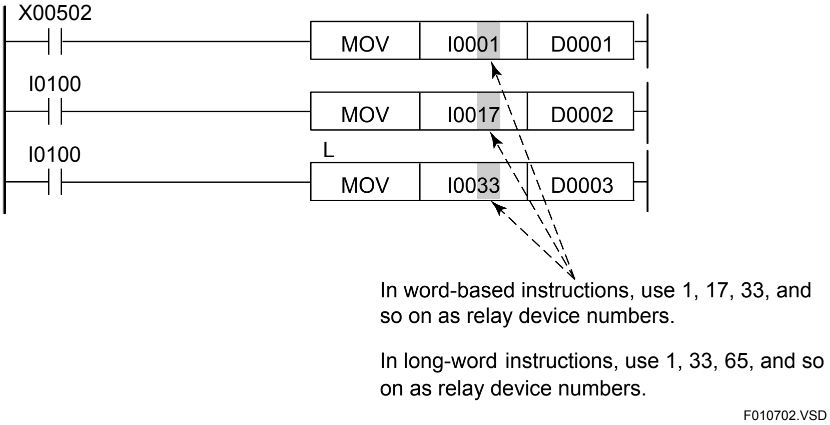

X00502
MOV
I0001
D0001
I0100
MOV
I0017
D0002
I0100
L
MOV
I0033
D0003
In word-based instructions, use 1, 17, 33, and so on as relay device numbers.
In long-word instructions, use 1, 33, 65, and so on as relay device numbers.
F010702.VSD

Figure 1.9.5 Example of High-speed Processing of Application Instructions (1)

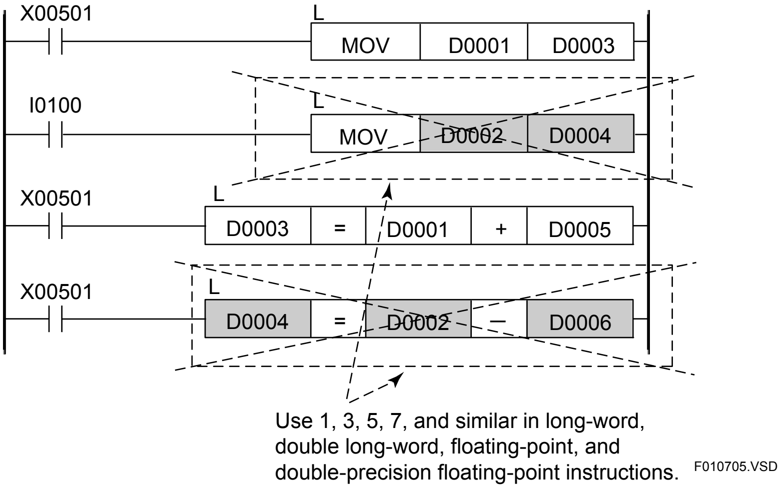

X00501
L
MOV
D0001
D0003
I0100
L
MOV
D0002
D0004
X00501
L
D0003
=
D0001
+
D0005
X00501
L
D0004
=
D0002
–
D0006
Use 1, 3, 5, 7, and similar in long-word, double long-word, floating-point, and double-precision floating-point instructions.
F010705.VSD

Figure 1.9.6 Example of High-speed Processing of Application Instructions (2)

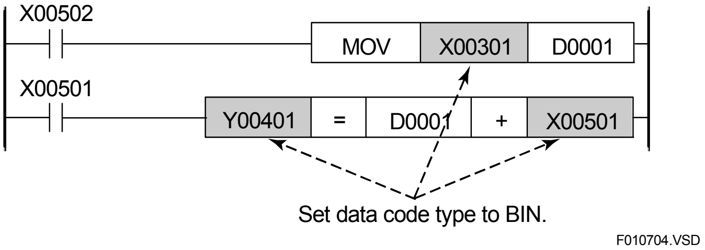


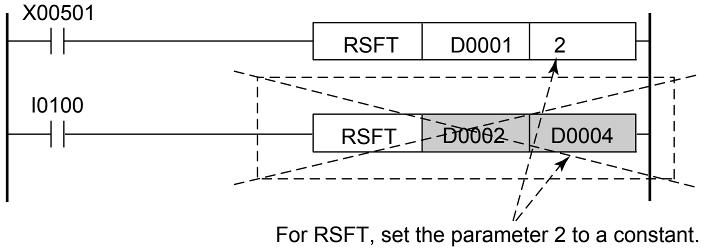
Figure 1.9.7 Example of High-speed Processing of Application Instructions (3)

(2) Using the Following Instructions in Table 1.9.5

- High-speed processing is available only when the conditions in the following table are satisfied.

Table 1.9.6 Device Specification

Classification	FUNC NO.	Instruction	Conditions for Literal Position
Rotate	30/30P	Right rotate	The parameter 1 meets the condition (1) and the parameter 2 is a constant. Example: RROT D1 2 ✓ RROT D1 D2 × LSFT L D1 2 ✓ LSFT L D1 D2 ×
		Right rotate long-word	
	31/31P	Left rotate	
		Left rotate long-word	
Shift	32/32P	Right shift	
		Right shift long-word	
	33/33P	Left shift	
		Left shift long-word	
Special module	81/81P	Read special module	The parameter 3 meets the condition (1) and the other parameters are constants. Example: READ 2 1 D1 1 ✓ READ 2 1 D1 D2 ×
		Read special module in long-word units	
	82/82P	Write special module	The parameter 1 meets the condition (1) and the other parameters are constants. Example: WRITE D1 2 1 1 ✓ WRITE D1 2 1 D2 ×
		Write special module in long-word units	



F010706.VSD

Figure 1.9.8 Example of High-speed Processing of Application Instructions (4)

(3) Using index modification with instructions in Table 1.9.5

- High-speed processing is available only when the conditions in the following table are satisfied. However, only index modification for the register devices (D, B, F, W, R, V, A, U, and Z) are processed.

PAR1 to PAR4 in the table below indicate the parameter positions of ladder instructions. These parameters are PAR1, PAR2, PAR3, and PAR4 from left to right.

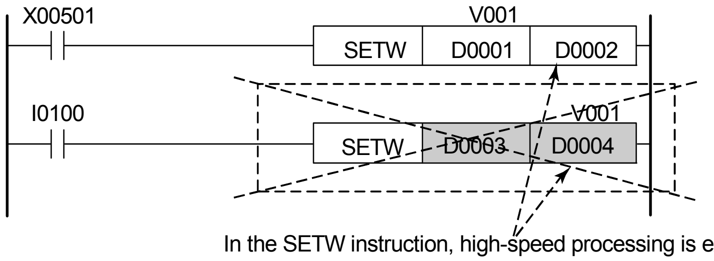
Table 1.9.7 Index Modification Position Specifications (1/2)

Classification	FUNC NO.	Instruction	Mnemonic	Index Modification Positions			
				PAR1	PAR2	PAR3	PAR4
Basic	311	Load specified bit	LDW	✓	×	-	-
		Load specified long-word bit	LDW L	×	×	-	-
	312	Out specified bit	OUTW	✓	×	-	-
		Out specified long-word bit	OUTW L	×	×	-	-
	313	Set specified bit	SETW	✓	×	-	-
		Set specified long-word bit	SETW L	×	×	-	-
	314	Reset specified bit	RSTW	✓	×	-	-
		Reset specified long-word bit	RSTW L	×	×	-	-
Comparison	10	Compare	CMP	✓	✓	-	-
		Compare long-word data	CMP L	×	×	-	-
		Compare double long-word data	CMP D	×	×	-	-
	904	Compare float	FCMP	×	×	-	-
		Compare double-precision float	FCMP E	×	×	-	-
Arithmetic operation	20/20P	Add	CAL	✓	✓	✓	-
		Subtract		✓	✓	✓	-
		Multiply [*1]		✓	✓	✓	-
		Divide [*1]		✓	✓	✓	-
		Add long-word data	CAL L	✓	✓	✓	-
		Subtract long-word data		✓	✓	✓	-
		Multiply long-word data		×	×	×	-
		Divide long-word data		×	×	×	-
		Add double long-word data	CAL D	×	×	×	-
		Subtract double long-word data		×	×	×	-
	903/903P	Add float	FCAL	×	×	×	-
		Subtract float		×	×	×	-
		Multiply float		×	×	×	-
		Divide float		×	×	×	-
		Add double-precision float	FCAL E	×	×	×	-
		Subtract double-precision float		×	×	×	-
		Multiply double-precision float		×	×	×	-
		Divide double-precision float		×	×	×	-
	120/120P	Increment	INC	✓	-	-	-
		Increment long-word data	INC L	✓	-	-	-
	121/121P	Decrement	DEC	✓	-	-	-
		Decrement long-word data	DEC L	✓	-	-	-

*1: If index modification is used for PAR1, high-speed processing is available only when index modification is not used for both PAR2 and PAR3.

Table 1.9.7 Index Modification Position Specifications (2/2)

Classification	FUNC NO.	Instruction	Mnemonic	Index Modification Positions			
				PAR1	PAR2	PAR3	PAR4
Logical operation	20/20P	AND	CAL	✓	✓	✓	-
		OR		✓	✓	✓	-
		XOR		✓	✓	✓	-
		NXOR		✓	✓	✓	-
		AND long-word data	CAL L	✓	✓	✓	-
		OR long-word data		✓	✓	✓	-
		XOR long-word data		✓	✓	✓	-
		NXOR long-word data		✓	✓	✓	-
	21/21P	Two's complement	NEG	✓	-	-	-
		Two's complement long-word data	NEG L	×	-	-	-
	22/22P	Not	NOT	✓	-	-	-
		Not long-word data	NOT L	×	-	-	-
Rotate	30/30P	Right rotate	RROT	✓	×	-	-
		Right rotate long-word data	RROT L	×	×	-	-
	31/31P	Left rotate	LROT	✓	×	-	-
		Left rotate long-word data	LROT L	×	×	-	-
Shift	32/32P	Right shift	RSFT	✓	×	-	-
		Right shift long-word data	RSFT L	×	×	-	-
	33/33P	Left shift	LSFT	✓	×	-	-
		Left shift long-word data	LSFT L	×	×	-	-
Data Processing	52/52P	Binary conversion	BIN	✓	✓	-	-
		Long-word binary conversion	BIN L	×	×	-	-
	53/53P	BCD conversion	BCD	✓	✓	-	-
		Long-word BCD conversion	BCD L	×	×	-	-
	901/901P	Integer to float	ITOF	×	✓	-	-
		Long-word integer to float	ITOF L	×	✓	-	-
	920/920P	Long-word integer to double-precision float	ITOE	×	×	-	-
		Double long-word integer to double-precision float	ITOE D	×	×	-	-
	902/902P	Float to integer	FTOI	×	✓	-	-
		Float to long-word integer	FTOI L	×	✓	-	-
	922/922P	Double-precision float to long-word integer	ETOI L	×	×	-	-
		Double-precision float to double long-word integer	ETOI D	×	×	-	-
	925/925P	Float to double-precision float	FTOE	×	×	-	-
	926/926P	Double-precision float to float	ETOF	×	×	-	-
Data transfer	40/40P	Move	MOV	✓	✓	-	-
		Move long-word data	MOV L	✓	✓	-	-
		Move double long-word data	MOV D	×	×	-	-
	42/42P	Block move	BMOV	×	×	×	-
	43/43P	Block set	BSET	✓	✓	✓	-
Special module	81/81P	Read special module	READ	×	×	✓	×
		Read special module in long-word units	READ L	×	×	✓	×
	82/82P	Write special module	WRITE	✓	×	×	×
		Write special module in long-word units	WRITE L	✓	×	×	×



In the SETW instruction, high-speed processing is executed even when index modification is used for the parameter1.

High-speed processing is not executed when index modification is used for the parameter2.

Figure 1.9.9 Example of High-speed Processing of Application Instructions (5)

1.10 Index Modification and Indirect Specification of Addresses

You can manipulate addresses using either index modification or indirect specification.

1.10.1 Index Modification

Index modification is a technique of addressing a device using an index register (Vnnn) or an index constant to offset (add to or subtract from) a device number specified directly in a basic or application instruction.

■ Using an Index Constant

F3SP22-0S
F3SP28-3S
F3SP38-6S | F3SP53-4S
F3SP58-6S
F3SP59-7S | F3SP66
F3SP67 | F3SP71
F3SP76

If an instruction uses an index constant to address a device, the index constant is added to the device number that is specified directly in the instruction.

An index constant may be any integer between 0 and 2047. It can also be used together with indirect specification.

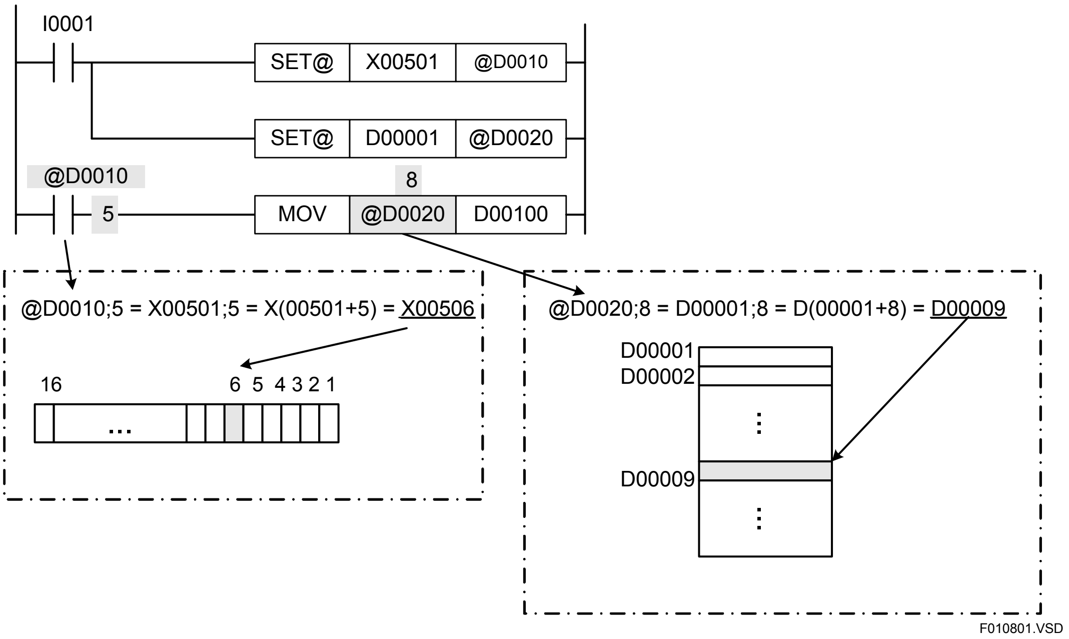
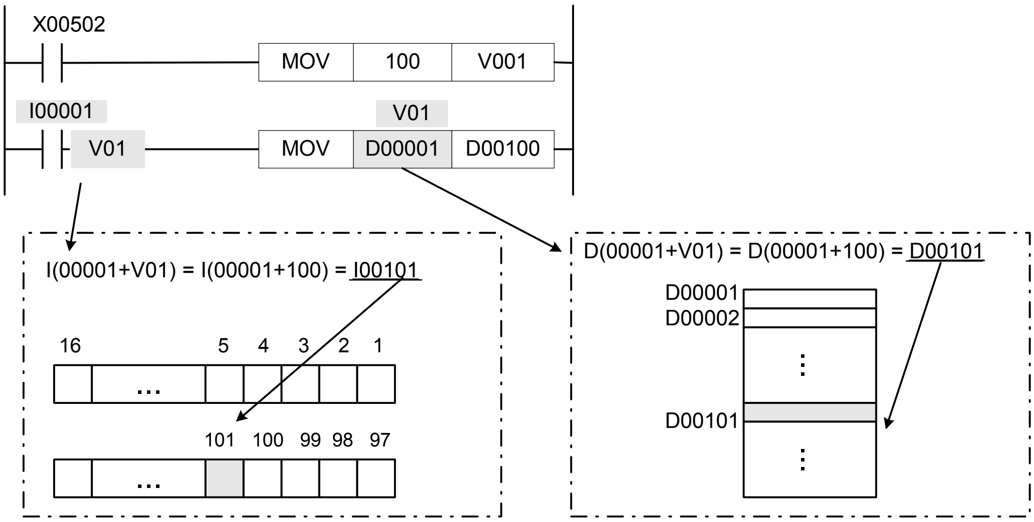


Figure 1.10.1 Index Modification Using an Index Constant

■ When an Index Register Contains a Positive Integer

When the index register contains a positive integer, the integer is added to the device number specified directly in the instruction to determine the device to be processed.

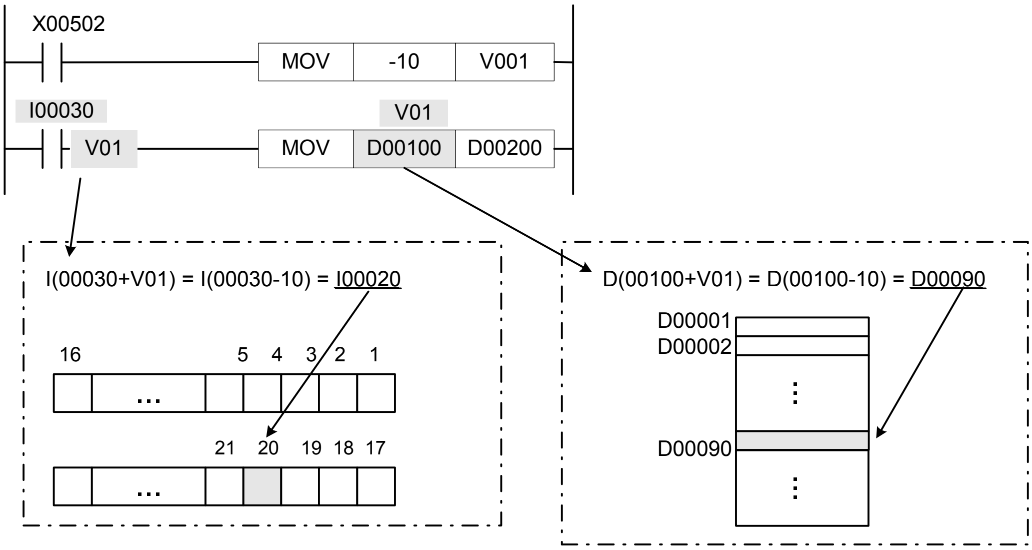


F010802.VSD

Figure 1.10.2 Index Modification (Positive Integer)

■ When an Index Register Contains a Negative Integer

When the index register contains a negative integer, the integer is subtracted from the device number specified directly in the instruction to determine the device to be processed.



F010803.VSD

Figure 1.10.3 Index Modification (Negative Integer)

For an input/output relay, the value of the index register is converted within the slot and index modification is carried out if the input/output relay is found to exist in the same unit. An error is raised if a negative value is specified in the index register that is used to address an input/output relay in the index modification mode.

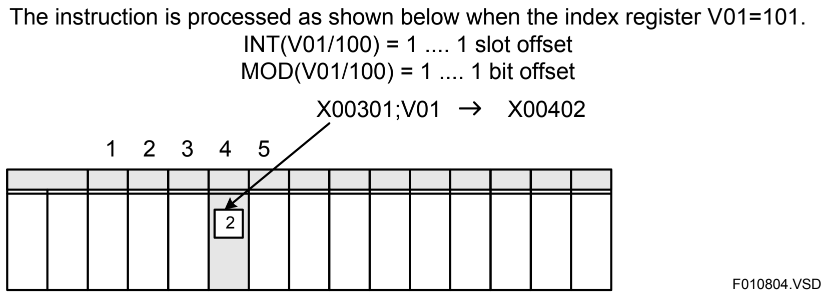

The instruction is processed as shown below when the index register V01=101.
INT(V01/100) = 1 .... 1 slot offset
MOD(V01/100) = 1 .... 1 bit offset
X00301;V01 → X00402
1
2
3
4
5
2
F010804.VSD

Figure 1.10.4 Example of Index Modification on an Input/output Relay X/Y

SEE ALSO

For details on the slots, see Section 1.3.2 of "Sequence CPU Instruction Manual – Functions (for F3SP22-0S, F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S)" (IM 34M06P13-01E), Section A1.3.2 of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E), or Section A1.3.2 of "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

Long-word Index Modification

F3SP71
F3SP76

Long-word index modification reads out data in an index register on a long-word (32-bit) basis to perform index modification. In the long-word index modification, the device is specified adding index register (VnnnL) to the modified device.

Long-word index modification processes points in the index register in 2-point units. If V001L is specified, V001 and V002 areas in the index register are used, and if V003L is specified, V003 and V004 areas are used.

For long-word index modification, odd-numbered devices (V001L, V003L, V005L, and similar) can be specified in the index register. Only devices for which index modification can be used are data registers (D), file registers (B), and cache registers (F).

Table 1.10.1 Devices for which Long-word Index Modification is Available

X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V
								✓	✓	✓				

In the same instruction, both index modification (16 bits) and long-word index modification (32 bits) can be used at the same time.

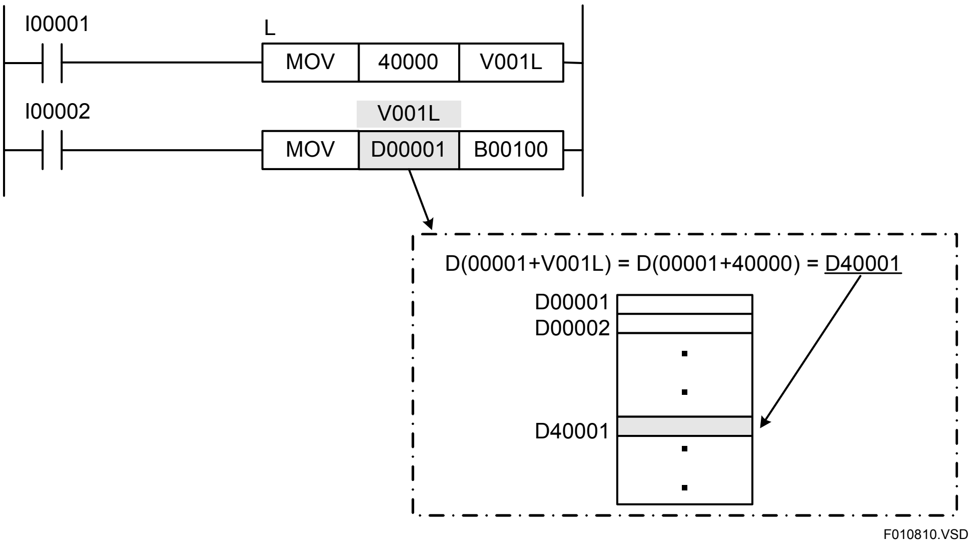
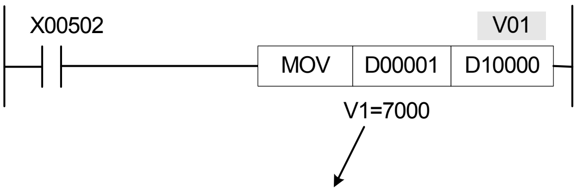


Figure 1.10.5 Long-word Index Modification

CAUTION

- Make sure that the BIN and BCD definitions of addresses after index modification processing are identical to those of the addresses before index modification. The system cannot perform BIN and BCD conversions correctly unless the BIN and BCD definitions are the same.
- The sequence CPU module other than the F3SP71 or F3SP76 makes no check on device numbers addressed in the index modification mode to ensure high-speed execution. Consequently, it signals no instruction error when the device number that results from index modification exceeds the address range of that device. Make sure that any device numbers subject to index modification do not exceed their valid address range. Normal system operation cannot be guaranteed if the address range is exceeded because data other than the specified devices may be altered.
  When using index modification, exercise adequate care with respect to the creation, use, and management of programs and devices.

Example: Consider the example (for the F3SP28) shown below.

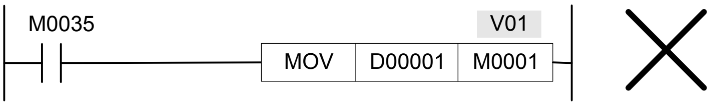


If it is assumed that V1=7000, then D(10000+V01) = D(10000+7000) = D17000.
Normal operation cannot be guaranteed because the resultant address exceeds the value range of the D register (D00001 to D16384).

F010805.VSD

Figure 1.10.6 Device Range Check

- For the F3SP22, F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66, F3SP67, F3SP71, and F3SP76 sequence CPU modules, writing a special relay (M) or special register (Z) with index modification results in an instruction error.



F010806.VSD

Figure 1.10.7 Instruction Error

To write a special relay (M) or special register (Z), address the device directly.

SEE ALSO

For details on the device range check of F3SP71 and F3SP76, see Subsection 1.10.3 "Device Boundary Check."

1.10.2 Indirect Specification

F3SP22-0S F3SP28-3S F3SP38-6S | F3SP53-4S F3SP58-6S F3SP59-7S | F3SP66 F3SP67 | F3SP71 F3SP76

Indirect specification is a technique of addressing a device not directly with its address but indirectly with registers containing its address. A basic or application instruction may indirectly address, or specify, a device by specifying registers that contain its address.

Indirectly specified devices are identified with a prefix '@' in their device number.

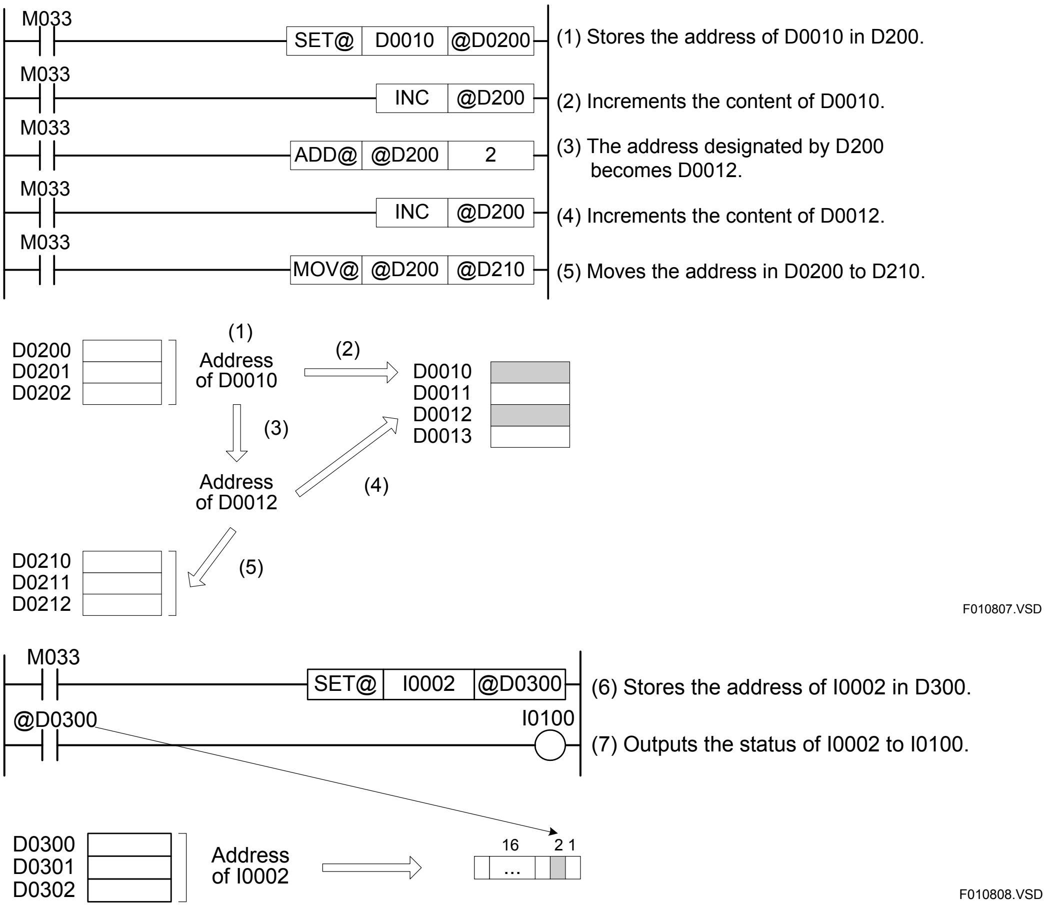
Indirect specification uses three words of registers to store an address.

To store an address in registers for indirect specification (indirect specification registers), use the Indirect Address Set (SET@) instruction.

To manipulate an address stored in indirect specification registers, use the Indirect Address Add (ADD@) instruction.

Specifying “+n” in an Indirect Address Add instruction adds n to the address stored in the indirect specification registers.

To move the content of indirect specification registers, use the Indirect Address Move (MOV@) instruction.



(1) Stores the address of D0010 in D200.

(2) Increments the content of D0010.

(3) The address designated by D200 becomes D0012.

(4) Increments the content of D0012.

(5) Moves the address in D0200 to D210.

F010807.VSD

(6) Stores the address of I0002 in D300.

(7) Outputs the status of I0002 to I0100.

F010808.VSD

Figure 1.10.8 Indirect Specification

Indirect specification may be combined with index modification.

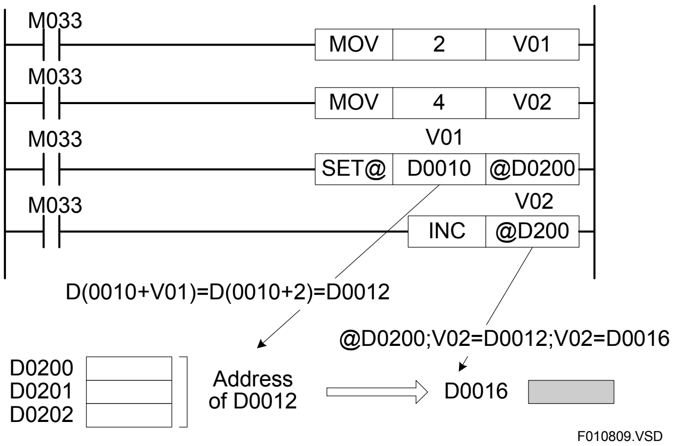


Figure 1.10.9 Indirect Specification Combined with Index Modification

If an Indirect Address Set instruction specifies a timer (T) or a counter (C), the resultant address is the current value of the timer or counter. Time-out relays or end-of-count relays may not be used for indirect specification.

Indirectly specified addresses are valid only in the own CPU. You may not pass and use them in other CPUs through the use of a shared or link register.

CAUTION

The CPU module other than the F3SP71 or F3SP76 does not check whether an indirectly specified address is within the acceptable address range for a device type. You must ensure that the address range is not exceeded. If the address range is exceeded, it may result in modification of unintended devices so proper operation is not guaranteed.

SEE ALSO

For details on the device range check for F3SP71 and F3SP76, see Subsection 1.10.3 "Device Boundary Check."

CAUTION

Devices with indirect specification may only be used in the Indirect Address Set (SET@), Indirect Address Add (ADD@) and Indirect Address Move (MOV@) instructions.

F3SP71
F3SP76

1.10.3 Device Boundary Check

The device boundary check is the functionality that causes a device boundary error (instruction error) if the data is read or written across each device area during index modification, indirect specification, or successive device access when an instruction is executed. This functionality prevents not-specified device areas and system areas from being corrupted at an application level.

You can select whether the device boundary check is enabled (default) or disabled in the CPU configuration.

If the device boundary check is disabled, any access across each device area are not checked. In this case, any index modification may result in reading out or changing data in unintended device areas because accessing such areas other than the specified area will not cause an error.

The device areas are divided into 11 areas altogether. The table below shows the types of areas and the devices that belong to the areas.

Table 1.10.2 Device Area Classification

No.	Area Name	Device	Name
1	Input/Output Relay	Input Relay	X
		Output Relay	Y
2	Relay	Internal Relay	I
		Link Relay	L
		(Extended) Shared Relay	E
		Macro Relay	H
3	Timer/Counter Relay	Time-out Relay	T
		End-of-count Relay	C
4	Special Relay	Special Relay	M
5	Special Register	Special Register	Z
6	Timer/Counter Current Value	Timer Current Value	T
		Counter Current Value	C
7	Register	Data Register	D
		(Extended) Shared Register	R
		Link Register	W
		Macro Register	A
8	File Register	File Register	B
9	Index Register	Index Register	V
		Macro Index Register	U
10	Cache Register	Cache Register	F
11	Others	Pointer Register	P
		Structure Pointer Register	Q

CAUTION

An error occurs only when an instruction accesses across areas described in Table 1.10.2. Any areas with multiple devices in one device area (for example, the timer/counter relay area that has the T and C relays) can be accessed across their devices without an error. Note that your application does not access multiple device areas across their device range.

CAUTION

The CPU modules other than the F3SP71 or F3SP76 do not check whether a specified address is within the acceptable address range for a device type. Note that depending on how addresses are modified by index modification, indirect specification, or successive device accesses, the address range of each device area may be exceeded and other type of device may be addressed. If the address range is exceeded, it may result in data modification of unintended device areas so proper operation is not guaranteed.

SEE ALSO

For details on how to configure the device range check for F3SP71 and F3SP76, see Subsection D3.1.11 "Error Processing Setup" on the "FA-M3 Programming Tool WideField3" (IM 34M06Q16-□□E).

1.11 Differential Type Instructions

Differential type instructions are divided into differential load instructions (LDU and LDD), differential operation instructions (UP and DWN), differential operation instructions using specified device (UPX and DWNX), differential output instructions (DIFU and DIFD), and input differential instructions. Differential type instructions have a preceding cycle execution condition flag that has a value of either ON or OFF. A result signal of a differential type instruction turns on when the preceding and current cycle execution condition flags have different values. Otherwise, the result signal turns off.

The differential operation instructions using a specified device (UPX or DWNX) are used when you want to use a differential type instruction in a FOR-NEXT instruction or save the output of the differential type instructions in the event of a power failure.

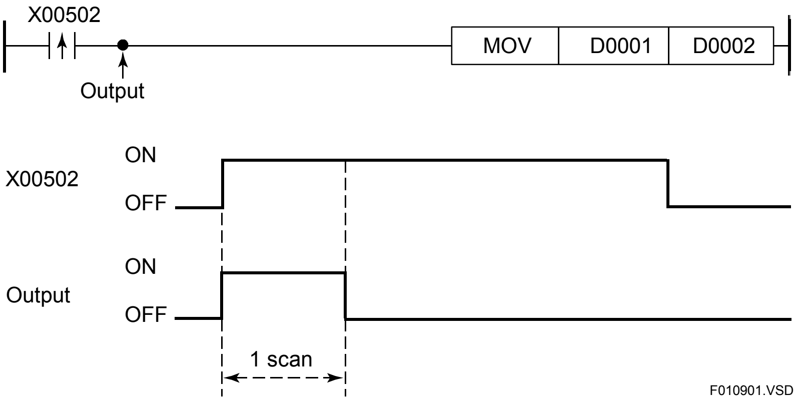


Figure 1.11.1 Differential Load Instruction (LDU)

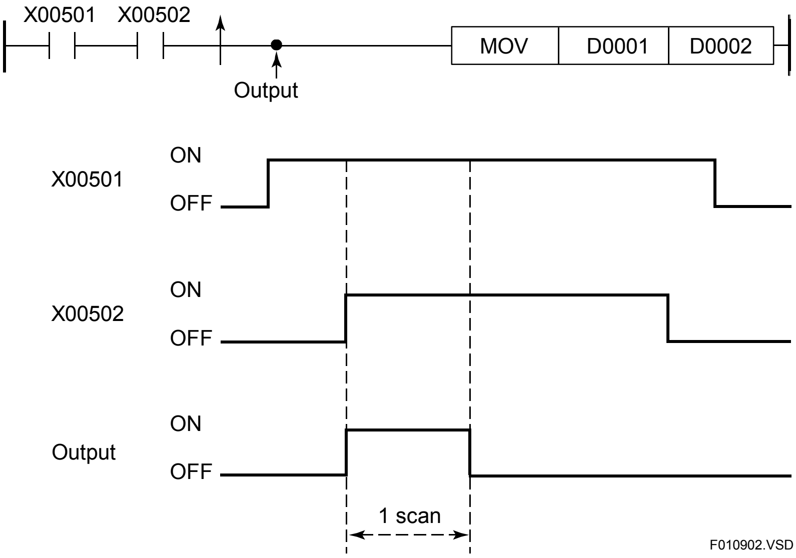
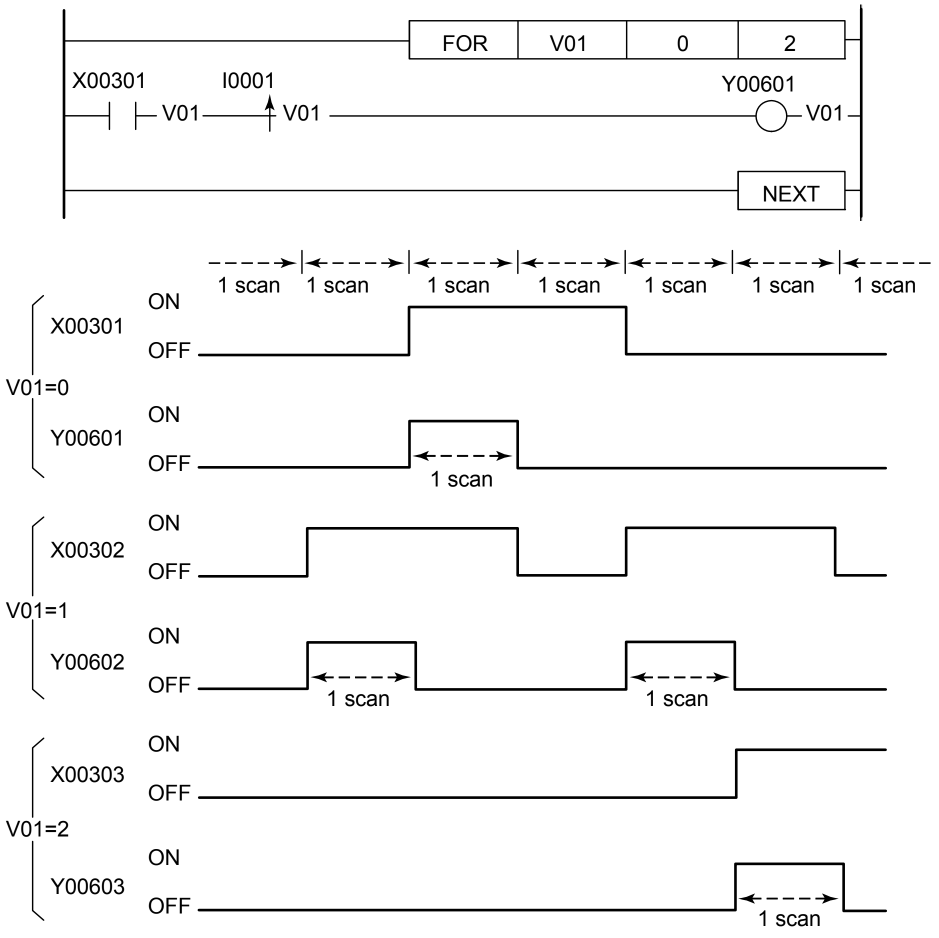


Figure 1.11.2 Differential Operation Instruction (UP)

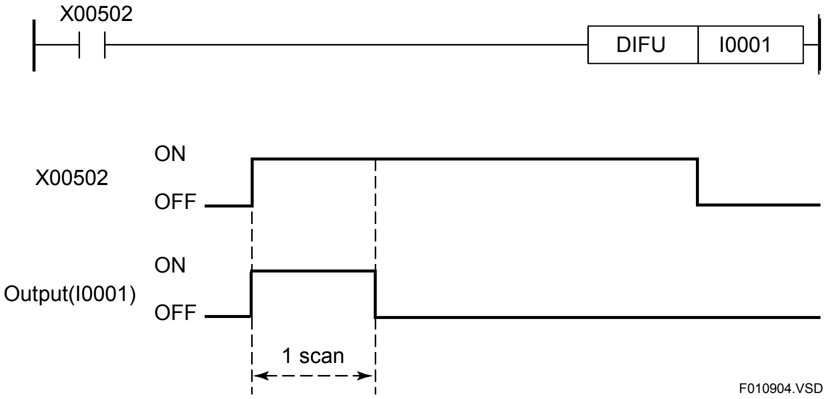
SEE ALSO

For details on data latch at power failure, see Section 3.3.3 of "Sequence CPU – Functions (for F3SP22-0S, F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S)" (IM 34M06P13-01E), Section A3.3.3 of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E), or Section A3.3.3 of "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).



F010903.VSD

Figure 1.11.3 Differential Operation Instruction Using Specified Device (UPX)



F010904.VSD

Figure 1.11.4 Differential Output Instruction (DIFU)

CAUTION

A preceding cycle execution condition flag will be initialized to OFF when the power turns on or operation restarts after changing settings, and any similar flag contained in the circuit that is changed in online edit mode will also be initialized to OFF. To save this type of flag, specify the use of a back-up relay with a differential operation instruction using a specified device.

SEE ALSO

See the individual instruction descriptions for the operation of differential type instructions that are used with or in the IL-ILC instruction, JMP instruction, subroutine program, FOR-NEXT instruction, and interrupt programs.

1.12 Execute-while-ON Instructions and Input Differential Instructions

Application instructions are divided into execute-while-ON instructions and input differential instructions.

■ Execute-while-ON instructions

An execute-while-on application instruction executes every scan while its execution conditions are ON.

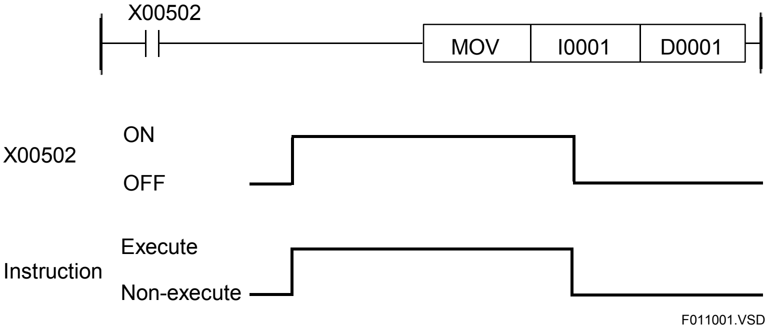


Figure 1.12.1 Executing on Every Scan

■ Input differential instructions

An input differential instruction executes only once when its execution conditions change from OFF to ON state. Since input differential instructions dispense with the need to make an input circuit with a differential instruction to execute only for one scan cycle, they save program coding and shorten scan time.

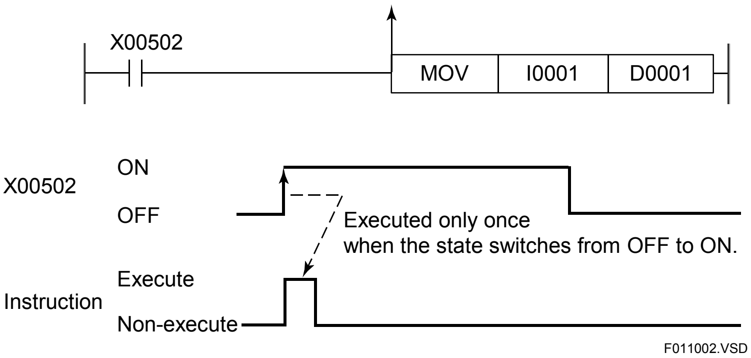


Figure 1.12.2 Example of an Input Differential Instruction

SEE ALSO

For details on the scan and scan time, see Section 3.4 of "Sequence CPU Instruction Manual – Functions (for F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S)" (IM 34M06P13-01E), Section A3.4 of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E), or Section A3.4 of "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

1.13 High-speed READ/WRITE Instructions (HRD/HWR)

The READ/WRITE instructions for accessing special modules are divided into ordinary READ/WRITE instructions and high-speed READ (HRD)/WRITE (HWR) instructions.

The READ/WRITE instructions access special modules while they are being executed. On the other hand, the HRD/HWR instructions refresh the specified special module while executing the program, and the data in CPU memory can be processed while they are being executed.

SEE ALSO

See the individual descriptions on the HRD/HWR instructions for restrictions and other implications.

Table 1.13.1 Devices Available for HRD/HWR Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s1																✓	No	No
n1																✓	No	No
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
k																✓	No	No

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value (cannot be used with the long-word high-speed read instruction)

*3: Counter current value (cannot be used with the long-word high-speed read instruction)

1.14 Number Processing

Since the sign 16-, 32-, or 64-bit BIN data that is handled through application instructions is determined by its most significant bit, the value range of such data is determined as listed below. Similarly, the value range of BCD data is determined as listed below.

-	16-bit data	-32768 to 32767 (BIN)
		0000 to 9999 (BCD)
-	32-bit data	-2147483648 to 2147483647 (BIN)
		00000000 to 99999999 (BCD)
-	64-bit data	-9223372036854775808 to 9223372036854775807 (BIN)

The system actions that the system takes when one of the value ranges of the 16-, 32-, or 64-bit data is exceeded are summarized in the table given below.

Table 1.14.1 System actions taken when the 16-bit data value range is exceeded

Item	BIN Data		BCD Data
Overflow	32765	7FFD	9997
	32766	7FFE	9998
	↓ 32767	↓ 7FFF	↓ 9999
	- 32768	8000	0000
	- 32767	8001	0001
	- 32766	8002	0002
Underflow	- 32766	8002	0002
	- 32767	8001	0001
	↓ - 32768	↓ 8000	↓ 0000
	32767	7FFF	Error[*1]
	32766	7FFE	
	32765	7FFD	

*1: An error is raised if an underflow condition (negative value) occurs in BCD data.

Table 1.14.2 System actions taken when the 32-bit data value range is exceeded

Item	BIN Data		BCD Data
Overflow	2147483645	7FFFFFFD	99999997
	2147483646	7FFFFFFE	99999998
	2147483647	7FFFFFFF	99999999
	- 2147483648	80000000	00000000
	- 2147483647	80000001	00000001
	- 2147483646	80000002	00000002
Underflow	- 2147483646	80000002	00000002
	- 2147483647	↓ 80000001	00000001
	- 2147483648	80000000	00000000
	2147483647	7FFFFFF	Error[*1]
	2147483646	7FFFFFE	
	2147483645	7FFFFFD	

*1: An error is raised if an underflow condition (negative value) occurs in BCD data.

Table 1.14.3 System actions taken when the 64-bit data value range is exceeded

Item	BIN Data		BCD Data
Overflow	9223372036854775805	7FFFFFFFFFFFFFFD	N/A
	↓ 9223372036854775806	↓ 7FFFFFFFFFFFFFFE	
	9223372036854775807	7FFFFFFFFFFFFFFF	
	- 9223372036854775808	8000000000000000	
	- 9223372036854775807	8000000000000001	
	- 9223372036854775806	8000000000000002	
Underflow	- 9223372036854775806	8000000000000002	N/A
	↓ - 9223372036854775807	8000000000000001	
	- 9223372036854775808	8000000000000000	
	9223372036854775807	7FFFFFFFFFFFFFF	
	9223372036854775806	7FFFFFFFFFFFFFE	
	9223372036854775805	7FFFFFFFFFFFFFD	

1.15 Error Processing

When an error occurs during the execution of a basic or application instruction, the error flag (special relay M201) is set to ON and the error instruction number and other information are stored in error instruction number registers (special registers Z022 to Z024). The destination data remains unchanged when an error occurs.

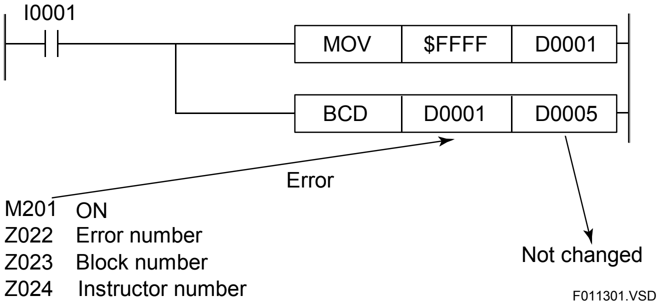


Figure 1.15.1 Error Processing

CAUTION

- See the individual instruction descriptions for instruction-specific errors.
- The user can specify, through configuration, whether the program is to be terminated or not when an error occurs. By default, the program is terminated when an error occurs.

1.16 Automatic Binary ↔ BCD Conversion

When input/output relays (X/Y) are used in a comparison, arithmetic, or move instruction, the system automatically performs binary to BCD conversion, or vice versa, according to the I/O module settings established through the configuration facility. When an input/output relay is defined in BCD, its data is converted from BCD to binary if the relay is an input relay (X) and from binary to BCD if the relay is an output relay. If the data to be handled with external devices is coded in BCD, the programmer can handle it easily without being aware of it during programming.

Binary and BCD definitions must be made in 16-point units through the I/O Module Setup (Data Code Type) of the configuration facility.

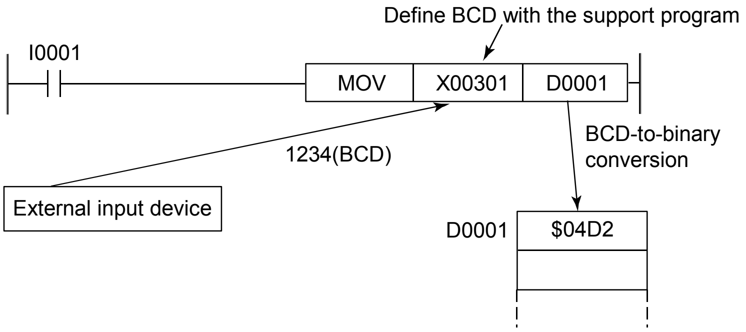


Figure 1.16.1 Example of an input Relay (X)

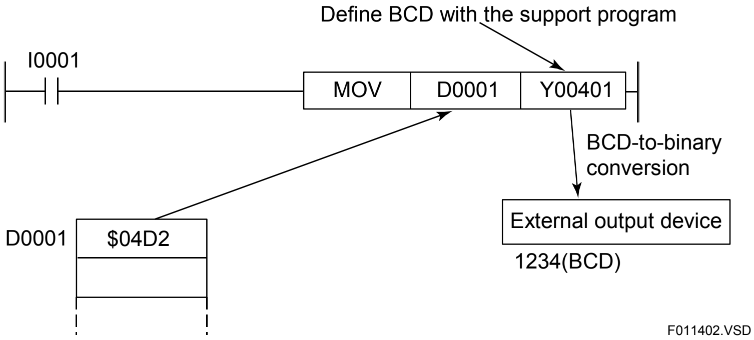


Figure 1.16.2 Example of an Output Relay (Y)

CAUTION

When using input/output relays that are defined in BCD in application instructions, use in 16-point units (Xlmm01, Xlmm17, Xlmm33, …, Ylmm01, Ylmm17, …) for F3SP76 and F3SP71, and use in 4-point units (XImm01, XImm05, XImm09, ..., YImm01, YImm05, ...) for the other sequence CPU modules.

SEE ALSO

For details on configuration, see Section 1.2.3 of "Sequence CPU Instruction Manual – Functions (for F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S)" (IM 34M06P13-01E), Section A9.3 of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E), or Section A9.3 of "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

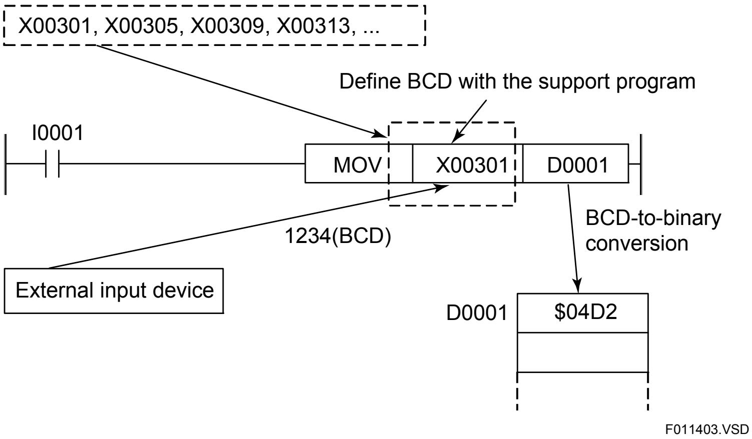


Figure 1.16.3 Example of an Input Relay (X)

Table 1.16.1 Applicable Application Instructions

Classification	FUNC NO.	Instruction	Processing Unit	Mnemonic
Comparison	10	Compare	16 bit	CMP
		Compare long-word data	32 bit	CMP L
Arithmetic operation	20/20P	Add	16 bit	CAL
		Subtract	16 bit	
		Multiply	16 bit	
		Divide	16 bit	
		Add long-word data	32 bit	CAL L
		Subtract long-word data	32 bit	
		Multiply long-word data	32 bit	
		Divide long-word data	32 bit	
	120/120P	Increment	16 bit	INC
		Increment Long-word data	32 bit	INC L
	121/121P	Decrement	16 bit	DEC
		Decrement Long-word data	32 bit	DEC L
Data transfer	40/40P	Move	16 bit	MOV
		Move long-word data	32 bit	MOV L
	41/41P	Partial move	16 bit	PMOV
	42/42P	Block move	n word	BMOV
	43/43P	Block set	n word	BSET
	47/47P	Exchange	16 bit	XCHG
		Exchange long-word data	32 bit	XCHG L

1.17 Devices Available as Instruction Parameters

The table below lists the restrictions that are placed on parameters specified in each instruction.

Table 1.17.1 Devices Available as Instruction Parameters

Device Types	Restrictions
Input Relay (X)	-
Output Relay (Y)	-
Internal Relay (I)	-
Shared Relay (E) Extended Shared Relay (E)	For destination parameters, only writable shared or extended shared relays of the own CPU are available.
Link Relay (L)	For destination parameters, only writable link relays of local station are available.
Special Relay (M)	For destination parameters, only writable special relays are available.
Timer (T)	If the timer current value is used as an instruction parameter, only word-sized parameters are available.
Counter (C)	If the counter current value is used as an instruction parameter, only word-sized parameters are available.
Data Register (D)	-
File Register (B)	File registers cannot be used on F3SP05, F3SP08, and F3SP21.
Cache Register (F)	Cache registers are only available on F3SP71 and F3SP76.
Link Register (W)	For destination parameters, only writable link registers of local station are available.
Special Register (Z)	For destination parameters, only writable special registers are available.
Shared Register (R) Extended Shared Register (R)	For destination parameters, only writable shared or extended shared registers of the own CPU are available.
Index Register (V)	-

Macro-instruction-specific devices

- **The macro relays (H), macro registers (A), and macro index registers (U) are available in instructions that can use the internal relays (I), data registers (D), and index registers (V), respectively, in a macro instruction object (called object).**

The following restrictions apply to F3SP22, F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66, F3SP67, F3SP71, and F3SP76 CPU modules:

- **Index-modified special relays (M) or special registers (Z) cannot be specified as a destination. Otherwise, an instruction error will be raised during execution.**
- **Indirectly-specified special relays (M) or special registers (Z) may not be specified as a destination. Otherwise, an instruction error will be raised during execution.**
- **Block move instructions (BMOV, BSET, SMOV, etc.) and table output instructions (ULOGR, FIFWR, etc.) do not allow special relays (M) or special registers (Z) as the destination. Otherwise, an instruction error will occur during execution.**

1.18 Continuous Type Application Instructions

F3SP66 F3SP71
F3SP67 F3SP76

Execution of many file access and communications instructions cannot be completed within one scan period. To avoid affecting control processing, a processing request is issued at the time of instruction execution but the time-consuming actual processing is carried out in the background. Such instructions are known as "continuous type application instructions."

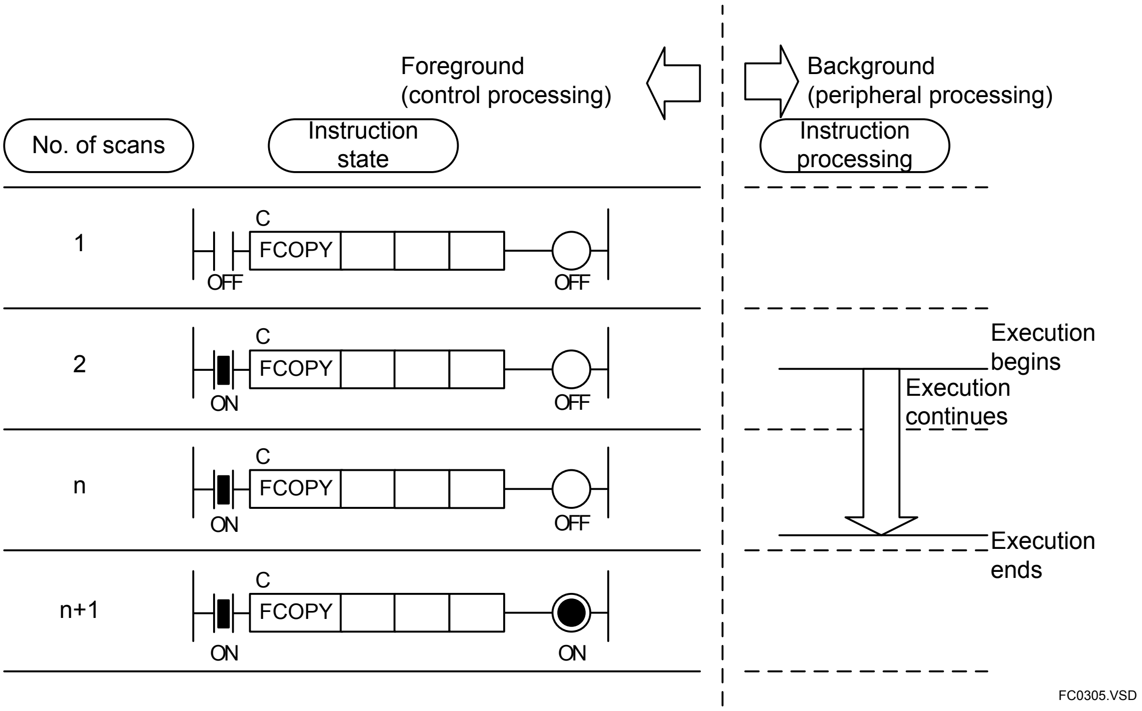


Figure 1.18.1 Concept of Continuous Type Application Instruction

1.18.1 Operation of Continuous Type Application Instructions

This subsection describes the operation of a continuous type application instruction. In the description, the term "input condition" refers to the ON/OFF state of the circuit connection line immediately preceding the continuous type application instruction.

- To execute the instruction:

  Change its input condition from OFF to ON.
- To continue instruction execution:

  Hold its input condition in ON state.
- When instruction execution completes:

  The result signal (on the circuit line connected to the output (right) end of the instruction) is held to ON for one scan period. A user program can check the completion of a continuous type application instruction by monitoring an OUT instruction or some other output-type instruction placed on the output end of the instruction.
- To re-execute the instruction after it has completed execution:

  Turn off and again turn on its input condition. The condition must be held in OFF state for at least 1 scan period.
- To cancel (abort) instruction execution:

  Turn off its input condition during instruction execution. The result signal is held to ON for one scan period. However, the background instruction processing does not end immediately. For more details, see Subsection 1.18.5, "Canceling Execution of Continuous Type Application Instructions."

Table 1.18.1 Operation of Continuous Type Application Instructions

Instruction State of Preceding Scan	Input Condition of Preceding Scan	Input Condition of Current Scan	Transition of Instruction State in Current Scan	Result Signal of Current Scan
Stopped	OFF	ON	Execute	OFF
		OFF	Stopped	OFF
Execute	ON	ON	Continue Execution	OFF
			Execution Completed[*1]	ON for 1 scan
		OFF	Cancelled	ON for 1 scan
Execution Completed[*1]	ON	ON	Execution Completed	OFF
		OFF	Stopped	OFF
Cancelled	OFF	ON	Start execution	OFF
		OFF	Stopped	OFF

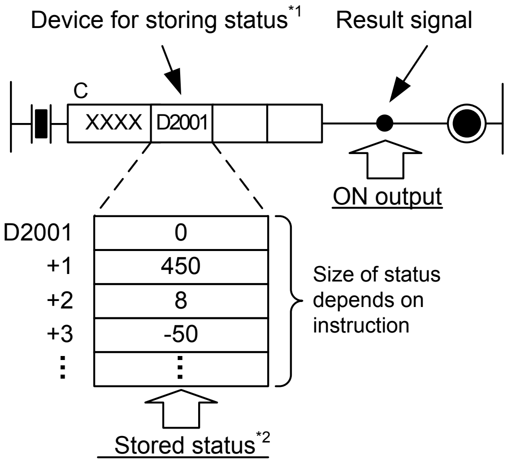
[*1] The transition to 'Execution Completed' state is independent of the input condition, and is triggered by completion of background instruction processing.

1.18.2 Operation Result of Continuous Type Application Instructions

Continuous type application instructions output two types of operation result at the end of instruction execution. A user program determines the completion of instruction execution using the result signal, and checks whether execution is successful using the status.

Table 1.18.2 Operation Result of Continuous Type Application Instructions

Operation Result	Description
Result signal	At the end of instruction execution, the result signal is held to ON for one scan. The result signal is OFF at other times. A user program determines whether instruction execution has completed by checking the ON/OFF state of the result signal.
Status	Regardless of whether instruction execution is successful, a status value is stored in a user-specified device. Some devices may store other return values in addition to the status so the status has a multi-word table structure. If an error status is returned, a user program should perform application error processing such as retry processing.



*1: D2001 is used as an example for illustration purpose.
*2: This is an example of stored status values.

FC0306.VSD

Figure 1.18.2 Operation Result Output of Continuous Type Application Instructions

1.18.3 Error Processing of Continuous Type Application Instructions

If instruction execution ends normally, a zero or positive integer is stored in status. If execution ends in error, a negative integer is stored in status.

A user program should read the execution result status and perform whatever error processing (e.g. retry) as appropriate if an error status is returned.

Even if an error status is returned, the module does not store an error code in a special register, write to the system log (error log), turn on the ALM LED or ERR LED, or switch the program operating mode.

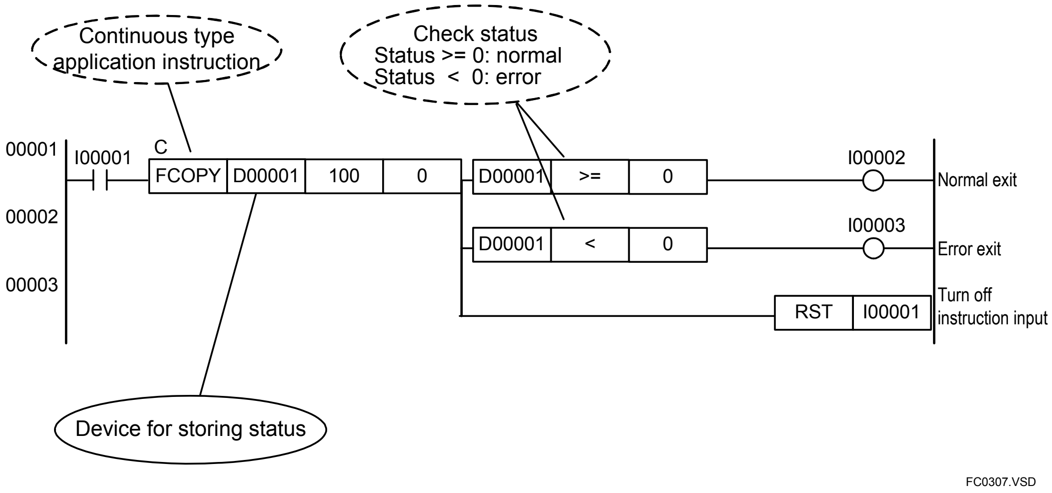


Figure 1.18.3 Error Processing of Continuous Type Application Instructions

1.18.4 Error Status of Continuous Type Application Instructions

The table below shows the error status codes of continuous type application instructions.

Table 1.18.3 Continuous Type Application Instruction Status (timeout-related (-1xxx), non-error-related (-2xxx), exclusive control related (-3xxx))

Category	Continuous Type Application Instruction Status		
	Value	Name	Description
Timeout	-1000	Instruction Timeout	Processing failed to end within the timeout interval specified by an instruction parameter.
	-1001	Internal Communication Timeout	No response was received within the internal communication timeout interval. The following timeout interval can be defined by a user as a CPU property. - FTP Client Network Timeout
Non-error	-2000	End of File Detected	End of file was detected during processing.
	-2001	No Match Found	No match was found.
	-2002	Disconnected by Remote Node	Connection was terminated by the remote node. Check the status of the remote node. This status is also returned if high network load causes data loss.
	-2003	Specified Size/Times Processed	Processing has been completed for the specified data size or iterations. - The size of data received by a TCP/IP Receive Instruction (TCPRCV instruction) reaches the specified receive area size.
	-2004	Block Size Error	Data size is smaller than the specified block size.
Exclusive control	-3001	Redundant Use of Function	A function or resource that disallows redundant use was used redundantly. - Redundant execution of FTP client instruction - Redundant execution of file operation instruction or disk operation instruction - Redundant use of file ID or socket ID
	-3003	Write-prohibit Destination	A write attempt to a destination was unsuccessful because: - the destination was being accessed - the destination is a directory - the destination is read-only
	-3004	Redundant Write Mode	An attempt was made to open a file, which is already open in Write (Append) mode. In Write mode, an attempt was made to open a file which is already opened.
	-3005	Internal Resource Depleted	Internal resource is temporarily depleted. To resolve the problem, retry later. If the problem persists, consider reducing processing load. - FA-M3 internal resource - Protocol stack internal resource

Table 1.18.4 Continuous Type Application Instruction Status (network-related (-5xxx))

Category	Continuous Type Application Instruction Status		
	Value	Name	Description
Network	-5000	Connection Error	Error was detected during connection.
	-5001	Unknown Destination	The destination was not found.
	-5002	Buffer Overflow	Send/receive buffer used by socket instructions has overflowed.
	-5030	FTP User Authentication Failure	Access was denied by FTP server's user authentication process.
	-5031	FTP Password Authentication Failure	Access was denied by FTP server's password authentication process.
	-5032	FTP Command Sequence Error	FTP client processing could not continue because a reply received from the FTP server was out of sequence. This error may be due to repeated cancel operations or bad line quality.
	-5421	FTP Negative Reply 421	FTP server returns a negative reply. The last three digits of this error code (positive value) represent the reply code received from the FTP server.*1
	-5425	FTP Negative Reply 425	
	-5426	FTP Negative Reply 426	
	-5450	FTP Negative Reply 450	
	-5451	FTP Negative Reply 451	
	-5452	FTP Negative Reply 452	
	-5500	FTP Negative Reply 500	
	-5501	FTP Negative Reply 501	
	-5502	FTP Negative Reply 502	
	-5503	FTP Negative Reply 503	
	-5504	FTP Negative Reply 504	
	-5530	FTP Negative Reply 530	
	-5532	FTP Negative Reply 532	
	-5550	FTP Negative Reply 550	
	-5551	FTP Negative Reply 551	
	-5552	FTP Negative Reply 552	
	-5553	FTP Negative Reply 553	

*1: For details on the meaning of each reply code, see the official FTP specification (RFC959). Note that the causes and meanings of reply codes may vary with individual FTP server implementations.

Table 1.18.5 Continuous Type Application Instruction Status (file system related (-6xxx))

Category	Continuous Type Application Instruction Status		
	Value	Name	Description
File system	-6000	Duplicate Filename	Specified destination filename already exists
	-6002	Insufficient Space	There is insufficient space on the storage media. Or, number of files or directories exceeded maximum limit.
	-6004	Memory Card Not Installed	Processing is not allowed because no memory card is installed.
	-6005	Memory Card Not Mounted	Processing is not allowed because no memory card is mounted.
	-6006	Protection Switch is ON	Processing is not allowed because the protection switch is ON.
	-6007	File System Failure	Processing could not continue because a file system failure was detected or the file system is not in FAT16 or FAT32 format. Reformat the disk in proper format, or replace the memory card. This status may be returned occasionally when there is insufficient space on the storage media.
	-6008	Memory Card Failure	Processing could not continue because a memory card failure was detected. Replace the memory card.
	-6009	Unknown Write Error	An error of unknown cause was detected during write processing. Reformat the disk in proper format, or replace the memory card.
	-6010	FLS Processing Sequence Error	Executions of FLSFIRST, FLS and FLSFIN instructions were out of sequence.
	-6011	File Interpretation Error	The NULL byte was detected during interpretation of a text file.

Table 1.18.6 Continuous Type Application Instruction Status (General Instruction (-9xxx))

Category	Continuous Type Application Instruction Status		
	Value	Name	Description
General Instructions	-9000	Cancel Request Issued	A cancel request was issued. Check the resource relay to determine when cancellation is completed.
	-9010	Resource Not Opened	The specified file ID or socket ID is not open. Execute an Open instruction for the ID.
	-9011	Resource Depleted	- No more unused socket ID or file ID is available. Check the resource relay. - An attempt was made to run multiple FTP clients. Concurrent execution of FTP clients is not allowed.
	-9012	Resource Released by External Factor	Processing could not continue because a user has caused the resource relay to be turned off so writing to the resource relay is prohibited. This error may occur if the SD memory card is unmounted when a file is open.
	-9013	Function Not Started	- A function required for processing is not running. - FTP client is not running. Execute an FTPOPEN instruction.
	-9014	Invalid Device Access	An attempt was made to access an invalid device number. Check index modification, indirect specification, data size and status size.
	-9015	Data Processing Error	The requested processing could not continue because of invalid data.
	-9020	Security Error	The specified password or keyword is incorrect.
	-9021	CPU Property ROM Write Error	An attempt to write CPU property data to the internal ROM failed.
	-9999	Internal Error	Internal error was detected.

Table 1.18.7 Continuous Type Application Instruction Status (parameter error related (-1xxxx))

Category	Continuous Type Application Instruction Status		
	Value	Name	Description
Parameter Error	-10xxx	Parameter Error	The specified parameter is invalid. The last 3 digits of the error code indicate the position of the invalid instruction parameter, and its offset from the beginning of the table in words if the parameter is a table. Status : -10 △□□ △ : Parameter number (1 to 3) □□ : Offset in table (00 to 99)
	-12xxx	Invalid Pathname	The specified pathname is invalid. This error is generated if path interpretation failed because the specified file pathname violated a syntax rule. The third digit of the error code indicates the location of the invalid parameter. Status : -12 △□□ △ : 1 to 3 : Text parameter number 4 : CPU property 9 : Unknown type □□ : System reserved (currently 00)
	-13xxx	Pathname Object Not Found	The object designated by the pathname is not found. This error is generated if the specified pathname contains an invalid file or directory. For instance, "\RAMDISK\MYDIR" is specified but there is no directory named "MYDIR" on the RAM disk. This error may also be generated if a wildcard is specified but no match is found. The third digit of the error code indicates the location of the invalid parameter. Status : -13 △□□ △ : 1 to 3 : Text parameter number 4 : CPU property 9 : Unknown type □□ : System reserved (currently 00)
	-15xxx	Invalid String Length	The string length parameter is invalid. This error is generated if the string length exceeds the maximum limit, or if NULL is specified for a parameter that does not allow a NULL value. The third digit of the error code indicates the location of the invalid parameter. Status : -15 △□□ △ : 1 to 3 : Text parameter number 4 : CPU property 9 : Unknown type □□ : System reserved (currently 00)

1.18.5 Canceling Execution of Continuous Type Application Instructions

Execution of a continuous type application instruction can be cancelled by turning off its input condition during execution. When a falling edge is detected in the input condition, the result signal is immediately held to ON to notify termination of execution, and a Cancel Request Issued status code (-9000) is stored in the instruction status.

However, note that despite notification of instruction termination, background instruction processing is not yet terminated. Instead, a cancellation request is issued to background processing, and a few seconds may be required to complete the termination.

If the same continuous type application instruction is executed before background processing cancellation is completed, resource competition occurs and an exclusive control related error will be generated. To avoid this, you should include the resource relay in the input condition of a continuous type application instruction.

TIP

Just as with instruction cancellation, in the event of an instruction timeout (error code -1000), background instruction processing continues to run for a short while. Therefore, it is also necessary in this case to incorporate exclusive control in the program using resource relays.

SEE ALSO

For details on resource relays, see Subsection 1.18.6, "Resource Relays."

CAUTION

When the input of a continuous type application instruction is turned off, the instruction immediately returns a Cancel Request Issued status and terminates execution. However, actual background processing such as background communications is not terminated immediately. To check for termination of actual processing, check that the associated resource relay is turned off.

1.18.6 Resource Relays

Resource relays are special relays for preventing competition between continuous type application instructions. A resource relay indicates the status of a resource, which is subject to exclusive control. Resources include file IDs socket IDs, functions and instructions.

By inserting a resource relay in the input condition of a continuous type application instruction, you can prevent errors due to resource competition. In particular, resource relays are required for checking for completion of cancellation processing or instruction timeout processing in user applications where cancellation request for a continuous type application instruction, or timeout (-1000) may occur.

Table 1.18.8 Resource Relays (related to file system instructions)

Category	Continuous Type Application Instruction Resource Relays		
No.	Name	Function	Description
M1026	No Unused File ID	No unused file ID is available.	Turns on when all file IDs are in use.
M1025	File/Disk Operation Group Busy	File operation instruction group or disk operation instruction group is running.	Turns on during execution of any file operation instruction or disk operation instruction such as an FCOPY or DISKINFO instruction. Execution of any other file operation instruction or disk operation instruction is not allowed while this relay is ON. This relay is not affected by file access instructions. This is a read-only relay. Do not write to it.
M1041 to M1056	File ID Open	File ID is open.	Each file ID is associated with one special relay. The relay for a file ID turns on while the file ID is open. When the relay for a file ID is OFF, no instruction using the file ID can be executed. This is a read-only relay. Do not write to it.
M1057 to M1072	File ID Busy	File ID is busy.	Each file ID is associated with one special relay. The relay for a file ID turns on during execution of any file system instruction using the file ID. When the relay for a file ID is ON, no other file system instruction using the same file ID can be executed. This is a read-only relay. Do not write to it.

Table 1.18.9 Special Relays (related to socket instructions)

Category	Continuous Type Application Instruction Resource Relays		
No.	Name	Function	Description
M1028	No Unused UDP Socket	No unused UDP socket is available.	Turns on when all UDP/IP sockets are in use.
M1029	No Unused TCP Socket	No unused TCP socket is available.	Turns on when all TCP/IP sockets are in use.
M1105 to M1120	Socket Open	Socket is open.	Each socket ID is associated with one special relay. The relay for a socket ID turns on while the socket ID is open. When the relay for a socket ID is OFF, the socket ID cannot be used. This is a read-only relay. Do not write to it.
M1121 to M1136	Socket Busy	Socket is busy.	Each socket ID is associated with one special relay. The relay for a socket ID turns on during execution of any socket instruction using the socket ID. When the relay for a socket ID is ON, no other socket communication instruction using the same socket ID can be executed except for concurrent execution of sending and receiving. This is a read-only relay. Do not write to it.
M1073 to M1088	Socket Sending	Socket is performing send processing.	Each socket ID is associated with one special relay. The relay for a socket ID turns on during send processing of the socket. When the relay for a socket ID is ON, no send request is allowed for the same socket ID. This is a read-only relay. Do not write to it.
M1089 to M1104	Socket Receiving	Socket is performing receive processing.	Each socket ID is associated with one special relay. The relay for a socket ID turns on during receive processing of the socket. When the relay for a socket ID is ON, no receive request is allowed for the same socket ID. This is a read-only relay. Do not write to it.

Table 1.18.10 Resource Relays (related to FTP client instructions)

Category	Continuous Type Application Instruction Resource Relays		
No.	Name	Function	Description
M1027	FTP Client Busy	An FTP client instruction is being executed.	This relay turns on during execution of any FTP client instruction. When the relay is ON, no other FTP client instruction can be executed. By inserting this relay in the input condition of an FTP client instruction, you can prevent inadvertent redundant execution. This is a read-only relay. Do not write to it.

1.18.7 Precautions When Executing Continuous Type Application Instructions

By nature, continuous type application instructions require multiple scans to complete processing, and thus should not be executed only once but must be executed repeatedly until execution completes.

The table below shows the precautions when executing continuous type application instructions from different program types.

Table 1.18.11 Precautions when Executing Continuous Type Application Instructions

Program Type	Precaution
Ladder block (execute-all-blocks mode)	None
Ladder block (execute-specified-blocks mode)	Executing an Inactivate Block (INACT) instruction during execution of a continuous type application instruction forces cancellation of instruction processing.
Sensor control block	Continue execution of a sensor control block until the execution of a continuous type application instruction ends. If you stop a sensor control block before instruction execution ends, instruction processing cannot be completed.
I/O interrupt routine	Use of continuous type application instructions in I/O interrupt routines is not allowed.
Subroutine	Repeat a subroutine call until the execution of a continuous type application instruction ends. If you stop subroutine call before instruction execution ends, instruction processing cannot be completed.
Macro and input macro	Repeat a macro call until the execution of a continuous type application instruction ends. If you stop macro call before execution of continuous type application instruction ends, instruction processing cannot be completed. Calling a macro containing an executing continuous type application instruction from a different location in the program is not allowed.

CAUTION

- A continuous type application instruction will not execute correctly if it is executed in only one scan.
- Do not execute the same continuous type application instruction more than once within the same scan using macros. Repeat execution using FOR-NEXT instruction or JMP instruction is also disallowed.

1.18.8 Restrictions for Inserting Continuous Type Application Instructions

There are some restrictions for inserting continuous type application instructions in a ladder diagram. Placing a continuous type application instruction in an invalid location generates a program syntax error in WideField3.

The figure below illustrates some locations where continuous type application instructions cannot be inserted.

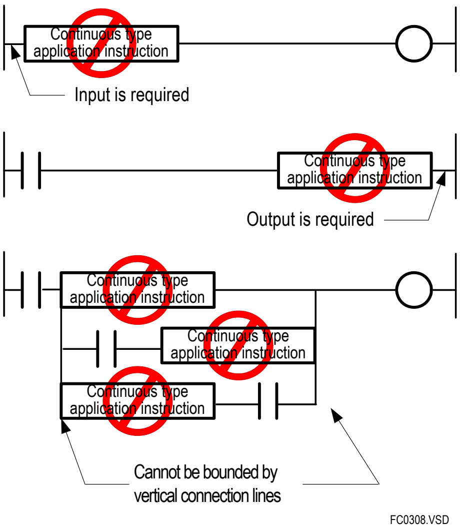

Continuous type application instruction
Input is required
Continuous type application instruction
Output is required
Continuous type application instruction
Continuous type application instruction
Continuous type application instruction
Cannot be bounded by vertical connection lines
FC0308.VSD

Figure 1.18.4 Restrictions for Inserting Continuous Type Application Instructions

1.18.9 Online Edit of Continuous Type Application Instructions

Do not online edit a circuit containing an executing continuous type application instruction. If an executing continuous type application instruction is edited online, instruction processing will be forcedly terminated and re-executed using the modified parameters (including text parameters). In this case, the result signal of the continuous type application instruction will not be held to ON for 1 scan to indicate end of instruction processing.

Even if parameter values are not modified during online edit, a Redundant Use of Function error (status code -3001) may still be generated during re-execution depending on the status of the resource.

CAUTION

Before performing online edit of a circuit containing continuous type application instructions, check to ensure that no continuous type application instruction is running.

1.19 Text Parameter

F3SP66 F3SP71
F3SP67 F3SP76

Some file system instructions use text parameters as instruction parameters. A text parameter value can be stored using the Text Parameter (TPARA) instruction. The Text Parameter (TPARA) instruction must be executed before an instruction that requires text parameters.

1.19.1 Text Parameter (TPARA)

This instruction is used to specify a text parameter required by some continuous type application instructions.

Table 1.19.1 Text Parameter

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Application Instruction	–	Text Parameter	TPARA	┤ TPARA │ │ │ │ ├	✓	–	5	8 bit	–

■ Parameter

Text Parameter ┤ TPARA │ n │ s1 │ s2 │ s3 ├

n : Text parameter number (W) (1-3)

s1 : Device storing character string 1 (W)
(Up to 255 characters, terminated by a NULL character)

s2 : Device storing character string 2 (W)
(Up to 255 characters, terminated by a NULL character)

s3 : Device storing character string 3 (W)
(Up to 255 characters, terminated by a NULL character)

■ Available Devices

Table 1.19.2 Devices Available for the Text Parameter Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con stant	Index Modification	Indirect Specification, Pointer P
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
s1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
s2									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
s3									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

This instruction is used to specify text parameters required by some continuous type application instructions. You should specify the text parameter number according to the text parameter number of the instruction requiring the text parameter.

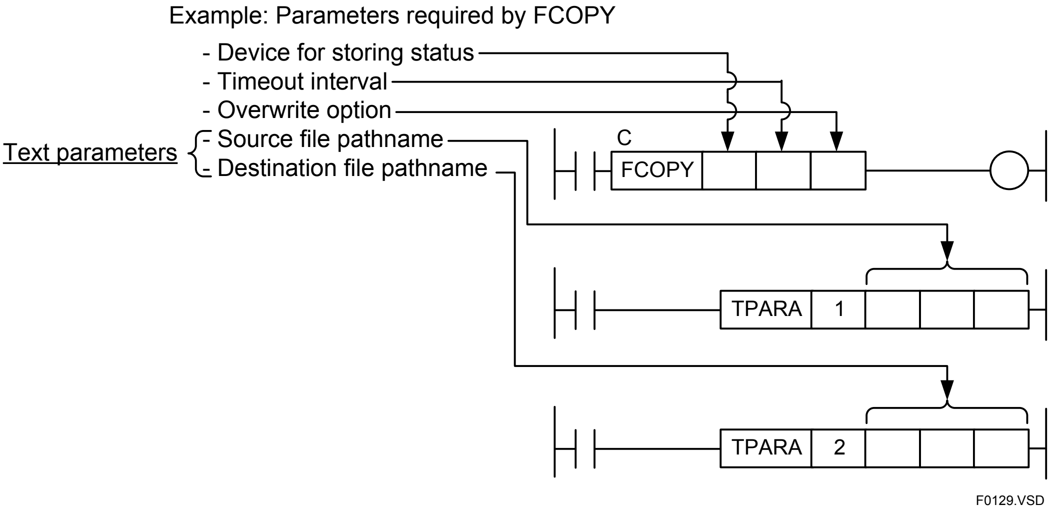


Figure 1.19.1 Text Parameter Number (TPARA instruction must be executed before continuous type application instruction)

The Text Parameter instruction must be executed to set up a text parameter before an instruction requiring the text parameter. Text parameters are stored in the system text parameter area. An instruction requiring a text parameter reads the text parameter from the text parameter area when it begins execution (at the rising edge of the input).

One text parameter area is provided for all continuous type application instructions executing in the normal scan, and another area is provided for all continuous type application instructions executing in the sensor control block. Therefore, normal blocks and the sensor control block do not compete for the text parameter area but continuous type application instructions sharing each area do compete. You should store text parameter value before each instruction execution to ensure proper execution.

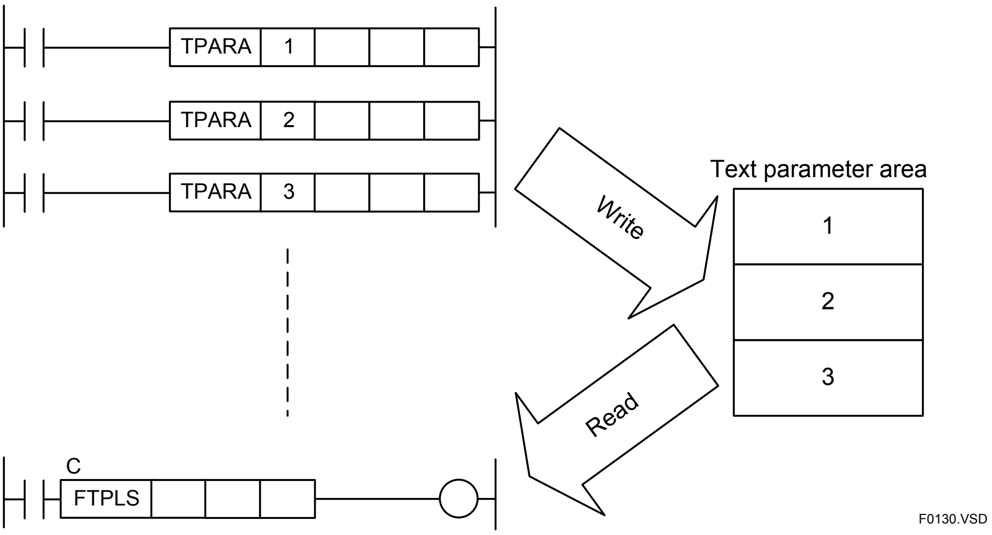


Figure 1.19.2 Text Parameter Area

This instruction performs character string concatenation. It concatenates strings A to C (see next figure) into one text parameter n (n=1 to 3). This string concatenation feature enables user programs to define smaller string units and increase reuse of defined string constants.

Specify NULL for all non-required strings A to C. Using a zero constant value in an instruction parameter is equivalent to specifying a NULL value.

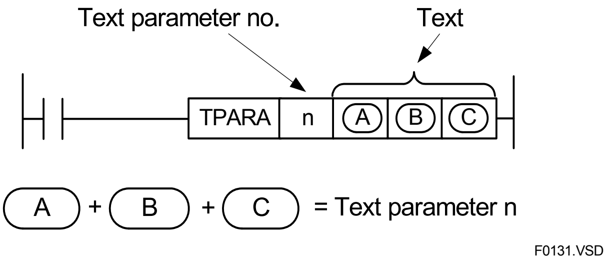


Figure 1.19.3 One Text Parameter

Each text parameter can contain up to 255 characters. The individual lengths and combined length of strings A to C must not exceed 255 characters.

TIP

Using string concatenation, you can specify as text parameter various string combinations such as string A, string B, string C, string (A+C), string (A+B), etc. You can also specify strings only for position A and C, without specifying a string for position B. In this case, the unused position B must be specified as NULL.

■ Programming Example

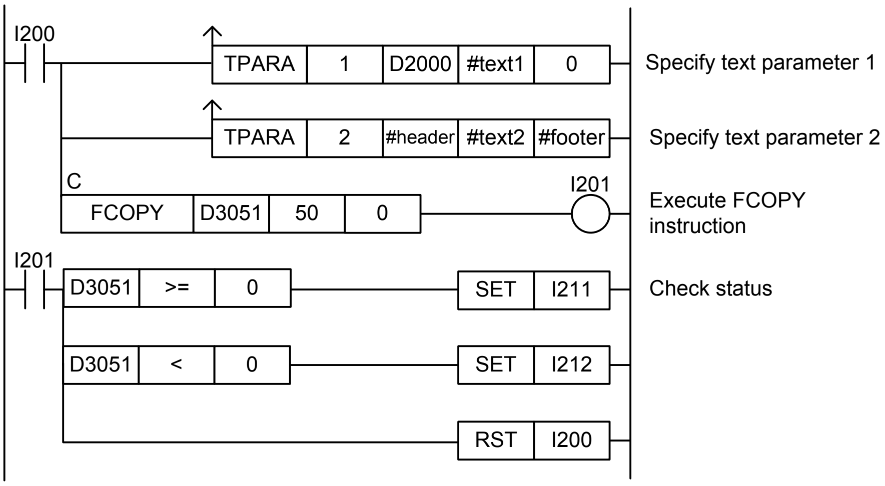


Figure 1.19.4 Example of a Text Parameter Program

This sample code sets up text parameter 1 and text parameter 2, which are to be passed to a Copy File instruction (FCOPY).

The string stored in devices starting with D2000, and the string defined by constant name #text1 are concatenated to become text parameter 1. The three strings defined by constant names #header, #text2 and #footer are concatenated to become text parameter 2.

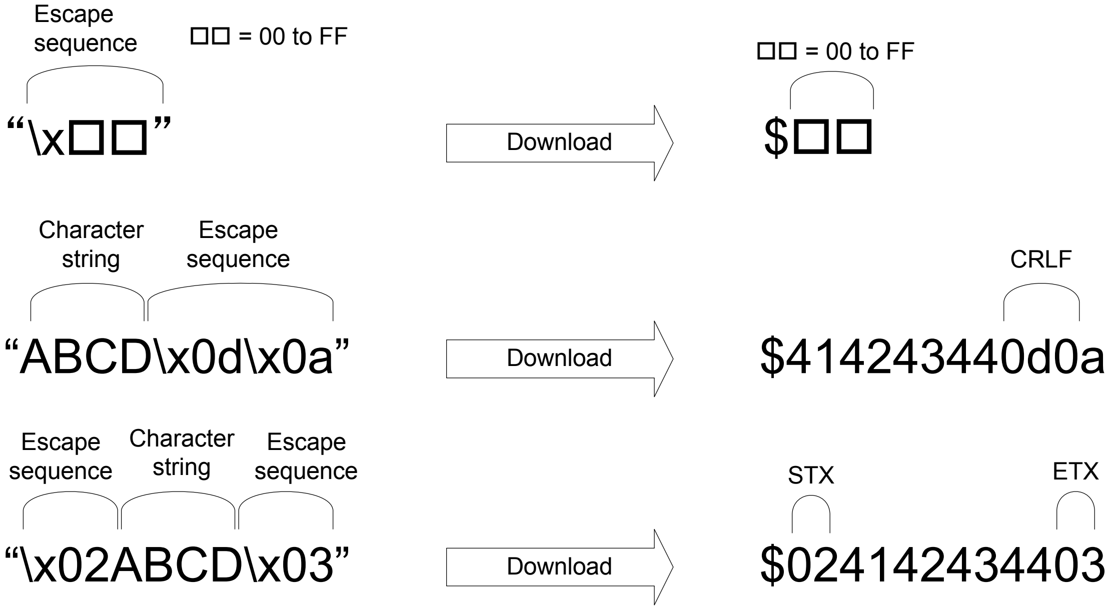
1.20 M3 Escape Sequence

F3SP66 F3SP71
F3SP67 F3SP76

This section describes the M3 escape sequence function.

■ Merits of M3 Escape Sequence

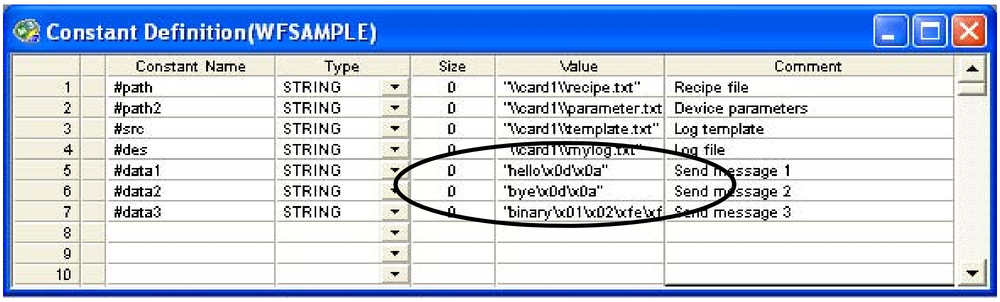
An escape sequence is a binary representation of a character string. When characters coded in a defined format (escape sequence) is included within a character string, WideField3 replaces the escape sequence with its binary data before downloading.



FA0654.VSD

Figure 1.20.1 Downloading Escape Sequence (Converting Escape Sequence to Binary Data)

Conversely, when reading a character string containing binary data, WideField3 replaces the binary data with an escape sequence character string before display.



		Constant Name	Type	Size	Value	Comment
1		#path	STRING	0	"\\card1\\recipe.txt"	Recipe file
2		#path2	STRING	0	"\\card1\\parameter.txt	Device parameters
3		#src	STRING	0	"\\card1\\template.txt"	Log template
4		#des	STRING	0	"\\card1\\mylog.txt"	Log file
5		#data1	STRING	0	"hello\x0d\x0a"	Send message 1
6		#data2	STRING	0	"bye\x0d\x0a"	Send message 2
7		#data3	STRING	0	"binary\x01\x02\xfe\xf	Send message 3
8						
9						
10						

FA0655.VSD

Figure 1.20.2 Replacing Binary Data by Character String

The following are some benefits of using M3 escape sequence.

- M3 escape sequence can be used to define control characters (ETX, STX, etc.) along with transmission text when creating a telegram. In the past, when the M3 escape sequence function was not available, transmission text and control characters had to be created separately and combined using application instructions.
- M3 escape sequence can be used to easily combine text with tab characters and line feed characters (CRLF, LF, etc.) when creating a CSV formatted file.

■ M3 Escape Sequence Specifications

The specifications of M3 escape sequence is described here.

● List of M3 escape sequences

The table below lists M3 escape sequences.

Table 1.20.1 List of M3 Escape Sequences

M3 Escape Sequence	Corresponding Binary Value (in hexadecimal)
\x00 – \x0F	$00 – $0F
\x10 – \x1F	$10 – $1F
\x20 – \x2F	$20 – $2F
\x30 – \x3F	$30 – $3F
\x40 – \x4F	$40 – $4F
\x50 – \x5F	$50 – $5F
\x60 – \x6F	$60 – $6F
\x70 – \x7F	$70 – $7F
\x80 – \x8F	$80 – $8F
\x90 – \x9F	$90 – $9F
\xA0 – \xAF	$A0 – $AF
\xB0 – \xBF	$B0 – $BF
\xC0 – \xCF	$C0 – $CF
\xD0 – \xDF	$D0 – $DF
\xE0 – \xEF	$E0 – $EF
\xF0 – \xFF	$F0 – $FF

● Range of values that can be represented using M3 escape sequence

M3 escape sequence can be used to represent hexadecimal values from $00 to $FF.

● Syntax of M3 escape sequence

Specify a hexadecimal value between "00" and "FF" prefixed by the "\x" character string.

As an example, specify "\xD0" for $D0.

M3 escape sequences are case-insensitive.

● Representing backslash (\) character

To represent a backslash (\) character, code it as two backslash characters (\\) (the first backslash character acts as the escape character).

Scope of M3 escape sequence

The table below lists the applicable scope of M3 escape sequence.

Table 1.20.2 Applicable Scope of M3 Escape Sequence

Category	Function	Operation
Input/edit	Constant definition	Defining and editing character string constants
	Block/macro edit	Entering or editing character string constants in instruction parameters
Display	Constant definition	Display of character string constants
	Block/macro edit	Display of character string constants in TIP help.
	Circuit monitor	Display of character string constants in TIP help.

Special relays and special registers

There are no special relays and special registers related to the M3 escape sequence function.

M3 Escape Sequence Setup

The M3 escape sequence function requires no setup before use.

Using M3 Escape Sequence

Entering and editing M3 escape sequence

- Entering Escape Sequence in Constant Definition

  Select [Project]–[Constant Definition] from the menu bar of WideField3 to open the constant definition window. M3 escape sequence can be entered directly into the constant definition window when defining a character string constant.

- Entering Escape Sequence in Block/Macro Edit

  Open a block or macro. M3 escape sequence can be entered directly as a character string constant for an instruction parameter.

Display of M3 escape sequence

- Display of M3 Escape Sequence in Constant Definition

  Select [Project]–[Constant Definition] from the menu bar of WideField3 to open the constant definition window. If binary data is included in a defined character string constant, it is displayed as an escape sequence.

- Display of M3 Escape Sequence in Block/Macro Edit Window and Circuit Monitor

  Moving the mouse cursor over a constant name displays its defined value as TIP help. If binary data is included in a defined character string constant, it is displayed as an escape sequence.

2. Basic Instructions

This chapter describes the basic instructions for the FA-M3 CPU modules with sample programs. Be sure to read this chapter before writing programs.

2.1 Basic Instructions

Chapter 2 explains how to use the basic instructions for the FA-M3 CPU modules. The notational conventions for the basic instruction descriptions are summarized below.

TIP

Basic instructions refer to a group of instructions that behave like electrical circuit components such as relays and coils. Except for some instructions, such as timer (TIM) instruction, they operate on a single bit.

■ Quick Function Reference Chart

Each basic instruction description begins with a quick function reference chart, which looks like what is shown below.

Table 2.1.1 How to Interpret the Basic Instruction Quick Reference Chart

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Basic Instruction	01	Set	SET	—[SET]	✓	—	↑	2	1 bit	—
	02	Reset	RST	—[RST]	✓	—	↑	2	1 bit	—
(1)	(2)	(3)	(4)	(5)		(6)	(7)	(8)	(9)	(10)

T020101.VSD

(1) Classification

Identifies the type of the instruction. The instructions described in this chapter are all basic instructions.

(2) FUNC No.

Indicates the function number of the instruction. A hyphen ("-") in this column indicates that the instruction has no function number assigned. An instruction that is identified by a function number followed by a letter P is a differential type instruction which is executed only once when its input is turned on.

(3) Instruction

Indicates the name of the instruction.

(4) Mnemonic

Contains the mnemonic of the instruction, which can be used in WideField3, WideField2, WideField, and Ladder Diagram Support Program M3.

(5) Symbol

Represents the graphical representation of the instruction in WideField3, WideField2, WideField or Ladder Diagram Support Program M3.

(6) Input Condition Required?

Indicates whether a contact needs to be specified as the input condition. A check mark in the "Yes" column indicates that a contact must always be specified as the input condition. A check mark in the "No" column indicates that no contact must be specified as the input condition.

(7) Execution Condition

Contains the execution condition for an instruction that requires an input condition.

Table 2.1.2 Execution Condition Symbols

Symbol	Description
⎍	Represents an execute-while-ON instruction. The instruction continues to execute while the previous condition is ON. Execution of the instruction is suppressed if the previous condition is OFF.
↑	Represents an execute-at-ON instruction. The instruction is executed only once when the state of its precondition switches from OFF to ON. Subsequently, the instruction is not executed even when its precondition is ON.
↓	Represents an execute-at-OFF instruction. The instruction is executed only once when the state of its precondition switches from ON to OFF. Subsequently, the instruction is not executed even when its precondition is OFF.
—	Represents an always-execute instruction. The instruction is executed regardless of whether its precondition is ON or OFF.

(8) Step Count

Indicates the number of steps required to execute the instruction. The step count varies with the execution condition and the presence or absence of index modification.

(9) Processing Unit

Indicates the processing unit of the instruction. Instructions whose processing unit is 1 are intended for relays. Instructions whose processing unit is 16 or 32 bits are intended for registers. 16 or 32 bits of relays, when combined, may be handled as data.

(10) Carry

When an instruction identified by a check mark in the Input Condition column is executed, the state of the special relay (M188) may be changed to represent the carry state. See the individual instruction descriptions.

TIP

The "Input Condition Required?" column indicates whether an input condition instruction such as Load (LD) or Compare (CMP) must be specified when an instruction is used.

TIP

The "Execution Condition" column indicates what operation result of the input condition (such as Load (LD) instruction) will trigger the execution of an instruction. There are four execution conditions corresponding to different operation results of the input condition.

■ Parameter

The Parameter section shows the parameters of WideField3, WideField2, WideField or Ladder Diagram Support Program M3. A single letter in the symbol column has the following meanings:

s: Identifies the source.

s1: Identifies the first source of two or more sources.

s2: Identifies the second source of two or more sources.

d: Identifies the destination.

d1: Identifies the first destination of two or more destinations.

d2: Identifies the second destination of two or more destinations.

Note: Source : Data before the operation is performed
Destination : Data after the operation is performed

Devices that may be specified as both source and destination are indicated in the "Available Devices" table in the individual the instruction description.

■ Available Devices

Check marks in the available device table indicate that the corresponding device is available. For instructions with two or more parameters, available devices are indicated for each of the parameters.

■ Function

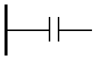
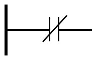
Describes the function of the instruction.

■ Programming Example

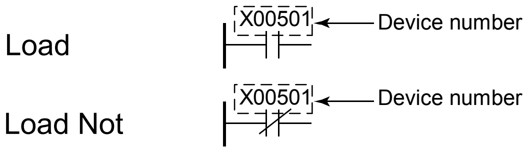
Shows sample codes which contain the instruction.

2.2 Load (LD), Load Not (LDN)

Table 2.2.1 Load, Load Not

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Processing Unit	Carry
					Yes	No		Without Index Modification	With Index Modification		
Basic Instruction	–	Load	LD		–	✓	–	1	2	1 bit	–
		Load Not	LDN		–	✓	–	1	2	1 bit	–

■ Parameter



F020201.VSD

■ Available Devices

Table 2.2.2 Devices Available for the Load and Load Not Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
Device	✓	✓	✓	✓	✓	✓	✓*1	✓*2									Yes	Yes

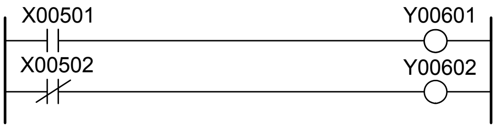
*1: Time-out relay

*2: End-of-count relay

■ Function

The LD instruction starts a logical operation (contact a) and the LDN instruction starts a logical NOT operation (contact b). They take in the ON/OFF information about a specified device as the execution result.

■ Programming Example



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	OUT	Y00601				
0003	LDN	X00502				
0004	OUT	Y00602				

F020202.VSD

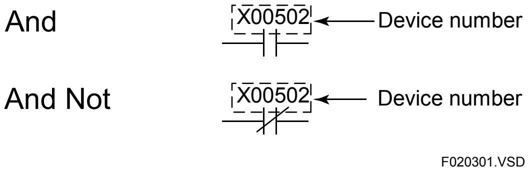
Figure 2.2.1 Sample Code for LD and LDN

2.3 And (AND), And Not (ANDN)

Table 2.3.1 And, And Not

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Pro-cessing Unit	Carry
					Yes	No		Without Index Modifi-cation	With Index Modifi-cation		
Basic Instruc-tion	–	And	AND	─┤├─	✓	–	–	1	2	1 bit	–
		And Not	ANDN	─┤/├─	✓	–	–	1	2	1 bit	–

■ Parameter



■ Available Devices

Table 2.3.2 Devices Available for the And and And Not Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
Device	✓	✓	✓	✓	✓	✓	✓[*1]	✓[*2]									Yes	Yes

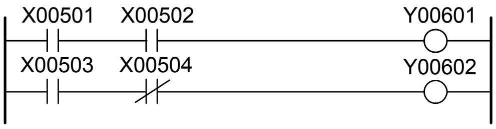
*1: Time-out relay

*2: End-of-count relay

■ Function

The AND instruction starts a logical AND operation (serially connected contact a) and the ANDN instruction starts a logical NAND operation (serially connected contact b). They take in the ON/OFF information about a specified device and performs an AND on it with the current execution result. The result of AND becomes the execution result.

■ Programming Example



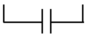
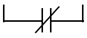
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	AND	X00502				
0003	OUT	Y00601				
0004	LD	X00503				
0005	ANDN	X00504				
0006	OUT	Y00602				

F020302.VSD

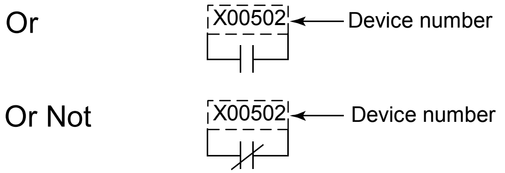
Figure 2.3.1 Sample Code for AND and ANDN

2.4 Or (OR), Or Not (ORN)

Table 2.4.1 Or, Or Not

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Processing Unit	Carry
					Yes	No		Without Index Modification	With Index Modification		
Basic Instruction	–	Or	OR		✓	–	–	1	2	1 bit	–
		Or Not	ORN		✓	–	–	1	2	1 bit	–

■ Parameter



F020401.VSD

■ Available Devices

Table 2.4.2 Devices Available for the Or and Or Not Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
Device	✓	✓	✓	✓	✓	✓	✓*1	✓*2									Yes	Yes

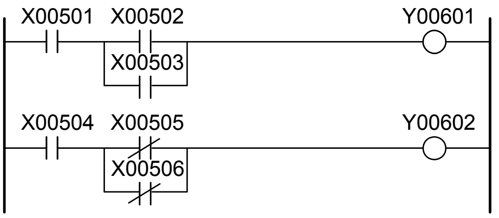
*1: Time-out relay

*2: End-of-count relay

■ Function

The OR instruction starts a logical OR operation (parallelly connected contact a) and the ORN instruction starts a logical NOR operation (serially connected contact b). They take in the ON/OFF information about a specified device and perform an OR on it with the current execution result. The result of OR becomes the execution result.

■ Programming Example



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	LD	X00502				
0003	OR	X00503				
0004	ANDLD					
0005	OUT	Y00601				
0006	LD	X00504				
0007	LDN	X00505				
0008	ORN	X00506				
0009	ANDLD					
0010	OUT	Y00602				

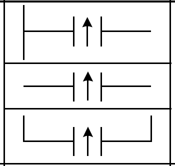
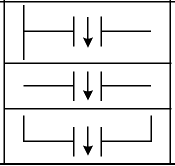
F020402.VSD

Figure 2.4.1 Sample Code for OR and ORN

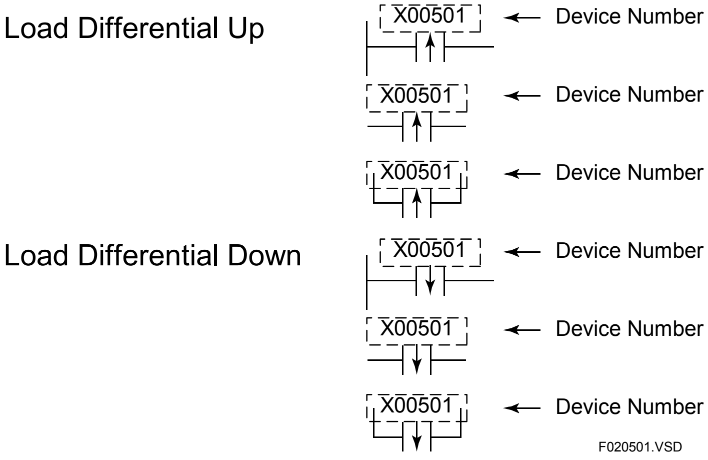
2.5 Load Differential Up (LDU), Load Differential Down (LDD)

F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 2.5.1 Load Differential Up [LDU], Load Differential Down [LDD]

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Processing Unit	Carry
					Yes	No		Without Index Modification	With Index Modification		
Basic Instruction	301	Load Differential Up	LDU		—	✓	—	2	3	1 bit	—
	302	Load Differential Down	LDD		—	✓	—	2	3	1 bit	—

■ Parameter



■ Available Devices

Table 2.5.2 Devices Available for Load Differential Up and Load Differential Down

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
Device	✓	✓	✓	✓	✓	✓	✓*1	✓*2									Yes	Yes

*1: Time-out relay

*2: End-of-count relay

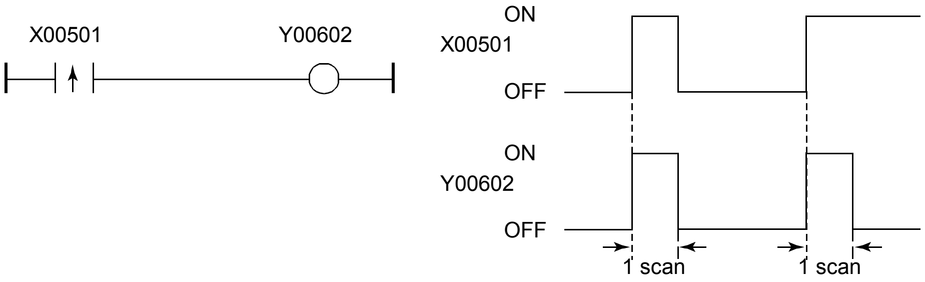
■ Function

(1) Load Differential Up

The LDU instruction sets and holds a result signal of logical operation to ON for one scan on the rising edge of the signal of a specified device. The result signal is held to OFF except on the rising edge of the specified device.

Table 2.5.3 Result of Load Differential Up Operation

Device	Operation Result
ON →ON	OFF
ON →OFF	OFF
OFF →OFF	OFF
OFF →ON	ON



F020502.VSD

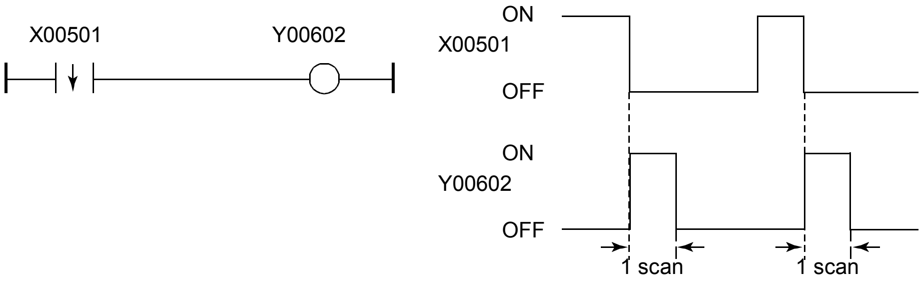
Figure 2.5.1 Timing of Load Differential Up Operation

(2) Load Differential Down

The LDD instruction sets and holds a result signal of logical operation to ON for one scan on the falling edge of the signal of a specified device. The result signal is held to OFF except on the falling edge of the specified device.

Table 2.5.4 Result of Load Differential Down Operation

Device	Operation Result
ON →ON	OFF
ON →OFF	ON
OFF →OFF	OFF
OFF →ON	OFF

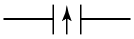
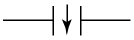
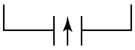
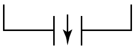


F020503.VSD

Figure 2.5.2 Timing of Load Differential Down Operation

When you want to express logical AND/OR of Load Differential Up/Down instruction and another circuit element (see the "Symbol" column in the following table), use LDU/LDD instruction in combination with ANDLD/ORLD instruction as shown in the following table. For details, see programming example below.

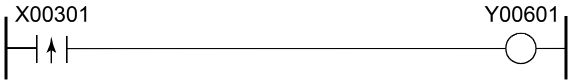
Table 2.5.5 AND/OR Notation Using Load Differential Up/Down

AND/OR Notation	Symbol	Mnemonic
AND		LDU ANDLD
		LDD ANDLD
OR		LDU ORLD
		LDD ORLD

■ Programming Example

(1) Load Differential Up

The program shown below sets Y00601 to ON when X00301 switches from OFF to ON.

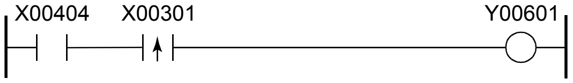


F020504.VSD

Line No.	Instruction	Operands				
0001	LDU	X00301				
0002	OUT	Y00601				

Figure 2.5.3 Load Differential Up 1

The program shown below sets Y00601 to ON when X00404 turns ON and X00301 switches from OFF to ON.

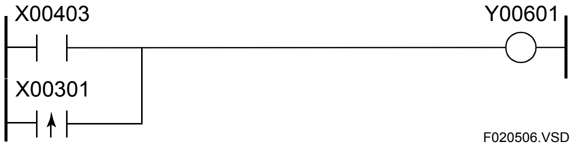


F020505.VSD

Line No.	Instruction	Operands				
0001	LD	X00404				
0002	LDU	X00301				
0003	ANDLD					
0004	OUT	Y00601				

Figure 2.5.4 Load Differential Up 2

The program shown below sets Y00601 to ON when X00403 turns ON or X00301 switches from OFF to ON.

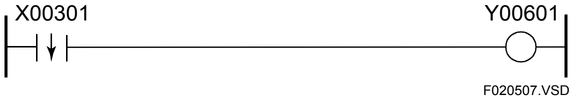


Line No.	Instruction	Operands				
0001	LD	X00403				
0002	LDU	X00301				
0003	ORLD					
0004	OUT	Y00601				

Figure 2.5.5 Load Differential Up 3

(2) Load Differential Down

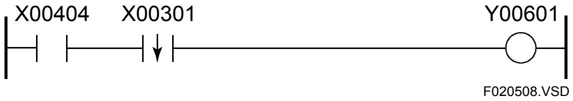
The program shown below sets Y00601 to ON when X00301 switches from ON to OFF.



Line No.	Instruction	Operands				
0001	LDD	X00301				
0002	OUT	Y00601				

Figure 2.5.6 Load Differential Down 1

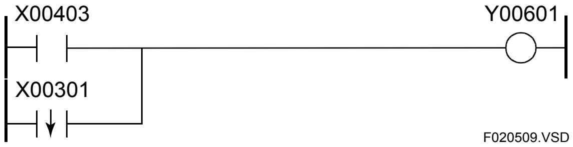
The program shown below sets Y00601 to ON when X00404 turns ON and X00301 switches from ON to OFF.



Line No.	Instruction	Operands				
0001	LD	X00404				
0002	LDD	X00301				
0003	ANDLD					
0004	OUT	Y00601				

Figure 2.5.7 Load Differential Down 2

The program shown below sets Y00601 to ON when X00403 turns ON or X00301 switches from ON to OFF.

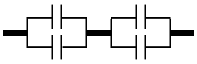
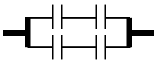


Line No.	Instruction	Operands				
0001	LD	X00403				
0002	LDD	X00301				
0003	ORLD					
0004	OUT	Y00601				

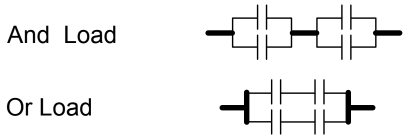
Figure 2.5.8 Load Differential Down 3

2.6 And Load (ANDLD), Or Load (ORLD)

Table 2.6.1 And Load, Or Load

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Basic Instruc-tion	–	And Load	ANDLD		✓	–	–	1	1 bit	–
		Or load	ORLD		✓	–	–	1	1 bit	–

■ Parameter



Note:
Neither ANDLD nor ORLD instructions are in bold in an actual ladder sequence program (circuit).

F020601.VSD

■ Function

● And Load

The ANDLD instruction performs a logical AND operation on circuit elements and passes its execution result to the next processing.

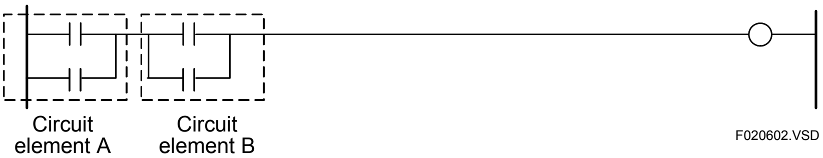


Figure 2.6.1 And Load Instruction

● Or Load

The ORLD instruction performs a logical OR operation on circuit elements and passes its execution result to the next processing.

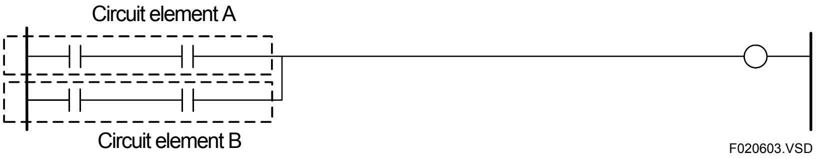


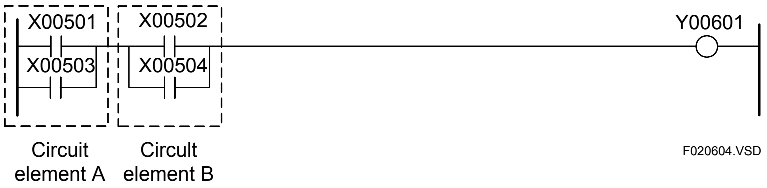
Figure 2.6.2 Or Load Instruction

TIP

Since either instruction is generated automatically within the program at a logical AND or OR between circuit elements of WideField3, WideField2, WideField, or Ladder Diagram Support Program M3 (See above two figures), you need not enter these instructions in your ladder program. Their mnemonics, however, need to be specified as instructions as they are not generated automatically within a program. These instructions cannot be monitored using the device monitor function of WideField3, WideField2, WideField, or Ladder Diagram Support Program M3.

■ Programming Example

● And Load

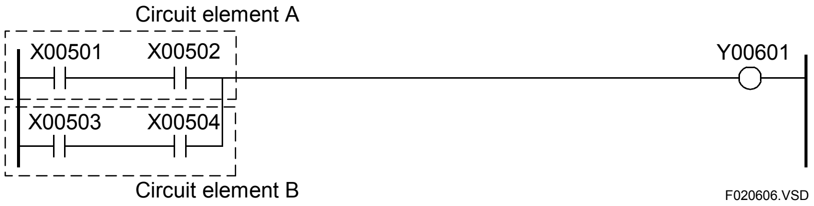


Line No.	Instruction	Operands					
0001	LD	X00501					Operation on circuit element A
0002	OR	X00503					
0003	LD	X00502					Operation on circuit element B
0004	OR	X00504					
0005	ANDLD						AND of A and B
0006	OUT	Y00601					Resultant output

F020605.VSD

Figure 2.6.3 Sample Code for And Load

● Or Load

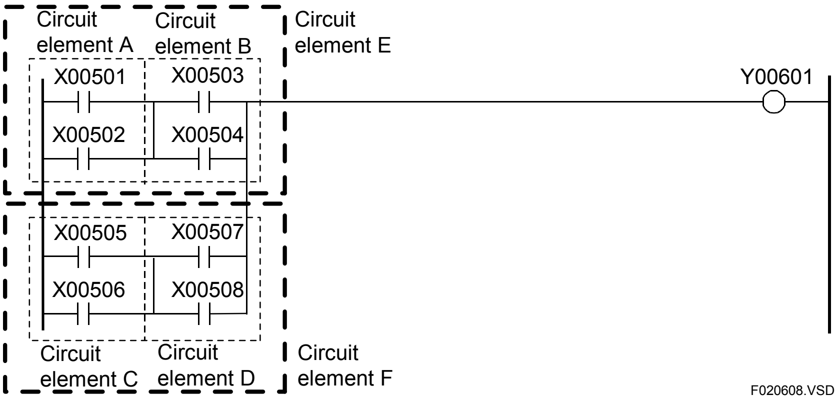


Line No.	Instruction	Operands					
0001	LD	X00501					Operation on circuit element A
0002	AND	X00502					
0003	LD	X00503					Operation on circuit element B
0004	AND	X00504					
0005	ORLD						OR of A and B
0006	OUT	Y00601					Resultant output

F020607.VSD

Figure 2.6.4 Sample Code for Or Load

● And Load and Or Load



F020608.VSD

Line No.	Instruction	Operands					
0001	LD	X00501					Operation on circuit element A
0002	OR	X00502					
0003	LD	X00503					Operation on circuit element B
0004	OR	X00504					
0005	ANDLD						AND of A and B
0006	LD	X00505					Operation on circuit element C
0007	OR	X00506					
0008	LD	X00507					Operation on circuit element D
0009	OR	X00508					
0010	ANDLD						AND of C and D
0011	ORLD						OR of E and F
0012	OUT	Y00601					Resultant output

F020609.VSD

Figure 2.6.5 Sample Code for ANDLD and ORLD

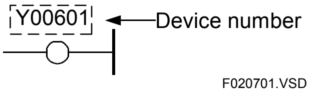
2.7 Out (OUT)

Table 2.7.1 Out

Classi-fication	FUNC No.	Instruc-tion	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Pro-cessing Unit	Carry
					Yes	No		Without Index Modifi-cation	With Index Modifi-cation		
Basic Instruc-tion	–	Out	OUT	─○─┤	✓	–	–	1	2	1 bit	–

■ Parameter

Out



F020701.VSD

■ Available Devices

Table 2.7.2 Devices Available for Out instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
Device		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3									Yes	Yes

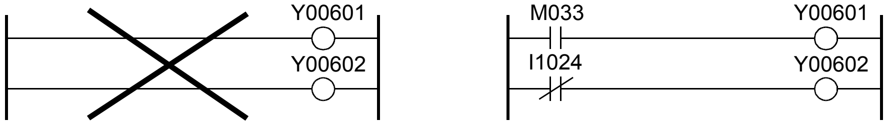
*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Time-out relay

*3: End-of-count relay

■ Function

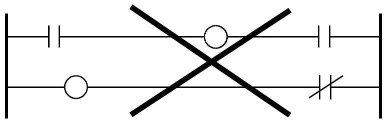
The OUT instruction outputs the result of the logical operations performed so far to the specified coil (device). You cannot output data directly from the power rail to a coil. If there is a need to output to a coil directly regardless of the ON/OFF state of contacts, use a special relay M033 (always-ON contact) or contact b of an unused internally relay as a dummy contact.



F020702.VSD

Figure 2.7.1 Dummy Contacts for OUT

You cannot insert a contact after an OUT.



F020703.VSD

Figure 2.7.2 OUT Disallowed

When two or more OUTs are used for the same coil (device), only the last OUT takes effect and the results of the previous OUTs are ignored.

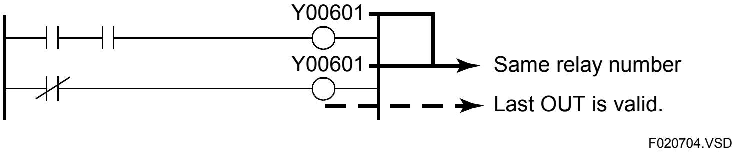


Figure 2.7.3 Using OUTs for the Same Relay

OUTs may be used in parallel.

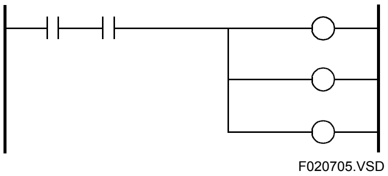


Figure 2.7.4 OUTs Used in Parallel

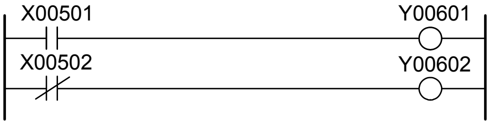
SEE ALSO

For details on the number of the OUT instructions that can be used in parallel, see "FA-M3 Programming Tool WideField3" (IM 34M06Q16-□□E).

TIP

The power rail is the leftmost vertical line in a ladder diagram.

■ Programming Example



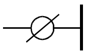
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	OUT	Y00601				
0003	LDN	X00502				
0004	OUT	Y00602				

F020706.VSD

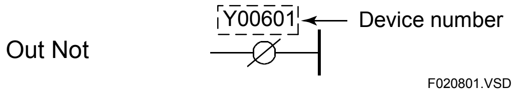
Figure 2.7.5 Sample Code for OUT

2.8 Out Not (OUTN)

Table 2.8.1 Out Not

Classi-fication	FUNC No.	Instruc-tion	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Pro-cessing Unit	Carry
					Yes	No		Without Index Modifi-cation	With Index Modifi-cation		
Basic Instruc-tion	7	Out Not	OUTN		✓	–	–	1	2	1 bit	–

■ Parameter



■ Available Devices

Table 2.8.2 Devices Available for the Out Not Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
Device		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3									Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Time-out relay

*3: End-of-count relay

■ Function

The OUTN instruction inverts the result of the logical operations performed so far and outputs it to the specified coil (device). You cannot output data directly from the power rail to a coil.

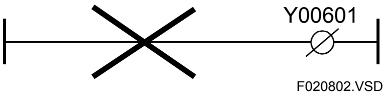


Figure 2.8.1 Direct Output from the Power Rail Disallowed

You cannot insert a contact after an OUTN.

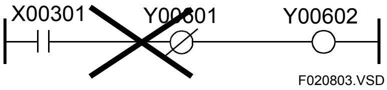


Figure 2.8.2 OUTN Disallowed

OUTNs may be used in parallel.

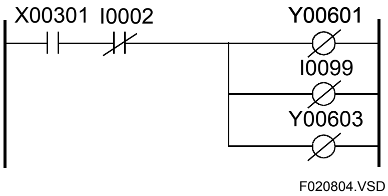
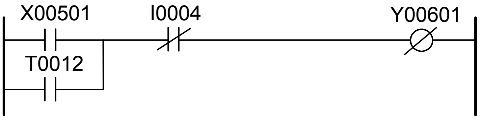


Figure 2.8.3 OUTNs Used in Parallel

SEE ALSO

For details on the number of the OUTN instructions that can be used in parallel, see "FA-M3 Programming Tool WideField3" (IM 34M06Q16-☐☐E).

■ Programming Example



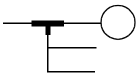
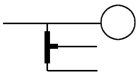
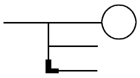
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	OR	T0012				
0003	ANDN	I0004				
0004	OUTN	Y00601				

F020805.VSD

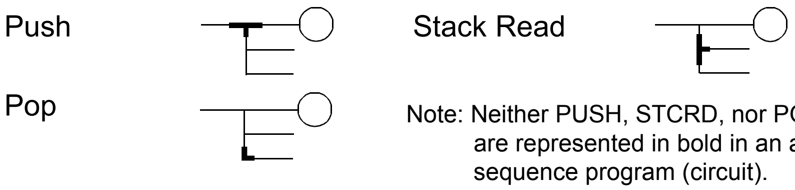
Figure 2.8.4   Sample Code for OUTN

2.9 Push (PUSH), Stack Read (STCRD), Pop (POP)

Table 2.9.1 Push, Stack Read, and Pop

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Basic Instruc-tion	–	Push	PUSH		✓	–	–	1	1 bit	–
		Stack Read	STCRD		✓	–	–	1	1 bit	–
		Pop	POP		✓	–	–	1	1 bit	–

■ Parameter



Note: Neither PUSH, STCRD, nor POP instructions are represented in bold in an actual ladder sequence program (circuit).

F020901.VSD

■ Function

● Push

The PUSH saves the result (ON/OFF) of the preceding logical operation. Make sure that a single circuit contains not more than 16 branches (Pushes).

● Stack Read

The STCRD instruction reads out the pushed result of the logical operation and passes it to the next processing.

● Pop

The POP instruction reads out the pushed result of the logical operation and passes it to the next processing. In addition, the instruction clears the result of the logical operation in the stack.

Making use of branches (Push, Stack Read, and Pop) saves the number of coding steps and makes the program more readable. Make sure that the numbers of Pushes and Pops are the same.

● Program with Branches

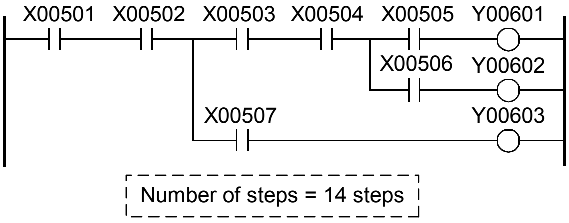


Figure 2.9.1 Program with Branches

● Program with no Branch

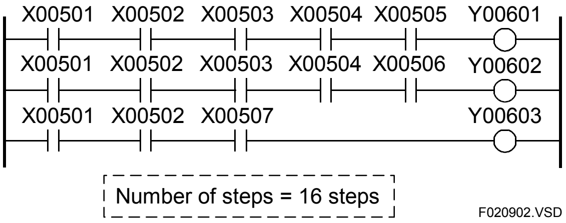


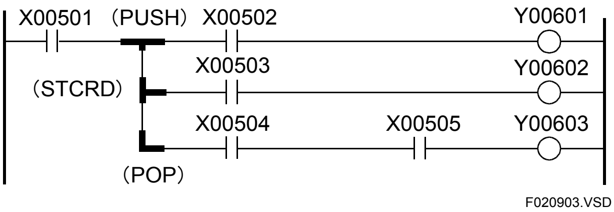
Figure 2.9.2 Program with no Branch

TIP

Since these instructions are generated automatically within WideField3, WideField2, WideField, or Ladder Diagram Support Program M3 (circuit) where the beginning of a branch, branch, and end of a branch occur, you need not enter these instructions in your ladder. Their mnemonics, however, need to be specified as instructions as they are not generated automatically within a program. These instructions cannot be monitored using the device monitor function of WideField3, WideField2, WideField, or Ladder Diagram Support Program M3.

■ Programming Example

● Sample Code Using Branches (1)

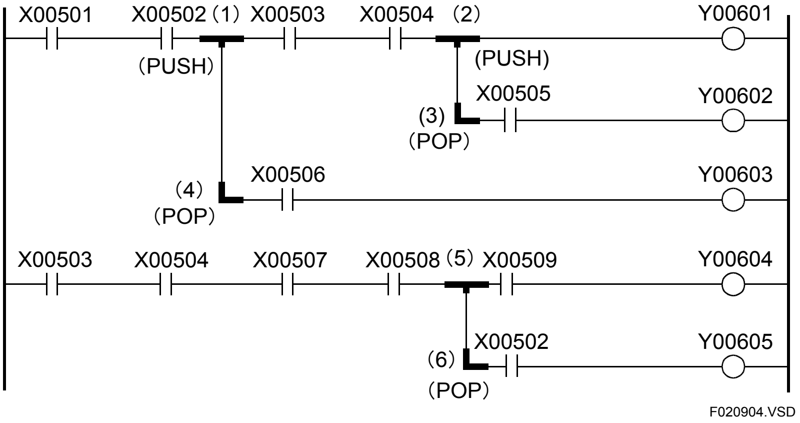


Note: Neither PUSH, STCRD, nor POP appears in the ladder diagram.

Line No.	Instruction	Operands				
0001	LD	X00501				
0002	PUSH					
0003	AND	X00502				
0004	OUT	X00601				
0005	STCRD					
0006	AND	X00503				
0007	OUT	X00602				
0008	POP					
0009	AND	X00504				
0010	AND	X00505				
0011	OUT	Y00603				

Figure 2.9.3 Sample Code Using Branches (1)

● Sample Code Using Branches (2)



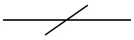
Line No.	Instruction	Operands					
0001	LD	X00501					
0002	AND	X00502					
0003	PUSH						(1)
0004	AND	X00503					
0005	AND	X00504					
0006	PUSH						(2)
0007	OUT	Y00601					
0008	POP						(3)
0009	AND	X00505					
0010	OUT	Y00602					
0011	POP						(4)
0012	AND	X00506					
0013	OUT	Y00603					
0014	LD	X00503					
0015	AND	X00504					
0016	AND	X00507					
0017	AND	X00508					
0018	PUSH						(5)
0019	AND	X00509					
0020	OUT	Y00604					
0021	POP						(6)
0022	AND	X00502					
0023	OUT	Y00605					

Figure 2.9.4 Sample Code Using Branches (2)

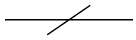
2.10 Inverter (INV)

F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 2.10.1 Inverter

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Processing Unit	Carry
					Yes	No		Without Index Modification	With Index Modification		
Basic Instruction	–	Inverter	INV		✓	–	–	1	–	1 bit	–

■ Parameter

Inverter 

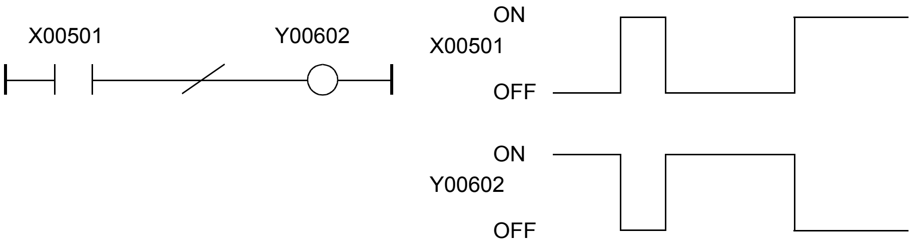
F021001.VSD

■ Function

The INV instruction inverts the result of the preceding logical operation and passes it on to the next process.

Table 2.10.2 Inverter Operation

Result of Preceding Operation	Operation Result
ON	OFF
OFF	ON



F021002.VSD

Figure 2.10.1 Timing of Inverter Operation

CAUTION

You cannot insert the INV instruction in place of the LD or OR instructions.

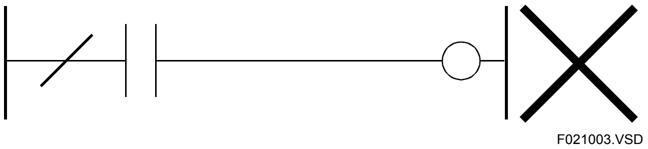


Figure 2.10.2 INV Disallowed (1) Position of LD

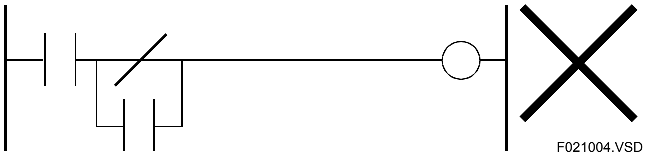


Figure 2.10.3 INV Disallowed (2) Position of OR 1

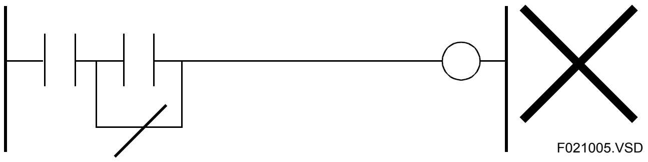


Figure 2.10.4 INV Disallowed (3) Position of OR 2

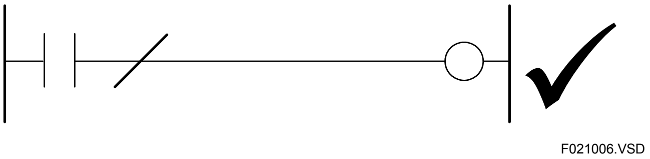


Figure 2.10.5 INV Allowed Position of AND

CAUTION

The result of the preceding logical operation, to which the INV instruction is applied, means a circuit element connected by the ANDLD or ORLD instruction only. For example, in the following figure, the INV instruction inverts not the result of the AND operation of I00001 and I00003, but the value of I00003. For the circuit element, see the description of "ANDLD, ORLD".

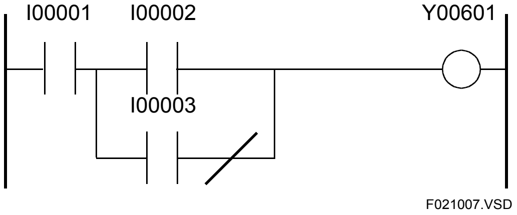
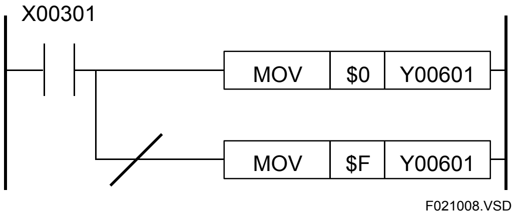


Figure 2.10.6 INV Operation

■ Programming Example

The program shown below moves $0 to the 1-word location starting at Y00601 if X00301 turns ON. It moves $F to the location if X00301 turns OFF.



Line No.	Instruction	Operands				
0001	LD	X00301				
0002	PUSH					
0003	MOV	$0	Y00601			
0004	POP					
0005	INV					
0006	MOV	$F	Y00601			

Figure 2.10.7 Example Inverter Program

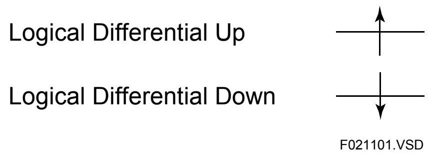
2.11 Logical Differential Up (UP), Logical Differential Down (DWN)

F3SP22 F3SP28 F3SP38	F3SP53 F3SP58 F3SP59	F3SP66 F3SP67	F3SP71 F3SP76

Table 2.11.1 Load Differential Up, Load Differential Down

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Processing Unit	Carry
					Yes	No		Without Index Modification	With Index Modification		
Basic Instruction	303	Logical Differential Up	UP	↑	✓	–	–	1	–	1 bit	–
	304	Logical Differential Down	DWN	↓	✓	–	–	1	–	1 bit	–

■ Parameter



■ Function

● Logical Differential Up

The UP instruction sets and holds a result signal of logical operation to ON for one scan on the rising edge of the signal of the result of the preceding logical operation. The result signal is held to OFF except on the rising edge of the result of the preceding logical operation.

The system automatically allocates a differential relay in a dedicated area. The differential relay holds the results of the logical operation when the UP instruction is executed. The value in the differential relay will be used for the next execution of the UP instruction. If you want to specify the differential relay, use the UPX (logical differential up using a specified device) instruction. The operation of the UP and UPX instructions are the same.

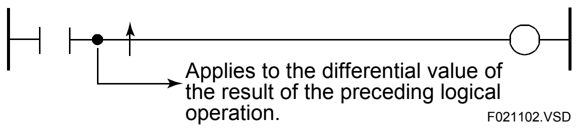

Applies to the differential value of the result of the preceding logical operation.
F021102.VSD

Figure 2.11.1 Operand of Logical Differential Up Instruction

Table 2.11.2 Result of Logical Differential Up Operation

Result of Preceding Operation	Operation Result
ON →ON	OFF
ON →OFF	OFF
OFF →OFF	OFF
OFF →ON	ON

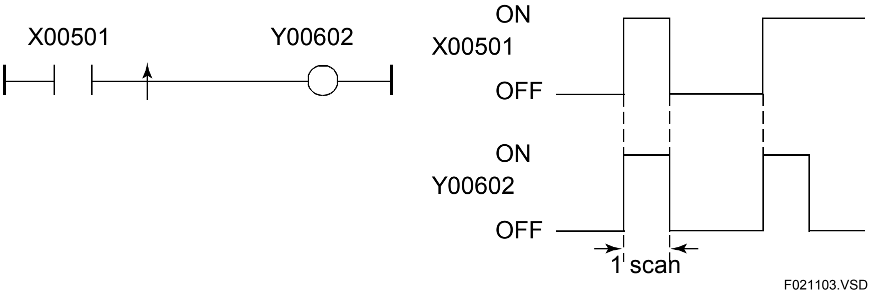


Figure 2.11.2 Timing of Logical Differential Up Operation

● Logical Differential Down

The DWN instruction sets and holds a result signal of logical operation to ON for one scan on the falling edge of the signal of the result of the preceding logical operation. The result signal is held to OFF except on the falling edge of the result of the preceding logical operation.

The system automatically allocates a differential relay in a dedicated area. The differential relay holds the results of the logical operation when the DWN instruction is executed. The value in the differential relay will be used for the next execution of the DWN instruction. If you want to specify the differential relay, use the DWNX (logical differential down using a specified device) instruction. The operation of the DWN and DWNX instructions are the same.

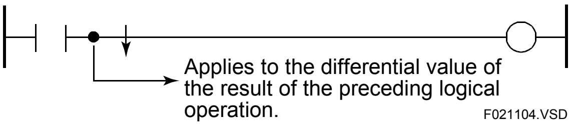

Applies to the differential value of the result of the preceding logical operation.
F021104.VSD

Figure 2.11.3 Operand of Logical Differential Down Instruction

Table 2.11.3 Result of Logical Differential Down Operation

Result of Preceding Operation	Operation Result
ON →ON	OFF
ON →OFF	ON
OFF →OFF	OFF
OFF →ON	OFF

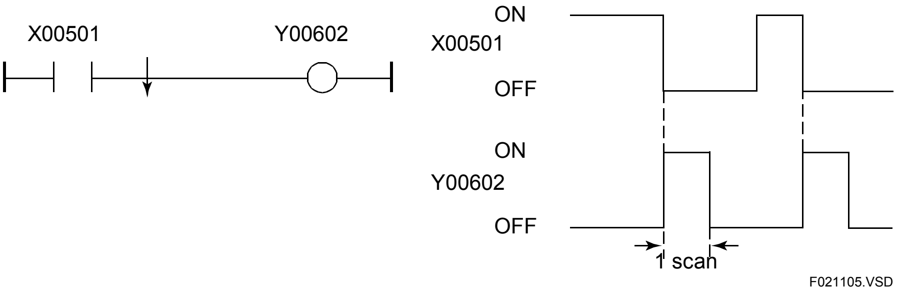


Figure 2.11.4 Timing of Logical Differential Down Operation

CAUTION

You cannot insert the UP and DWN instructions in place of the LD or OR instructions.

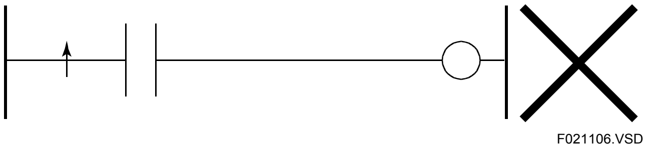


Figure 2.11.5 UP Disallowed (1) Position of LD

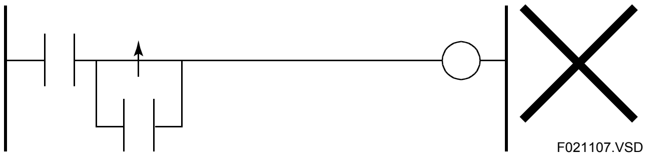


Figure 2.11.6 UP Disallowed (2) Position of OR 1

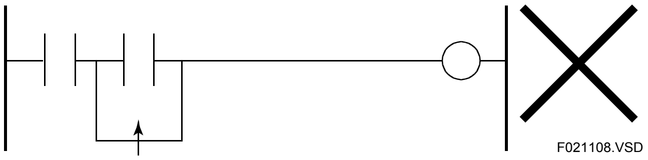


Figure 2.11.7 UP Disallowed (3) Position of OR 2

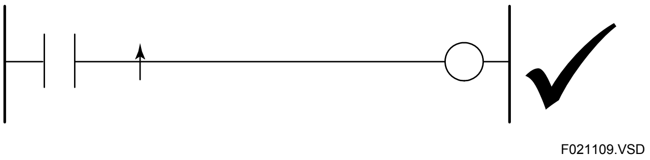


Figure 2.11.8 UP Allowed Position of AND

CAUTION

The result of the preceding logical operation, to which the UP and DWN instructions are applied, means a circuit element connected by the ANDLD or ORLD instruction only. For example, in the following figure, the UP instruction is applied to not the result of the AND operation of I00001 and I00003, but the value of I00003. For the circuit element, see Section 2.6, "ANDLD, ORLD".

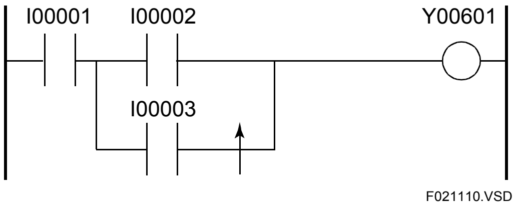
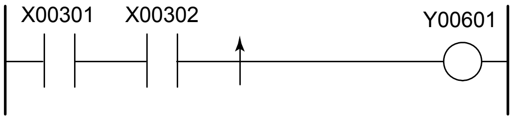


Figure 2.11.9 UP Operation

■ Programming Example

● Logical Differential Up

The program shown below sets Y00601 to ON for one scan when the result of AND of X00301 and X00302 switches from OFF to ON.



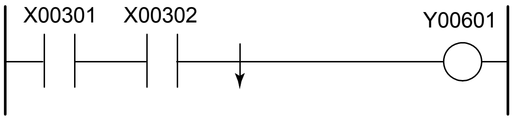
Line No.	Instruction	Operands				
0001	LD	X00301				
0002	AND	X00302				
0003	UP					
0004	OUT	Y00601				

F021111.VSD

Figure 2.11.10 Example of a Program Using Logical Differential Up Operation

● Logical Differential Down

The program shown below sets Y00601 to ON for one scan when the result of AND of X00301 and X00302 switches from ON to OFF.



Line No.	Instruction	Operands				
0001	LD	X00301				
0002	AND	X00302				
0003	DWN					
0004	OUT	Y00601				

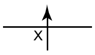
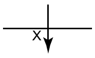
F021112.VSD

Figure 2.11.11 Sample Code Using Logical Differential Down Operation

2.12 Logical Differential Up Using Specified Device (UPX), Logical Differential Down Using Specified Device (DWNX)

F3SP22 F3SP28 F3SP38	F3SP53 F3SP58 F3SP59	F3SP66 F3SP67	F3SP71 F3SP76

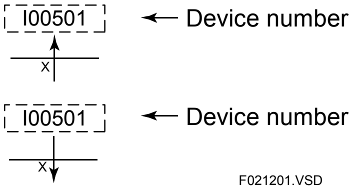
Table 2.12.1 Load Differential Up and Down Using Specified Device

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Pro-cessing Unit	Carry
					Yes	No		Without Index Modifi-cation	With Index Modifi-cation		
Basic Instruc-tion	305	Logical Differential Up Using Specified Device	UPX		✓	—	—	1	2	1 bit	—
	306	Logical Differential Down Using Specified Device	DWNX		✓	—	—	1	2	1 bit	—

■ Parameter

Logical Differential Up
Using Specified Device

Logical Differential Down
Using Specified Device



■ Available Devices

Table 2.12.2 Devices Available for Logical Differential Up Down Using Specified Device Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
Device		✓	✓	✓*1	✓*1												Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The UPX and DWNX instructions can be used when you want to perform a logical differential instruction in a FOR-NEXT instruction or save the output of the logical differential instructions in the event of a power failure. With the UPX and DWNX, you must specify a relay device to save the result of the preceding operation. Other than that, the operation of the UPX and DWNX instructions are the same as the UP and DWN instructions, respectively.

● Use in FOR-NEXT instruction

By combining a logical differential instruction with index modification, a differential operation can be performed in a FOR-NEXT instruction every loop.

The program shown below sets Y0060n to ON for one scan when X0030n switches from OFF to ON (n = 1 to 3).

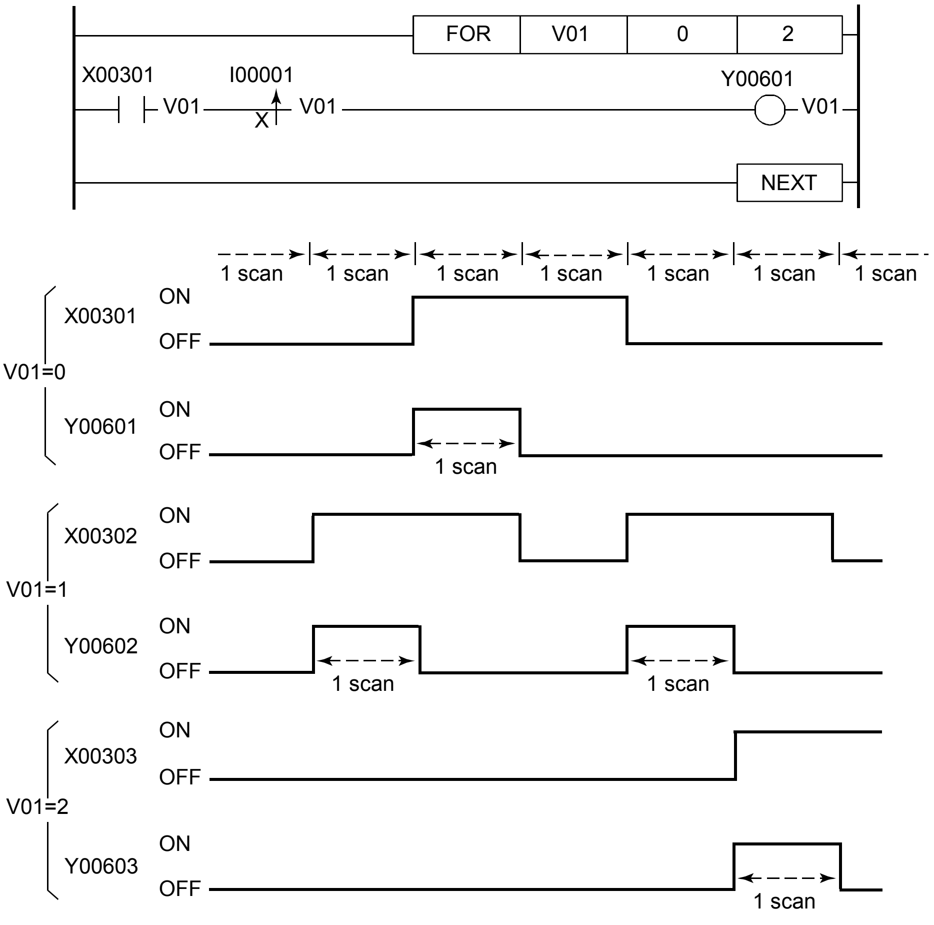
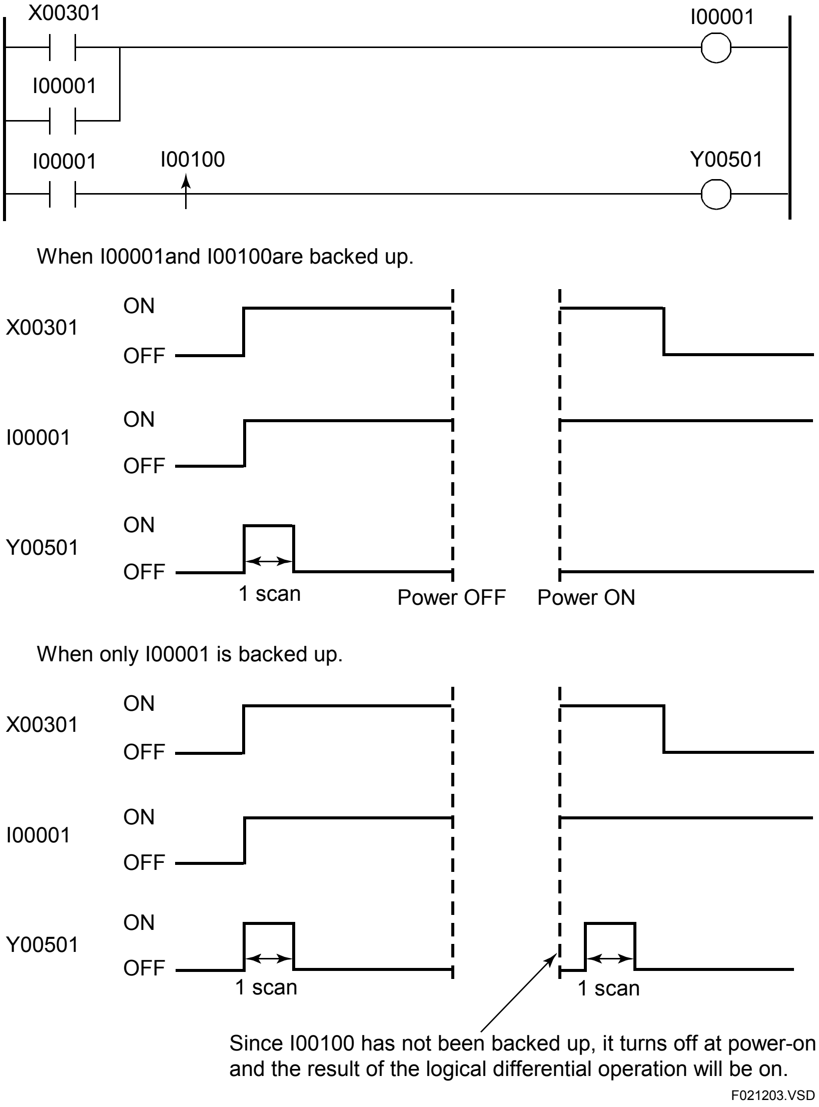


Figure 2.12.1 Performing Logical Differential Operation in FOR-NEXT Instruction

● Use for Saving Differential Circuit in the Event of a Power Failure

By specifying a back-up relay with the UPX and DWNX instructions, differential circuits can be backed up in case of a power failure.



When I00001and I00100are backed up.

When only I00001 is backed up.

Since I00100 has not been backed up, it turns off at power-on and the result of the logical differential operation will be on.

Figure 2.12.2 Operation of UPX Instruction

CAUTION

You cannot insert the UPX and DWNX instructions in place of the LD or OR instructions.

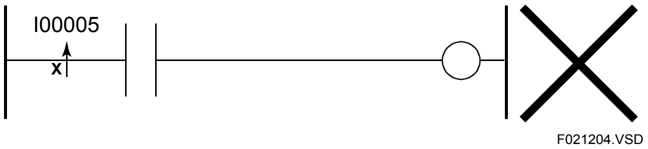


Figure 2.12.3 UPX Disallowed (1) Position of LD

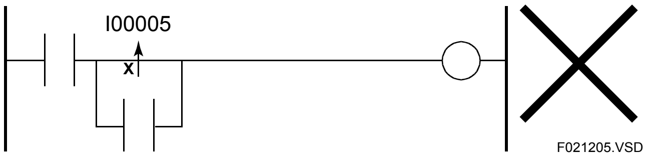


Figure 2.12.4 UPX Disallowed (2) Position of OR 1

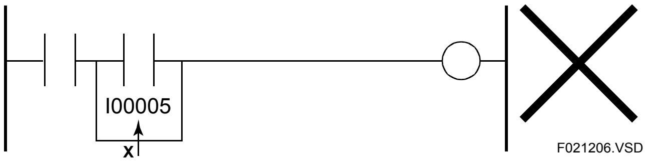


Figure 2.12.5 UPX Disallowed (3) Position of OR 2

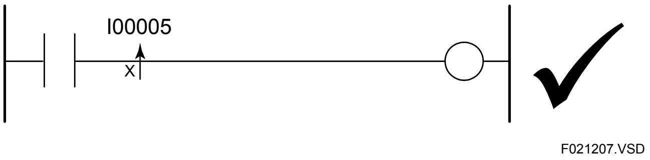


Figure 2.12.6 UPX Allowed Position of AND

CAUTION

The result of the preceding logical operation, to which the UPX and DWNX instructions are applied, means a circuit element connected by the ANDLD or ORLD instruction only. For example, in the following figure, the UPX instruction is applied no not the result of the AND operation of I00001 and I00003, but the value of I00003. For the circuit element, see Section 2.6, "ANDLD, ORLD".

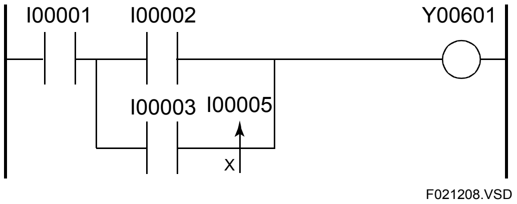
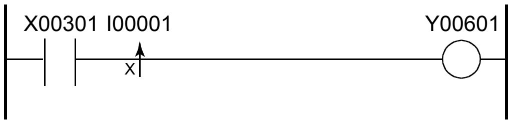


Figure 2.12.7 UPX Operation

■ Programming Example

● Logical Differential Up Using Specified Device

The program shown below sets Y00601 to ON for one scan when X00301 switches from OFF to ON.

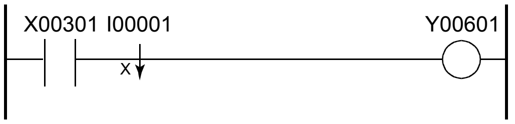


Line No.	Instruction	Operands				
0001	LD	X00301				
0002	UPX	I00001				
0003	OUT	Y00601				

Figure 2.12.8 Example of a Program with Logical Differential Up Operation Using Specified Device

● Logical Differential Down Using Specified Device

The program shown below sets Y00601 to ON for one scan when X00301switches from ON to OFF.

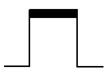
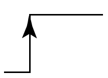
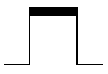
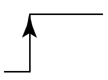


Line No.	Instruction	Operands				
0001	LD	X00301				
0002	DWNX	I00001				
0003	OUT	Y00601				

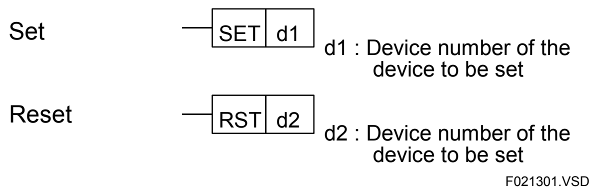
Figure 2.12.9 Example of a Program with Logical Differential Down Operation Using Specified Device

2.13 Set (SET), Reset (RST)

Table 2.13.1 Set, Reset

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Processing Unit	Carry
					Yes	No		Without Index Modification	With Index Modification		
Basic Instruction	01	Set	SET	—[SET |]	✓	–		1	2	1 bit	–
	01P		↑ SET	—[↑SET |]				2	3		
	02	Reset	RST	—[RST |]	✓	–		1	2	1 bit	–
	02P		↑ RST	—[↑RST |]				2	3		

■ Parameter


Set —[SET | d1]
d1 : Device number of the device to be set

Reset —[RST | d2]
d2 : Device number of the device to be set

F021301.VSD

■ Available Devices

Table 2.13.2 Devices Available for the Set and Reset Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification Pointer P
Device		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3									Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Time-out relay
*3: End-of-count relay

■ Function

● Set

The SET instruction sets the specified device to ON when the result of the preceding logical operation is ON or switches from OFF to ON.

The device that is set to ON remains ON even when the result of the preceding logical operation turns off.

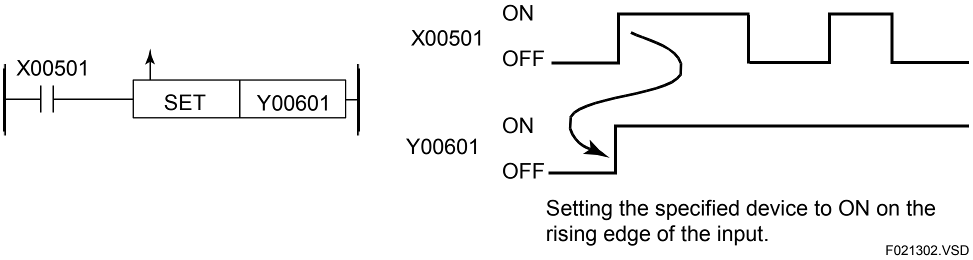

X00501
SET
Y00601
X00501
ON
OFF
Y00601
ON
OFF
Setting the specified device to ON on the rising edge of the input.
F021302.VSD

Figure 2.13.1 Set Timing

Use the execute-while-ON type Set instruction in interrupt routines and other routines where differential type instructions are disallowed.

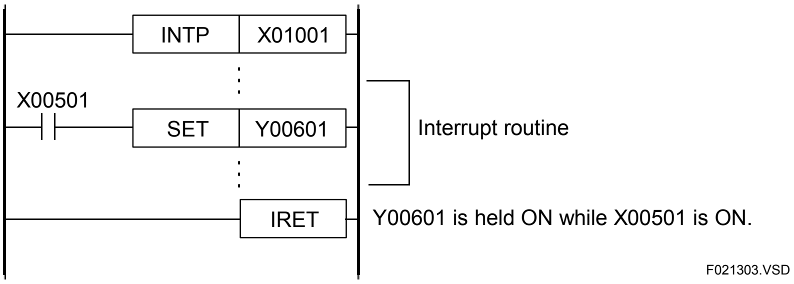


Figure 2.13.2 Setting within an Interrupt Routine

● Reset

The RST instruction sets the specified device to OFF when the result of the preceding logical operation is ON or switches from OFF to ON.

The device that is set to OFF remains OFF even when the result of the preceding logical operation turns off.

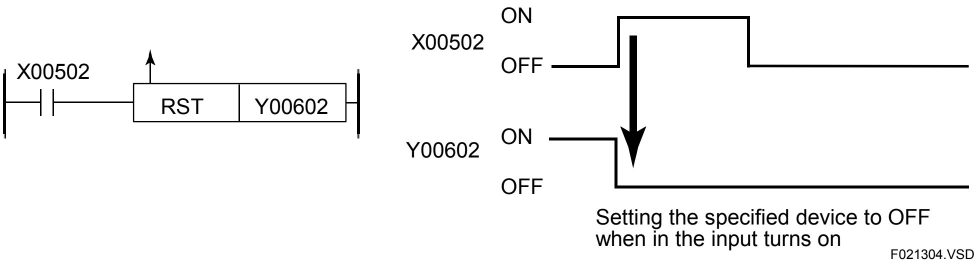

X00502
RST
Y00602
X00502
ON
OFF
Y00602
ON
OFF
Setting the specified device to OFF
when in the input turns on
F021304.VSD

Figure 2.13.3 Reset Timing

Use the execute-while-ON type Reset instruction in interrupt routines and other routines where differential type instructions are disallowed.

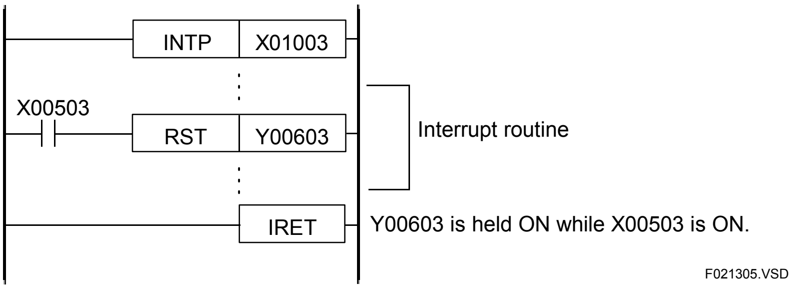
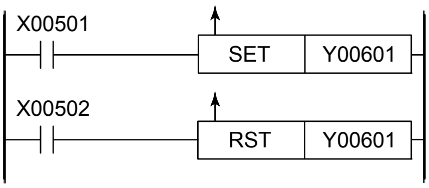


Figure 2.13.4 Resetting within an Interrupt Routine

■ Programming Example

The sample program shown below set Y00601 to ON when X00501 turns on and resets Y00601 to OFF when X00502 turns on.



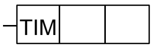
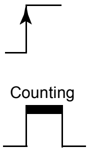
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	↑SET	Y00601				
0003	LD	X00502				
0004	↑RST	Y00601				

F021306.VSD

Figure 2.13.5 Example Program Using Set and Reset Instructions

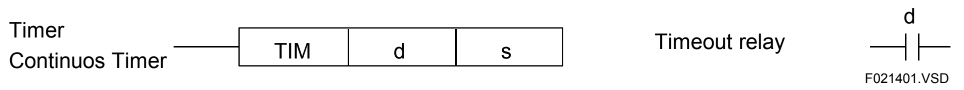
2.14 Timer (TIM)

Table 2.14.1 Timer

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Pro-cessing Unit	Carry
					Yes	No		Without Index Modifi-cation	With Index Modifi-cation		
Basic Instruc-tion	–	Timer	TIM		✓	–	Start time  Counting	2/4[*1]	2/4[*1]	Time-out relay (1 bit) Current value (16 bits)	–
		Continuous Timer									

*1: For F3SP71-4□, F3SP76-7□, and F3SP□□-□S, Step Count is 4

■ Parameter



d: Timer number

Table 2.14.2 Default Timer Numbers

Timer	F3SP21	F3SP22, F3SP25, F3SP28, F3SP53, F3SP66, F3SP71	F3SP35, F3SP38, F3SP58, F3SP59, F3SP67, F3SP76
100µs[*1]	None	None	None
1ms	None	None	None
10ms	T001 to T128	T001 to T512	T0001 to T1024
100ms	T129 to T240	T513 to T960	T1025 to T1920
Continuous 100ms	T241 to T256	T961 to T1024	T1921 to T2048

*1: The 100µs timer is available only on the F3SP22, F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66, F3SP67, F3SP71, and F3SP76.

Note: You can change the default timer numbers using the project configuration

s : Preset value
- Literal: Set either in seconds (s) or milliseconds (ms).
  Example: 10S100MS (10 seconds and 100 milliseconds)
  1.2MS (1 milliseconds and 200 microseconds)

Table 2.14.3 Types of Timers and Their Value Ranges

Timer	Resolution	Setting Range
Timer	100µs[*1]	0.1ms to 3s276.7ms
	1ms	1ms to 32s767ms
	10ms	10ms to 327s670ms
	100ms	100ms to 3276s700ms
Continuous Timer	100ms	100ms to 3276s700ms

*1: The 100µs timer is available only on the F3SP22, F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66, F3SP67, F3SP71, and F3SP76.

Note: You can change the default timer numbers using the project configuration.

When a device is specified: The current value of the device is loaded as the count value (1 to 32767).
1 count = Timer's resolution

■ Available Devices

Table 2.14.4 Devices Available for the Timer Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification Pointer P
d							✓										No	No
s	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes

*1: Timer current value
*2: Counter current value
*3: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The TIM instruction serves as a synchronous countdown timer. The timer continues to count down while its input condition is ON (and remains ON). The time-out relay is set to ON when the timer count reaches 0 (the timer times out when its value reaches 0).

"Synchronous" refers to a mode in which the ON/OFF state of the time-out relay and the timer's current value are held unchanged during the execution of the program for a single scan by performing the ON/OFF switching of the time-out relay and countdown of the timer during the END processing. All timers are provided with a time-out relay. The relationship between the timer's current value and the state of its time-out relay is shown below.

Table 2.14.5 Timer's Current Value

Timer Type	Resolution	Timer's Current Value			
		When the Input Remains ON		When the Input Switches from ON to OFF	When the Input Remains OFF
		Before Time-out	After Time-out		
Timer	100µs	Update (count down)	0	Preset value	Preset value
	1ms				
	10ms				
	100ms				
Continuous Timer	100ms	Update (count down)	0	Retained	Retained

Table 2.14.6 State of Time-out Relay (Contact a)

Timer Type	Resolution	State of Time-out Relay (Contact a)			
		When the Input Remains ON		When the Input Switches from ON to OFF	When the Input Remains OFF
		Before Time-out	After Time-out		
Timer	100µs	OFF	ON	OFF	OFF
	1ms				
	10ms				
	100ms				
Continuous Timer	100ms				

Table 2.14.7 State of Time-out Relay (Contact b)

Timer Type	Resolution	State of Time-out Relay (Contact b)			
		When the Input Remains ON		When the Input Switches from ON to OFF	When the Input Remains OFF
		Before Time-out	After Time-out		
Timer	100µs	ON	OFF	ON	ON
	1ms				
	10ms				
	100ms				
Continuous Timer	100ms				

The relationships between the input, timer's current value, and the time-out relay are shown in the following figures.

Timers (100μs, 1ms, 10ms, and 100ms) and continuous timer behave in the same way.

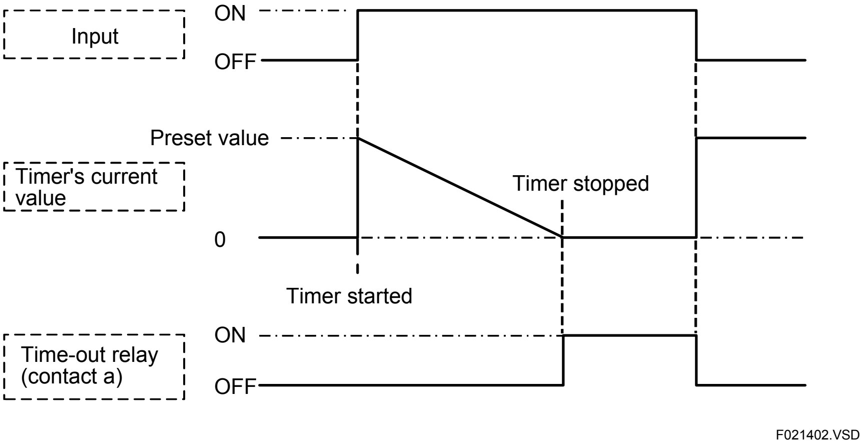


Figure 2.14.1 Timer Action (Time-up)

Timers (100μs, 1ms, 10ms, and 100ms)

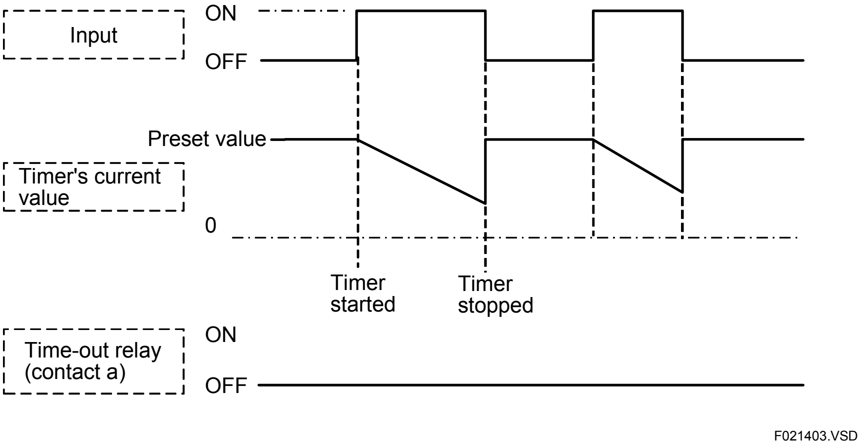


Figure 2.14.2 Timer Action (Input Switching from ON to OFF)

Continuous timer (100ms)

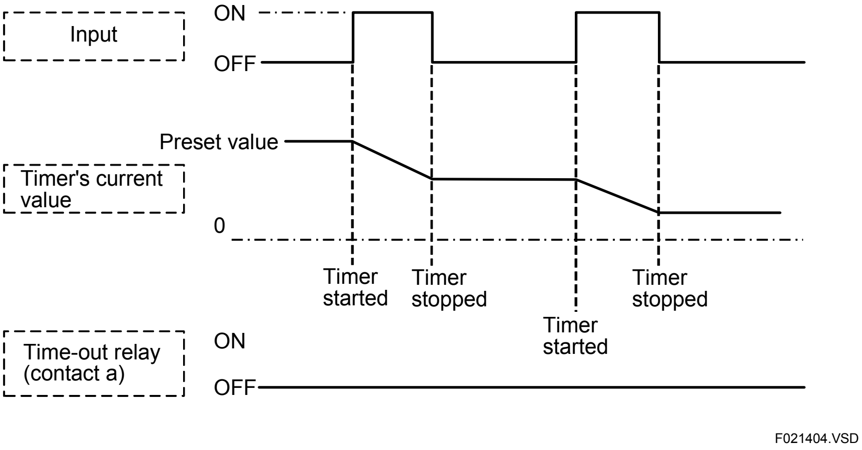


Figure 2.14.3 Continuous Timer Action (Input Switching from ON to OFF)

You can also specify a register listed in Table 2.14.4 as the timer's preset value (s).

The timer's preset value becomes 0 if the register's value is 0.

If the timer's device name is specified as the register in an application instruction whose processing unit is 1 word (16 bits), the contents of the register are taken as the timer's current value (the count value ranges from 0 to 32767).

The timer's update procedures and the timer's accuracy are described below.

(1) Timer update (timer's current value or time-out) is executed in END at the end of every scan. Consequently, the current value of the timer never changes during a scan.

(2) Timer update is suspended while the ladder sequence program is stopped (STOP or PAUSE state). An active timer is reset when the block in which the timer resides enters the STOP state.

(3) Timer update is deferred by at longest one scan period (timer accuracy).

(4) A continuous timer is reset by writing a zero in the transfer mode when the input is off.

(5) The procedures for performing forced set or reset from a program are summarized in the table below.

Table 2.14.8 Forced Timer Set/Reset Procedures

	Non-continuous Timer	Continuous Timer
Forced set	Set the time-out relay to ON when the input condition for Set the time-out relay to ON	Set the time-out relay to ON when the input condition for Set the time-out relay to ON
Forced reset	Set the input condition for the timer instruction to OFF	Set the current value to 0 when the input condition for the timer instruction to OFF

SEE ALSO

For details on Forced Set/Reset, see Section 6.5.3 of "Sequence CPU Instruction Manual – Functions (for F3SP22-0S, F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S)" (IM 34M06P13-01E), Section A6.5.1 of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E), or Section A6.5.1 of "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

CAUTION

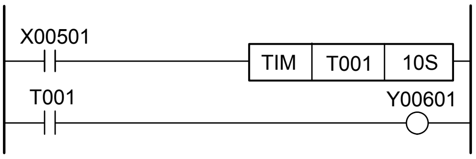
- A TIM instruction cannot be used in an input interrupt.
- A TIM instruction cannot be used in a sensor control block when using CPU modules other than F3SP7□-□□.
- Do not execute a TIM instruction for the same timer number more than once in a scan. In addition, exercise care when executing a TIM instruction in a subroutine, macro or in an iterative manner using a FOR-NEXT instruction or JMP instruction. Improper use may lead to incorrect operation.
- If there are multiple TIM instructions using the same timer in a program, specify the preset values using all constants or all devices. Having multiple timer instructions using the same timer may sometimes cause a failure at uploading.

CAUTION

Since a timer instruction executes in the synchronous mode, the result of forcing it into the set or reset state is reflected in one scan edit process. If the forced set or reset occurs in the program before the location where a TIM instruction is specified, the result is reflected in one scan edit process. Conversely, if the forced set or reset occurs in the program after the location where a TIM instruction is specified, the result is reflected in the next scan edit process.

■ Programming Example

The sample code shown below sets Y00601 to ON 10 seconds after X00501 is set to ON.



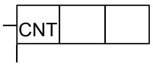
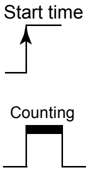
Line No.	Instruction	Operands					
0001	LD	X00501					
0002	TIM	T001	10s				
0003	LD	T001					
0004	OUT	Y00601					

F021405.VSD

Figure 2.14.4 Example of a Timer Program

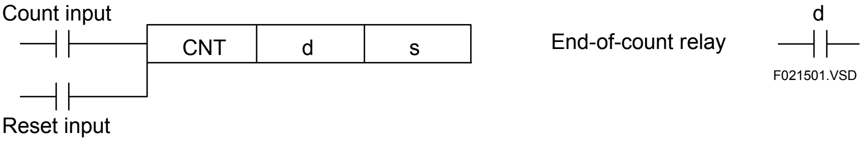
2.15 Counter (CNT)

Table 2.15.1 Counter

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Processing Unit	Carry
					Yes	No		Without Index Modification	With Index Modification		
Basic Instruction	–	Counter	CNT		✓	–		2	2	End-of-count relay (1 bit) Current value (16 bits)	–

■ Parameter

Counter



d: Counter number

s: Preset value

- Literal: Enter a count value (1-32767).

  Example: 1000

- Device specification: The device’s current value is taken as the count value (1-32767).

■ Available Devices

Table 2.15.2 Devices Available for the Counter Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification Pointer P
d								✓									No	No
s	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes

*1: Timer current value
*2: Counter current value
*3: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Count instruction serves as a count-down counter. The counter decrements the preset value by 1 every time the result of the previous operations (input) switches from OFF to ON. The end-of-count relay is set to ON when the counter's current value reaches 0 (the counter is said to have reached end-of-count when its current value reaches 0). The counter will not count when the result of operations (input) remains ON, OFF, or switches from ON to OFF. All counters are provided with an end-of-count relay. The relationship between the counter's current value and the state of its end-of-count relay is shown in Tables 2.15.3, 2.15.4, and 2.15.5. A counter must be reset by a reset input before it is given a count input. Normal counter operation cannot be guaranteed unless a counter is given count inputs without being reset in advance.

When a count input and a reset input occur simultaneously, the reset input takes precedence and the counter will not count.

Table 2.15.3 Counter's Current Value

Item	When the Input Switched from OFF to ON		When the Input Remains ON	When the Input OFF	When the Reset Input is ON	When the Input Switches from ON to OFF
	Before End-of-count	After End-of-count				
Counter’s Current value	Update (count down)	0	The current value is retained	The current value is retained.	Preset value	Preset value

Table 2.15.4 State of End-of-count Relay (Contact a)

Item	Before End-of-count	After End-of-count	When the Reset Input is ON
State of End-of-count Relay (Contact a)	OFF	ON	OFF

Table 2.15.5 State of End-of-count Relay (Contact b)

Item	Before End-of-count	After End-of-count	When the Reset Input is ON
State of End-of-count Relay (Contact b)	ON	OFF	ON

The relationships between the reset input, count input, counter's current value, and the end-of-count relay are shown in the following figures.

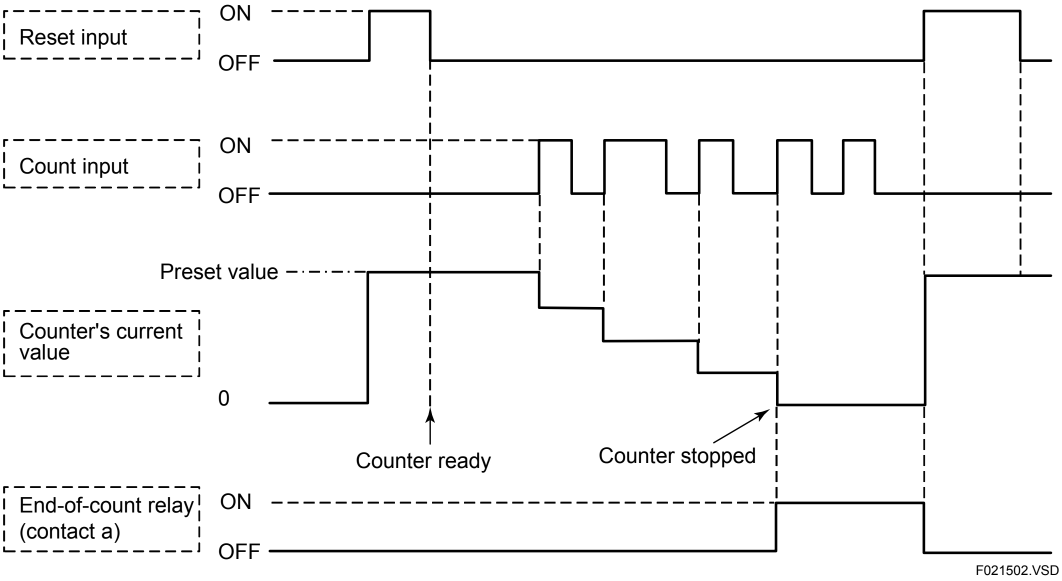


Figure 2.15.1 Counter Operation (When There is No Conflict between Reset and Count Inputs)

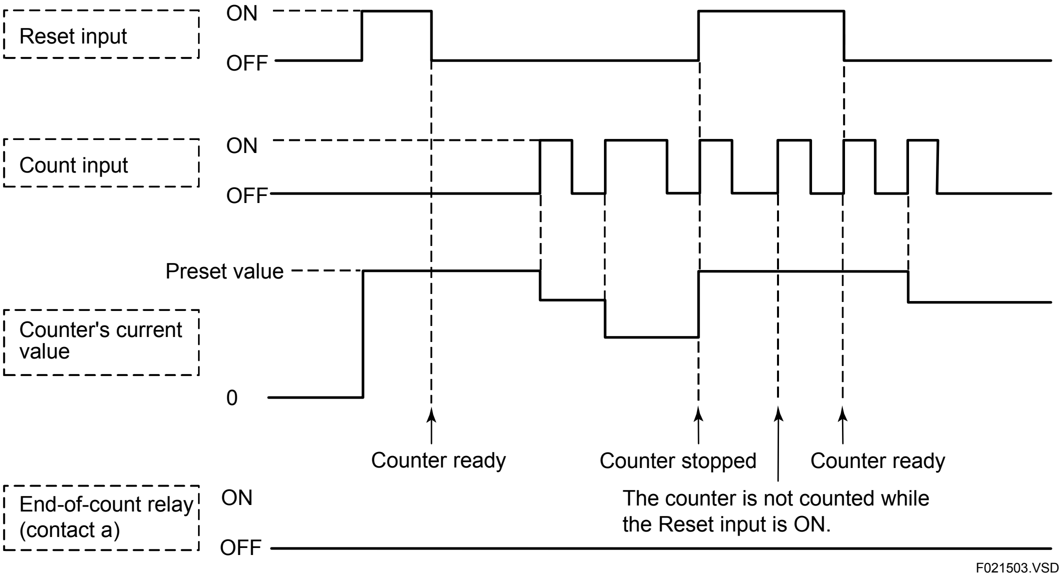


Figure 2.15.2 Counter Operation (When There is a Conflict between Reset and Count Inputs)

You can also specify a register listed in Table 2.15.2 as the counter’s preset value (s). If the register's value exceeds the counter's maximum value (32767), the lowest-order 15 bits of the register are taken as forming the counter's preset value. If the register value is 0, the counter operates with a preset value of 0

If the counter's device name is specified as the register in an application instruction whose processing unit is 1 word (16 bits), the contents of the register are taken as the counter's current value.

The counter's update procedures and accuracy are described below.

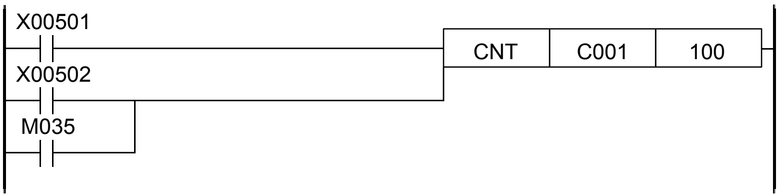
(1) Counter update (counter's current value or end-of-count) is executed at rising edge of the count input. Consequently, the current value of the counter may differ before and after its update even within the same scan period.

(2) There is no delay in counter update (the counter is updated without delay after the rising edge of the count input).

(3) The counter's current value is retained even if the block in which the counter resides enters the inactive state.

(4) The counter is of holding type as default (user-definable in the configuration setup) whose current value is retained by a backup battery when power is turned off.

(5) To have a counter reset automatically at power-on time, insert an M035 (a special relay which turns on only on the first scan that occurs after operation starts) to the counter's reset input. To resume counting at the old value at power-on time, however, you need not insert an M035 to the counter's reset input.



F021504.VSD

Figure 2.15.3 Automatic Counter Reset Circuit That Resets the Counter at Start Time

(6) There are situations where you would want to reset a counter when the program execution mode set to "Specified Blocks" before starting a block containing the counter. A sample code that is useful in such cases is shown in the following figure.

SEE ALSO

For details on the execution of specified blocks, see Section 6.4.2 of "Sequence CPU Instruction Manual – Functions (for F3SP22-0S, F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S)" (IM 34M06P13-01E), Section A6.4.2 of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E), or Section A6.4.2 of "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

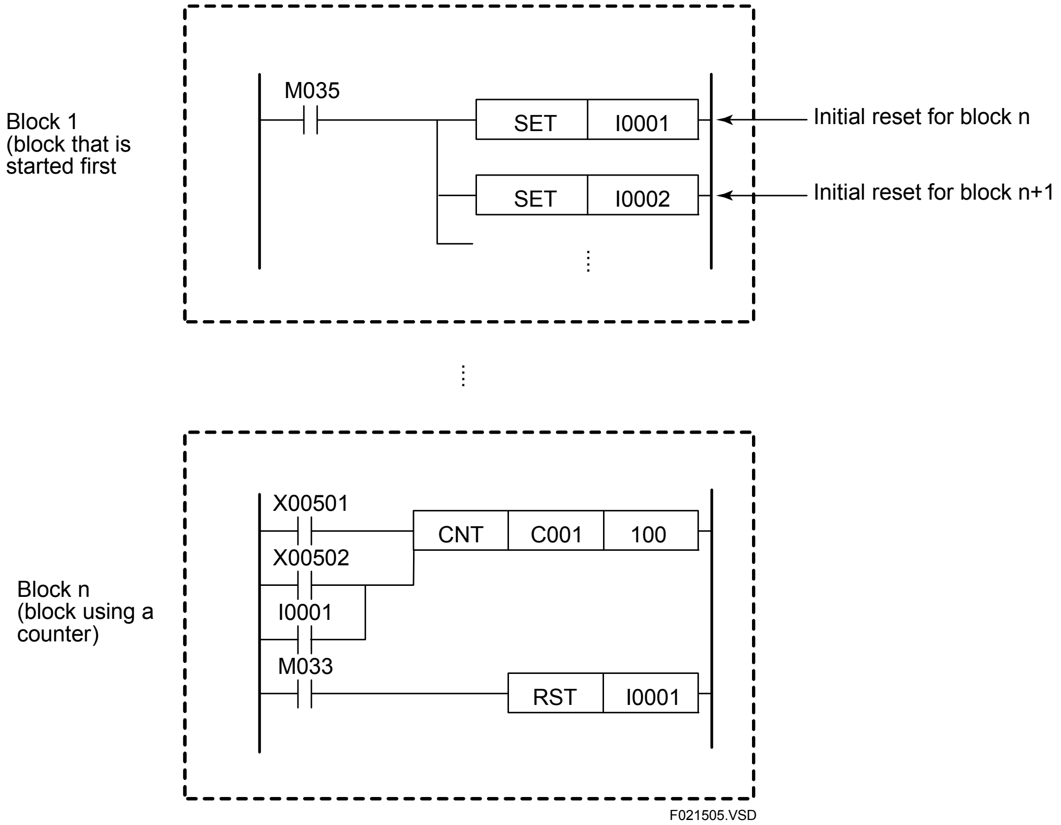


Figure 2.15.4 Method of Resetting a Counter in a Specified Block

Note: M033 is a special relay whose state is always ON.
M035 is a special relay that turns on only on the first scan after operation starts.

Explanation of figure

The counter C001 in block n has to reset inputs: I0001 and X00502. These inputs have the following roles:

I0001: Initial reset input to counter C0001. This input resets counter C001 when block n is activated for the first time. This input does not function as reset input during the second and subsequent executions of block n (because the input is forced to OFF at the end of block n).

X00502: Application reset input to counter C0001. Application-specific resets are carried out using this input relay. Any type of relay, such as an internal relay, may be used as the application reset input though an input relay is used in the figure.

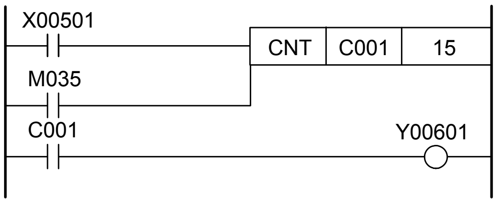
(7) The procedures for effecting forced set/reset from a program are summarized below.

Table 2.15.6 Forced Counter Set/Reset Procedures

	Forced Set/Reset Procedures
Forced set	Set the current value to 0 or turn on the end-of-count relay
Forced reset	Set the reset input to ON.

■ Programming Example

The sample code shown below sets Y00601 to ON when X00501 has turned on 15 times.



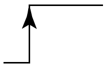
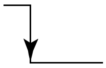
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	LD	M035				
0003	CNT	C001	15			
0004	LD	C001				
0005	OUT	Y00601				

F021506.VSD

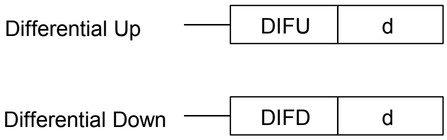
Figure 2.15.5 Example of a Counter Program

2.16 Differential Up (DIFU), Differential Down (DIFD)

Table 2.16.1 Differential Up, Differential Down

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Processing Unit	Carry
					Yes	No		Without Index Modification	With Index Modification		
Basic Instruction	03	Differential Up	DIFU		✓	–		2	3	1 bit	–
	04	Differential Down	DIFD		✓	–		2	3	1 bit	–

■ Parameter



d : Device number of the device to perform a 1-scan-ON output

d : Device number of the device to perform a 1-scan-ON output

F021601.VSD

■ Available Devices

Table 2.16.2 Devices Available for Differential Up and Differential Down Instructions

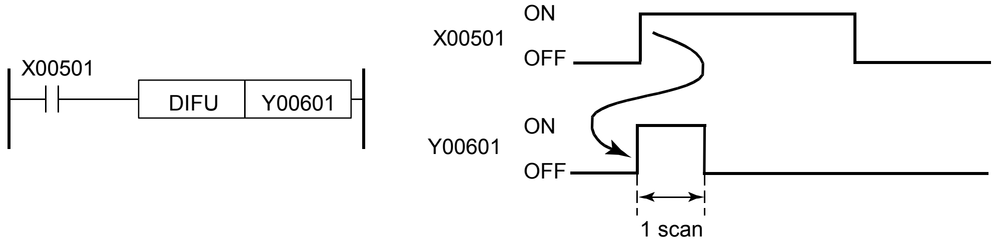
Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3									Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Time-out relay
*3: End-of-count relay

■ Function

● Differential Up

The Differential Up instruction sets and holds a specified device to ON for 1 scan period on the rising edge (OFF to ON) of the input signal. The specified device is held OFF except on the rising (OFF to ON) edge of the input signal.



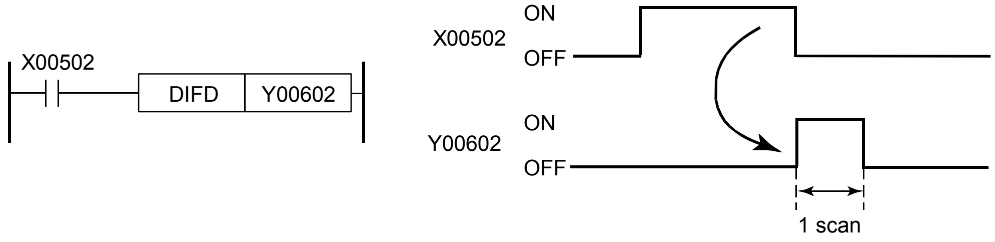
Set the specified device to ON for 1 scan period on the rising edge of the input.

F021602.VSD

Figure 2.16.1 Timing of Differential Up Operation

● Differential Down

The Differential Down instruction sets and holds a specified device to ON for 1 scan period on the falling edge (ON to OFF) of the input signal. The specified device is held OFF except on the falling (ON to OFF) edge of the input signal.



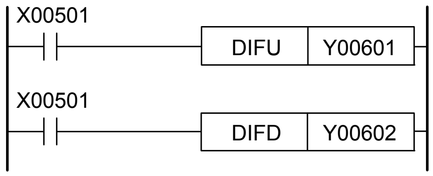
Set the specified device to ON for 1 scan period on the falling edge of the input.

F021603.VSD

Figure 2.16.2 Timing of Differential Down Operation

■ Programming Example

The sample code shown below sets and holds Y00601 to ON for 1 scan period when X00501 switches from OFF to ON and sets and holds Y00602 to ON for 1 scan period when X00501 switches from ON to OFF.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	DIFU	Y00601				
0003	LD	X00501				
0004	DIFD	Y00602				

F021604.VSD

Figure 2.16.3 Example of a Differential Up/Down Program

2.17 Flip-Flop (FF)

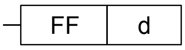
F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 2.17.1 Flip-Flop

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Processing Unit	Carry
					Yes	No		Without Index Modification	With Index Modification		
Basic Instruction	308	Flip-Flop	FF	FF	✓	–	↑	2	3	1 bit	–

■ Parameter

Flip-Flop



F021701.VSD

d: Device number to output the operation result.

■ Available Devices

Table 2.17.2 Devices Available for Flip-Flop Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3									Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Time-out relay
*3: End-of-count relay

■ Function

The FF instruction is an output instruction. It reads a specified coil (device) every rising edge of the input and inverts and outputs it to another specified coil (device).

Table 2.17.3 Flip-Flop Operation

Result of Preceding Operation	Operation Result
ON →ON	Hold
ON →OFF	Hold
OFF →OFF	Hold
OFF →ON	Reverse

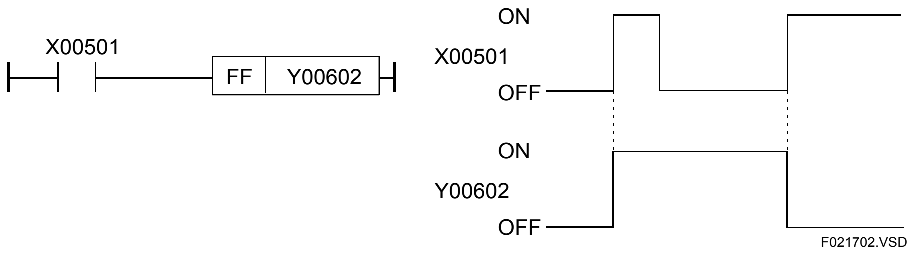
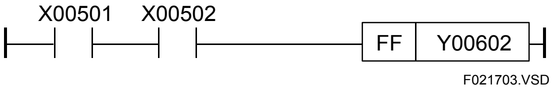


Figure 2.17.1 Timing of Flip-Flop Operation

■ Programming Example

The program shown below inverts Y00602 when the result of the AND operation of X00501 and X00502 switches from OFF to ON.

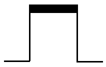


Line No.	Instruction	Operands				
0001	LD	X00501				
0002	AND	X00502				
0003	FF	Y00602				

Figure 2.17.2 Example of a Program with Flip-Flop Operation

2.18 Interlock (IL), Interlock Clear (ILC)

Table 2.18.1 Interlock, Interlock Clear

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Basic Instruction	05	Interlock	IL	IL	✓	–		1	–	–
	06	Interlock Clear	ILC	ILC	–	✓	–	1	–	–

■ Parameter

Interlock IL

InterlockClear ILC

F021801.VSD

■ Function

● Interlock

The Interlock (IL) instruction identifies the beginning of interlocking processing. The program area between the Interlock (IL) instruction and the Interlock Clear (ILC) instruction is called an interlock area. If the interlock condition (the relay immediately before the interlock) is on, the program in the interlock area is executed normally. If the interlock condition is off, executing instructions in the interlock area result in the device states shown in the table below.

Table 2.18.2 Device Status When the Interlock Condition is OFF

Device	Condition
Timer	Reset.
Continuous Timer	The current value is retained.
Counter	The current value is retained
Destination of OUT	Set to OFF*1
Destination of OUTN	Set to OFF*1
Destination of FF	Set to OFF*1
Destination of OUTW	Set to OFF*1
Destination of OUTW L	Set to OFF*1
Other devices	The old state is retained (no instruction is executed).

*1: Set any devices (coils) whose output needs to be held ON when the interlock condition is OFF to ON with the Set instruction.

CAUTION

Although FOR and NEXT instructions in an interlock area are not executed, any instructions in a FOR-NEXT loop operate based on Table 2.18.2. Therefore, if a FOR instruction uses an index device as a loop counter and the index device applies index modification to an OUT instruction, the device that is index-modified by the index device value before the FOR instruction is set to OFF.

CAUTION

- If the interlock input is set to OFF and interlocking processing is already started, a differential instruction in an interlock area operates differently depending on whether the instruction is an input (LDU/LDD/UP/DWN/UPX/DWNX) or output (DIFU/DIFD/pulse type instructions such as MOV P) instruction.

  For differential input instructions, any differential operations are not executed during an interlocking processing. Therefore, when the interlock is cleared, the preceding cycle execution condition flag still holds the value before the interlocking processing starts. For differential output instructions, differential operations are executed during an interlocking processing (they only hold the value and do not output it). Therefore, when the interlock is cleared, the preceding cycle execution condition flag holds the value when the instruction was previously executed.
- The DI/EI/CBD/CBE instructions in an interlock area are executed even if the interlock input is set to OFF and the interlocking processing is already started.

● Interlock Clear

The ILC instruction identifies the end of interlock processing.

● Interlock Nesting

- The F3SP22-0S, F3SP28-3S, F3SP38-6S, F3SP53-4S, F3SP58-6S, F3SP59-7S, F3SP66-4S, F3SP67-6S, F3SP71-4N, F3SP76-7N, F3SP71-4S and F3SP76-7S allow interlock processing to be nested up to 8 layers.

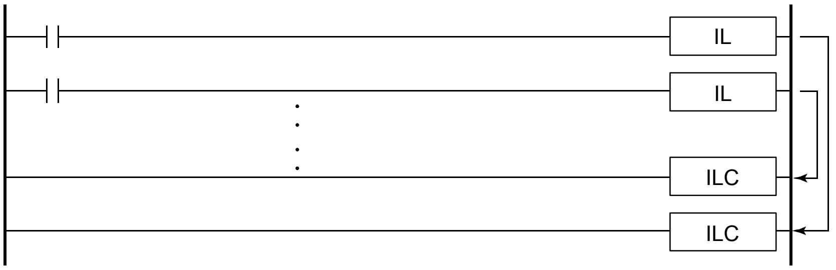


Figure 2.18.1 IL and ILC Nesting

- Other sequence CPU modules do not allow interlock nesting.

CAUTION

Do not execute a jump operation across an interlock area boundary, out of or into an interlock area. Otherwise, the program may not behave as expected.

Making use of the Interlock and Interlock Clear instructions when performing two or more output or ordinary operations under the same input condition saves the number of coding steps and makes the program more readable.

- Program that makes use of the Interlock and Interlock Clear instructions

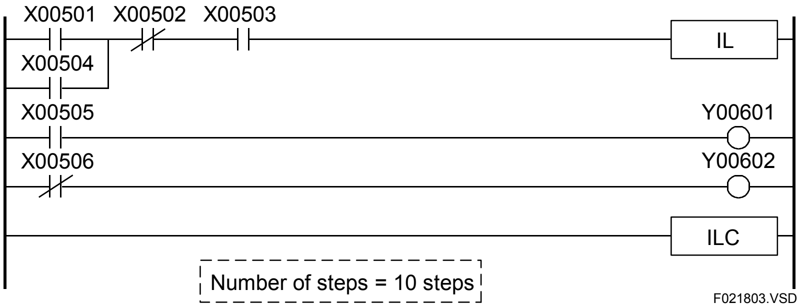


Figure 2.18.2 Program that makes use of the Interlock and Interlock Clear instructions

- Program that makes no use of the Interlock and Interlock Clear instructions

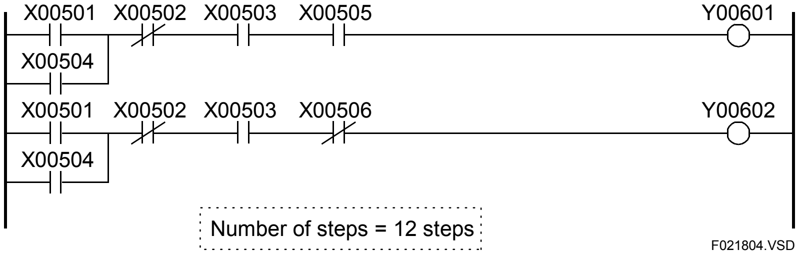
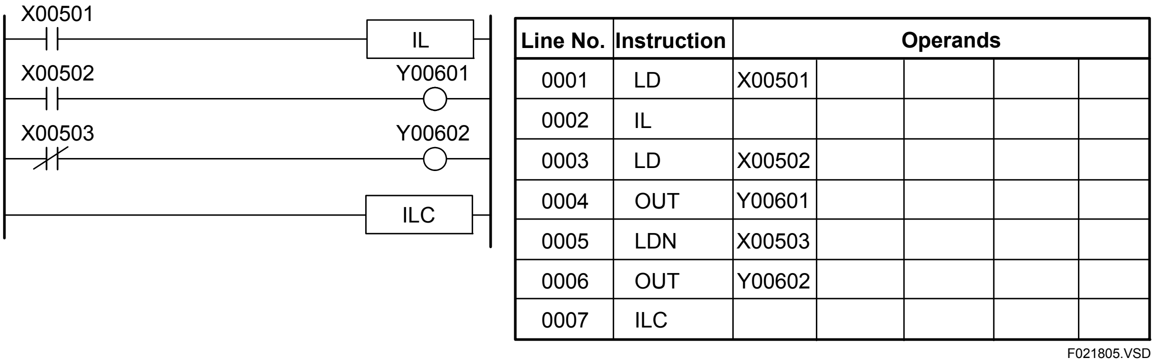


Figure 2.18.3 Program that makes no use of the Interlock and Interlock Clear instructions

■ Programming Example

The sample code shown below turns on the interlock if X00501 is on and turns off the interlock if X00501 is off.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	IL					
0003	LD	X00502				
0004	OUT	Y00601				
0005	LDN	X00503				
0006	OUT	Y00602				
0007	ILC					

Figure 2.18.4 Sample Code for the Interlock and Interlock Clear Instructions

2.19 Load Specified Bit (LDW/LDW L)

F3SP22-0S F3SP28-3S F3SP38-6S | F3SP53-4S F3SP58-6S F3SP59-7S | F3SP66 F3SP67 | F3SP71 F3SP76

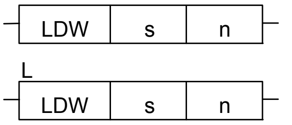
Table 2.19.1 Load Specified Bit, Load Specified Long-word Bit

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Processing Unit	Carry
					Yes	No		Without Index Modification	With Index Modification		
Basic Instruction	311	Load Specified Bit	LDW	LDW	—	✓	—	3	3	16 bits	—
	311L	Load Specified Long-word Bit	LDW L	L LDW				3	3	32 bits	

■ Parameter

Load Specified Bit

Load Specified Long-word Bit



F022101.VSD

s : Device containing a bit to be loaded

n : Position of a bit to be loaded

Load Specified Bit (0-15)

Load Specified Long-word Bit (0-31)

■ Available Devices

Table 2.19.2 Devices Available for Load Specified Bit, Load Specified Long-word Bit

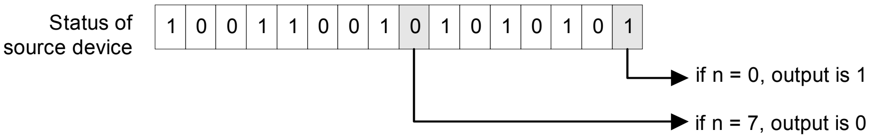
Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓	✓*1	✓	✓	✓	✓		Yes	Yes
n	✓	✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓	✓*1	✓	✓	✓	✓	✓	No	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value (may not be used as a parameter for the Long-word)
*3: Counter current value (may not be used as a parameter for the Long-word)

■ Function

● Load Specified Bit

The LDW instruction outputs the status of a specified bit of a 16-bit data item as a contact a. Position 0 represents the least significant bit, and Position 15 represents the most significant bit of the 6-bit data. If the specified bit position exceeds 15 (n > 15), the numerical value (0 to15) of the lower four bits of n is used to determine the bit position.

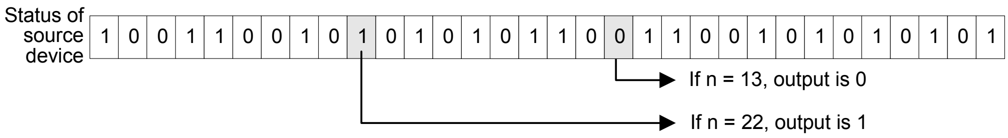


F022102.VSD

Figure 2.19.1   Example of a Load Specified Bit Operation

● Load Specified Long-word Bit

The LDW L instruction outputs the status of a specific bit of a 32-bit data item as a contact a. Position 0 represents the least significant bit, and Position 31 represents the most significant bit of the 32-bit data. If the specified bit position exceeds 31 (n > 31), the numerical value (0 to31) of the lower five bits of n is used to determine the bit position.

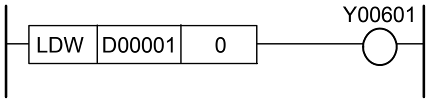


F022103.VSD

Figure 2.19.2   Example of a Load Specific Long-word Bit Operation

■ Programming Example

The sample code shown below turns on Y00601 if the least significant bit of D00001 is '1'.



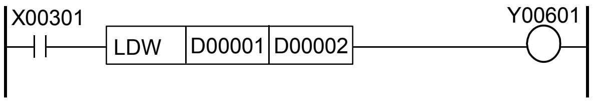
Line No.	Instruction	Operands				
0001	LDW	D00001	0			
0002	OUT	Y00601				

F022104.VSD

Figure 2.19.3   Sample Code for Load Specified Bit

CAUTION

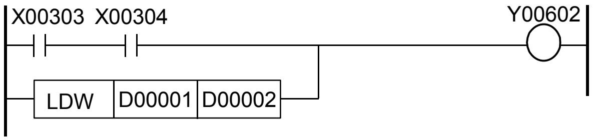
A bit test instruction represented in mnemonic functions like a normal circuit component in a program. Therefore, to represent the following ladder diagram containing a bit test instruction to a mnemonic-based program, you must insert ANDLD or ORLD after it, as shown below.



```
Mnemonic
LD      X00301
LDW     D00001 D00002
ANDLD          ←————————ANDLD is needed.
OUT     Y00601
```

F022105.VSD

Figure 2.19.4 Contact a ANDed with Load Specified Bit Instruction



```
Mnemonic
LD      X00303
AND     X00304
LDW     D00001 D00002
ORLD           ←————————ORLD is needed.
OUT     Y00602
```

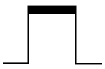
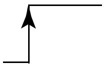
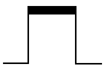
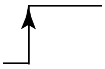
F022106.VSD

Figure 2.19.5 Contact a ORed with Load Specified Bit Instruction

2.20 Out Specified Bit (OUTW/OUTW L)

F3SP22-0S
F3SP28-3S
F3SP38-6S | F3SP53-4S
F3SP58-6S
F3SP59-7S | F3SP66
F3SP67 | F3SP71
F3SP76

Table 2.20.1 Out Specified Bit, Out Specified Long-word Bit

Classi-fication	FUNC No.	Instruc-tion	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Pro-cessing Unit	Carry
					Yes	No		Without Index Modifi-cation	With Index Modifi-cation		
Basic Instruction	312	Out Specified Bit	OUTW	OUTW	✓	–		3	3	16 bits	–
	312P		↑OUTW	↑ OUTW				4	4		
	312L	Out Specified Long-word Bit	OUTW L	L OUTW	✓	–		3	3	32 bits	–
	312LP		↑OUTW L	↑L OUTW				4	4		

■ Parameter

Out Specified Bit	OUTW	s	n
Out Specified Long-word Bit	L OUTW	s	n

F022201.VSD

s : Device containing the target bit for output

n : Position of the target bit for output

Out Specified Bit (0 to15)

Out Specified Long-word Bit (0 to 31)

■ Available Devices

Table 2.20.2 Devices Available for Out Specific Bit, Out Specified Long-word Bit Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓	✓*1	✓	✓	✓	✓		Yes	Yes
n	✓	✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓	✓*1	✓	✓	✓	✓	✓	No	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

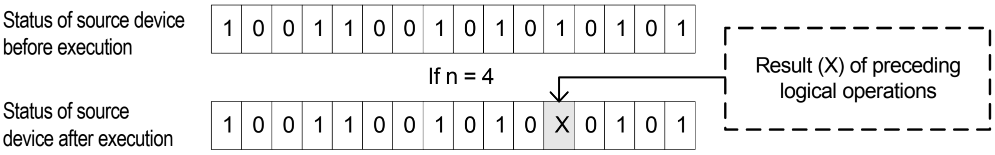
*2: Timer current value (may not be used as a parameter for the Long-word)

*3: Counter current value (may not be used as a parameter for the Long-word)

■ Function

● Out Specified Bit

The OUTW instruction is an output instruction. It outputs the result of the logical operations performed so far to a specified bit of a 16-bit data item. Position 0 represents the least significant bit, and Position 15 represents the most significant bit of the 16-bit data. If the specified bit position exceeds 15 (n > 15), the numerical value (0 to15) of the lower four bits of n is used to determine the bit position.

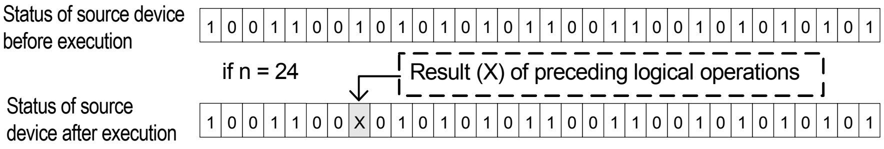

Status of source device before execution
1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1
If n = 4
Result (X) of preceding logical operations
Status of source device after execution
1 0 0 1 1 0 0 1 0 1 0 X 0 1 0 1

F022202.VSD

Figure 2.20.1 Example of an Out Specific Bit Operation

● Out Specified Long-word Bit

The OUTW L instruction is an output instruction. It outputs the result of the logical operations performed so far to a specified bit of a 32-bit data item. Position 0 represents the least significant bit, and Position 31 represents the most significant bit of the 32-bit data. If the specified bit position exceeds 31 (n > 31), the numerical value (0 to31) of the lower five bits of n is used to determine the bit position.

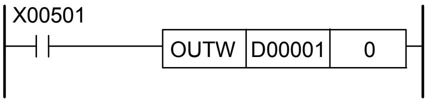

Status of source device before execution
1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1
if n = 24
Result (X) of preceding logical operations
Status of source device after execution
1 0 0 1 1 0 0 X 0 1 0 1 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1

F022203.VSD

Figure 2.20.2 Example of an Out Specific Long-word Bit Operation

■ Programming Example

The following sample code outputs the result of X00501 to bit position 0 of D00001.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	OUTW	D00001	0			

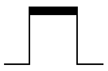
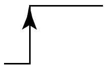
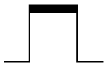
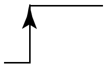
F022105.VSD

Figure 2.20.3 Sample Code for Out Specific Bit

2.21 Set Specified Bit (SETW/SETW L)

F3SP22-0S F3SP28-3S F3SP38-6S | F3SP53-4S F3SP58-6S F3SP59-7S | F3SP66 F3SP67 | F3SP71 F3SP76

Table 2.21.1 Set Specified Bit, Set Specified Long-word Bit

Classi-fication	FUNC No.	Instruc-tion	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Pro-cessing Unit	Carry
					Yes	No		Without Index Modifi-cation	With Index Modifi-cation		
Basic Instruction	313	Set Specified Bit	SETW	SETW	✓	–		3	3	16 bits	–
	313P		↑SETW	↑ SETW				4	4		
	313L	Set Specified Long-word Bit	SETW L	L SETW	✓	–		3	3	32 bits	–
	313LP		↑SETW L	↑L SETW				4	4		

■ Parameter

Set Specified Bit — SETW | s | n

Set Specified Long-word Bit — L SETW | s | n

F022301.VSD

s : Device containing the bit to be set

n : Position of the bit to be set

Set Specified Bit (0 to15)

Set Specified Long-word Bit (0 to 31)

■ Available Devices

Table 2.21.2 Devices Available for Set Specific Bit, Set Specified Long-word Bit

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓	✓*1	✓	✓	✓	✓		Yes	Yes
n	✓	✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓	✓*1	✓	✓	✓	✓	✓	No	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value (may not be used as a parameter for the Long-word)

*3: Counter current value (may not be used as a parameter for the Long-word)

■ Function

● Set Specific Bit

The SETW instruction sets the specified bit of a 16-bit data item to ON. The specified bit remains ON even if the result of the preceding operations becomes OFF. Position 0 represents the least significant bit, and Position 15 represents the most significant bit of the 16-bit data. If the specified bit position exceeds 15 (n > 15), the numerical value (0 to15) of the lower four bits of n is used to determine the bit position.

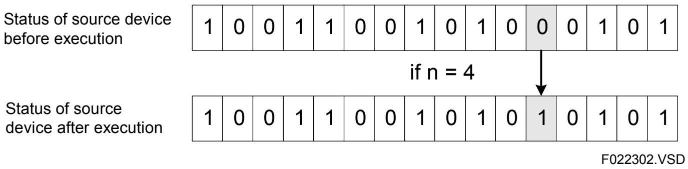


Figure 2.21.1 Example of a Set Specific Bit Operation

● Set Specific Long-word Bit

The SETW L instruction sets the specified bit of a 32-bit data item to ON. The specified bit remains ON even if the result of the preceding operations becomes OFF. Position 0 represents the least significant bit, and Position 31 represents the most significant bit of the 32-bit data. If the specified bit position exceeds 31 (n > 31), the numerical value (0 to 31) of the lower five bits of n is used to determine the bit position.

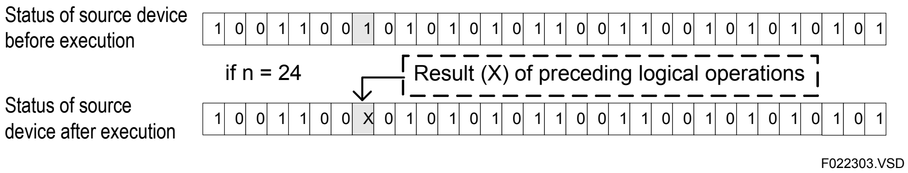
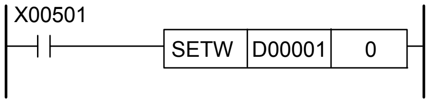

Status of source device before execution
1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1
if n = 24
Result (X) of preceding logical operations
Status of source device after execution
1 0 0 1 1 0 0 X 0 1 0 1 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1
F022303.VSD

Figure 2.21.2 Example of Set Specific Long-word Bit Operation

■ Programming Example

The sample code sets the bit at position 0 of D00001 to on when X00501 turns on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	SETW	D00001	0			

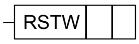
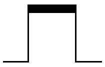
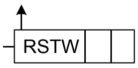
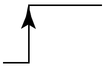
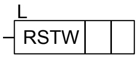
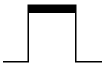
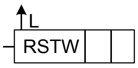
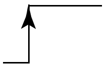
F022304.VSD

Figure 2.21.3 Example of a Set Specific Bit Program

2.22 Reset Specified Bit (RSTW/RSTW L)

F3SP22-0S F3SP28-3S F3SP38-6S | F3SP53-4S F3SP58-6S F3SP59-7S | F3SP66 F3SP67 | F3SP71 F3SP76

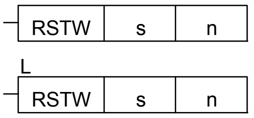
Table 2.22.1 Reset Specified Bit, Reset Specified Long-word Bit

Classi-fication	FUNC No.	Instruc-tion	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count		Pro-cessing Unit	Carry
					Yes	No		Without Index Modifi-cation	With Index Modifi-cation		
Basic Instruction	314	Reset Specified Bit	RSTW		✓	–		3	3	16 bits	–
	314P		↑RSTW					4	4		
	314L	Reset Specified Long-word Bit	RSTW L		✓	–		3	3	32 bits	–
	314LP		↑RSTW L					4	4		

■ Parameter

Reset Specified Bit

Reset Specified Long-word Bit



F022401.VSD

s : Device containing the bit to be reset

n : Position of the bit to be reset

Reset Specified Bit (0 to 15)

Reset Specified Long-word Bit (0 to 31)

■ Available Devices

Table 2.22.2 Devices Available for Reset Specific Bit, Reset Specified Long-word Bit

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓	✓*1	✓	✓	✓	✓		Yes	Yes
n	✓	✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓	✓*1	✓	✓	✓	✓	✓	No	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value (may not be used as a parameter for the Long-word)

*3: Counter current value (may not be used as a parameter for the Long-word)

■ Function

● Reset Specified Bit

The RSTW instruction sets the specified bit of a 16-bit data item to OFF. The specified bit remains OFF even if the result of the preceding operations becomes ON. Position 0 represents the least significant bit, and Position 15 represents the most significant bit of the 16-bit data. If the specified bit position exceeds 15 (n > 15), the numerical value (0 to15) of the lower four bits of n is used to determine the bit position.

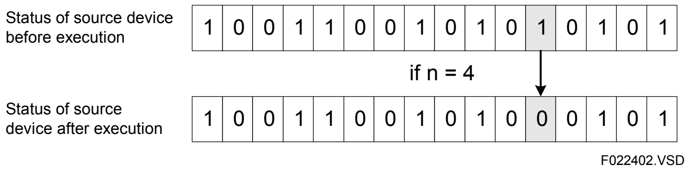


Figure 2.22.1 Example of Reset Specified Bit Operation

● Reset Specified Long-word Bit

The SETW L instruction sets the specified bit of a 32-bit data item to OFF. The specified bit remains OFF even if the result of the preceding operations becomes ON. Position 0 represents the least significant bit, and Position 31 represents the most significant bit of the 32-bit data. If the specified bit position exceeds 31 (n > 31), the numerical value (0 to31) of the lower five bits of n is used to determine the bit position.

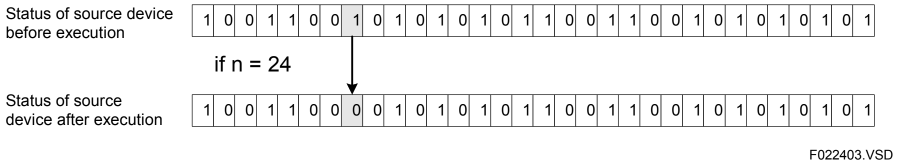
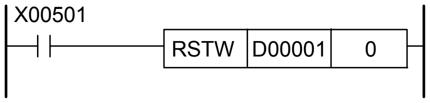


Figure 2.22.2 Example of Reset Specified Long-word Bit Operation

■ Programming Example

The following sample code turns off the bit at Position 0 of D00001 when X00501 turns on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	RSTW	D00001	0			

F022404.VSD

Figure 2.22.3 Example of Reset Specific Bit Program

2.23 End (END)

Table 2.23.1 End

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Basic Instruction	999	End	END	END	–	✓	–	1	–	–

■ Parameter

End

F021901.VSD

■ Function

The End instruction identifies the end of a scan. Any instructions appearing after an End instruction are not executed. To modify the latter part of a program without stopping the current device, place an End instruction before the program fragment to be modified in the program. This way the program will not run beyond the End instruction into the program fragment.

Since any timers are updated in the End processing, the processing time will vary depending on how many timers are used.

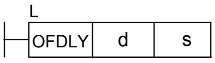
CAUTION

An End processing is automatically generated in all programs. Do not insert another End instruction unless you want to skip execution of the instructions appearing after it.

2.24 Off-Delay (OFDLY)

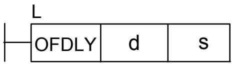
F3SP71
F3SP76

Table 2.24.1 Off-Delay

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Basic Instruction	–	Off-Delay	OFDLY		–	–	–	4	Preset value 32 bit	–

■ Parameter

Off-Delay



F224001.VSD

d : Relay device number

s : Preset value, treated as a long-word data.

- Literal: Set in seconds (s), milliseconds (ms), or microseconds (μs).

- When a device is specified: The current value of the device is loaded as the count value (1 to 134217727). (0 μs to 134 s 217 ms 727 μs)

■ Available Devices

Table 2.24.2 Devices Available for Off-Delay

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d	✓	✓	✓	✓	✓												Yes	Yes
s									✓	✓*1	✓*1	✓*1		✓*1		✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

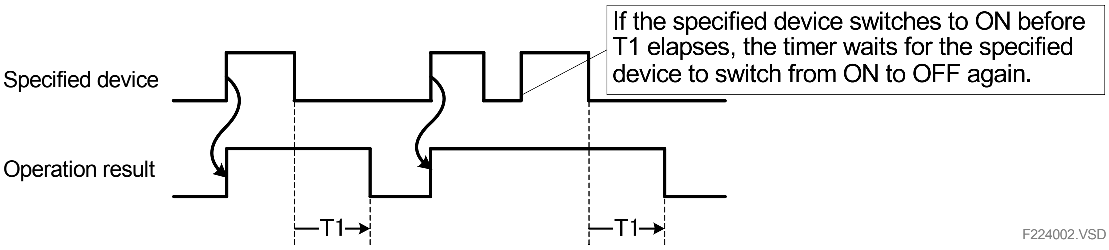
The Off-Delay instruction outputs the off-delay timer operation result for the specified device.

When the specified device switches from OFF to ON, the operation result will be ON.

When the specified device switches from ON to OFF, the preset value is loaded, and then the operation result will be OFF after the time T1 specified by the preset value elapses.

If the specified device switches to ON before the time T1 elapses, the timer waits for the specified device to switch from ON to OFF again.

Unlike the timer instruction, you cannot get the current value. If you want to use the current value, use the timer instruction.



If the specified device switches to ON before T1 elapses, the timer waits for the specified device to switch from ON to OFF again.

F224002.VSD

Figure 2.24.1 Example of an Off-Delay Operation

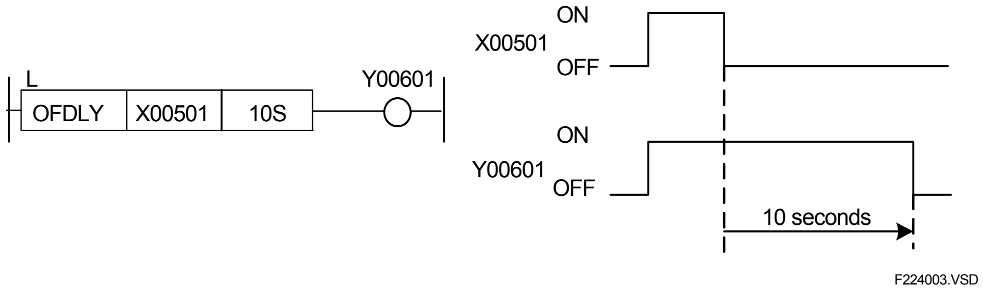
If the preset value is 0, the operation result will switch from ON to OFF when the specified device switches from ON to OFF.

If the preset value exceeds the maximum value 134217727, the lowest-order 27 bits are taken as the preset value.

If the instruction is executed when the interlock condition is OFF, and if the block that contains this instruction is inactive, the operation result will be OFF.

■ Programming Example

The sample code shown below sets Y00601 to OFF 10 seconds after X00501 is OFF.



Line No.	Instruction	Operands				
0001	OFDLY	X00501	10S			
0002	OUT	Y00601				

Figure 2.24.2 Sample Code for Off-Delay

2.25 On-Delay (ONDLY)

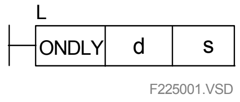
F3SP71
F3SP76

Table 2.25.1 On-Delay

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Basic Instruction	–	On-Delay	ONDLY	L ONDLY d s	–	–	–	4	Preset value 32 bit	–

■ Parameter

On-Delay



d : Relay device number

s : Preset value, treated as a long-word data.

- Literal: Set in seconds (s), milliseconds (ms), or microseconds (μs).

- When a device is specified: The current value of the device is loaded as the count value (1 to 134217727). (0 μs to 134 s 217 ms 727 μs)

■ Available Devices

Table 2.25.2 Devices Available for On-Delay

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d	✓	✓	✓	✓	✓												Yes	Yes
s									✓	✓*1	✓*1	✓*1		✓*1		✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

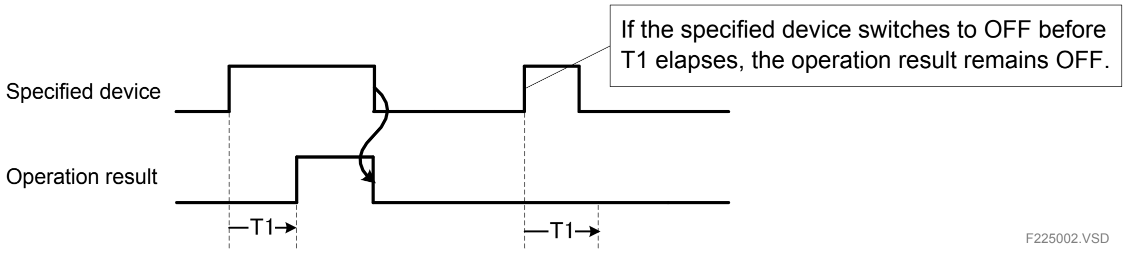
The On-Delay instruction outputs the on-delay timer operation result for the specified device.

When the specified device switches from OFF to ON, the preset value is loaded, and then the operation result will be ON after the time T1 specified by the preset value elapses.

When the specified device switches from ON to OFF, the operation result will be OFF.

If the specified device switches to OFF before the time T1 elapses, the operation result remains OFF.

Unlike the timer instruction, you cannot get the current value. If you want to use the current value, use the timer instruction.



If the specified device switches to OFF before T1 elapses, the operation result remains OFF.
Specified device
Operation result
T1
T1
F225002.VSD

Figure 2.25.1 Example of an On-Delay Operation

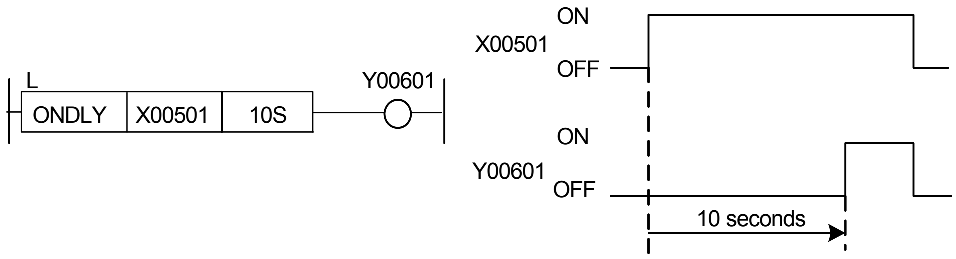
If the preset value is 0, the operation result will switch from OFF to ON when the specified device switches from OFF to ON.

If the preset value exceeds the maximum value 134217727, the lowest-order 27 bits are taken as the preset value.

If the instruction is executed when the interlock condition is OFF, and if the block that contains this instruction is inactive, the operation result will be OFF.

■ Programming Example

The sample code shown below sets Y00601 to ON 10 seconds after X00501 is set to ON.



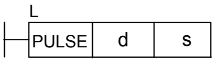
Line No.	Instruction	Operands				
0001	ONDLY	X00501	10S			
0002	OUT	Y00601				

Figure 2.25.2 Sample Code for On-Delay

2.26 Pulse (PULSE)

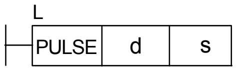
F3SP71
F3SP76

Table 2.26.1 Pulse

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Basic Instruction	—	Pulse	PULSE		–	–	–	4	Preset value 32 bit	—

■ Parameter

Pulse



F226001.VSD

d : Relay device number

s : Preset value, treated as a long-word data.

- Literal: Set in seconds (s), milliseconds (ms), or microseconds (μs).

- When a device is specified: The current value of the device is loaded as the count value (1 to 134217727). (0 μs to 134 s 217 ms 727 μs)

■ Available Devices

Table 2.26.2 Devices Available for Pulse

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d	✓	✓	✓	✓	✓												Yes	Yes
s									✓	✓*1	✓*1	✓*1		✓*1		✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

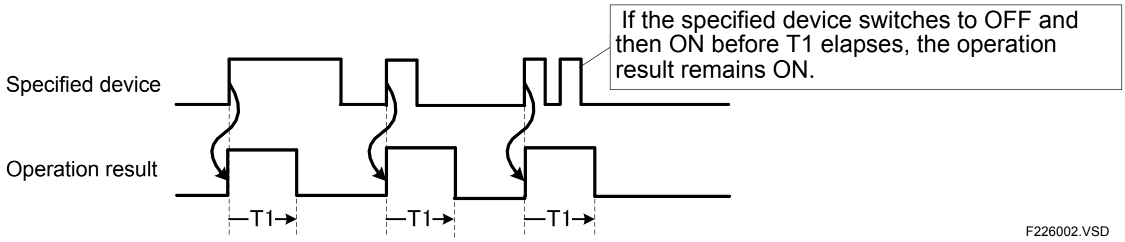
■ Function

The Pulse instruction outputs the pulse timer operation result for the specified device.

When the specified device switches from OFF to ON, the preset value is loaded, and then the operation result will be ON. After the time T1 specified by the preset value elapses, the operation result turns OFF.

If the specified device turns OFF or it turns OFF and then ON again before the time T1 elapses, the operation result doesn't turn OFF until the time T1 elapses after the specified device initially turns ON.

Unlike the timer instruction, you cannot get the current value. If you want to use the current value, use the timer instruction.


If the specified device switches to OFF and then ON before T1 elapses, the operation result remains ON.
Specified device
Operation result
T1
T1
T1

F226002.VSD

Figure 2.26.1 Example of a Pulse Operation

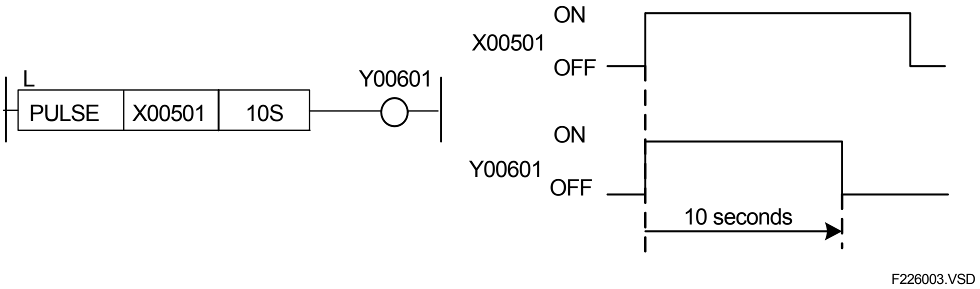
If the preset value is 0, the operation result remains OFF.

If the preset value exceeds the maximum value 134217727, the lowest-order 27 bits are taken as the preset value.

If the instruction is executed when the interlock condition is OFF, and if the block that contains this instruction is inactive, the operation result will be OFF.

■ Programming Example

The sample code shown below sets Y00601 to ON for 10 seconds after X00501 is set to ON.



Line No.	Instruction	Operands				
0001	PULSE	X00501	10S			
0002	OUT	Y00601				

Figure 2.26.2 Sample Code for Pulse

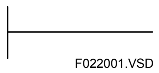
2.27 Nop (NOP)

Table 2.27.1 NOP

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Basic Instruction	00	Nop	NOP	├──	–	✓	–	1	–	–

■ Parameter

Nop



F022001.VSD

■ Function

The Nop instruction does nothing. A Nop instruction takes one clock of time to execute.

3. Application Instructions

This chapter describes the application instructions for the FA-M3 CPU modules with sample programs. Be sure to refer to this chapter before programming.

3.1 Application Instruction

Chapter 3 explains how to use the application instructions for the FA-M3 CPU modules. The notational conventions for the application instruction descriptions are summarized below.

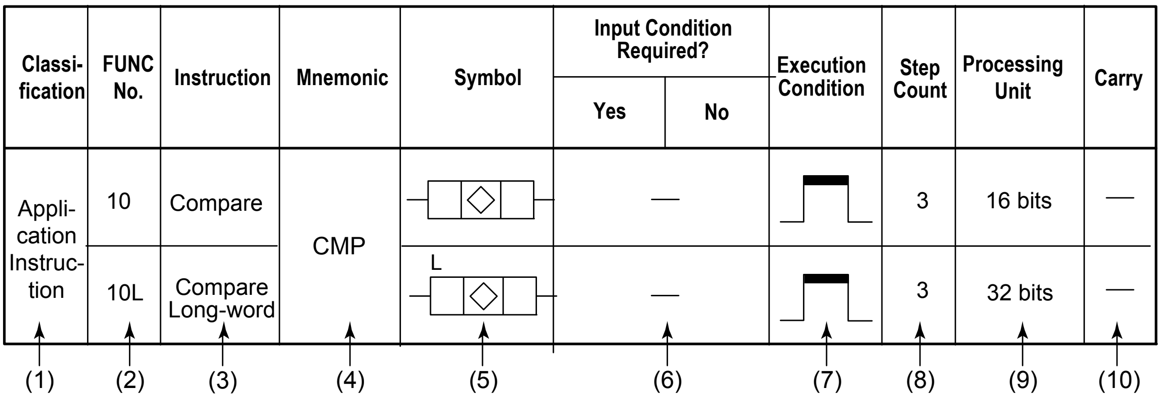
TIP

Application instructions include arithmetic, string processing, and other advanced function instructions. Most of them are 16-bit or 32-bit operations.

■ Quick Function Reference Chart

Each application instruction description begins with a quick function reference chart which looks like as shown below.

Table 3.1.1 How to Interpret the Application Instruction Quick Reference Chart



Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	10	Compare	CMP		—			3	16 bits	—
	10L	Compare Long-word		L	—			3	32 bits	—

(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)
(10)

F030101.VSD

(1) Classification
The instructions described in this chapter are application instructions and continuous type application instructions.

(2) FUNC No.
Indicates the function number of the instruction. An instruction that is identified by a function number followed by a letter P is a differential type instruction which is executed only once when its input is turned on.

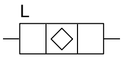
(3) Instruction
Indicates the name of the instruction.

(4) Mnemonic
Indicates the mnemonic representation of the instruction, as used in WideField3, WideField2, WideField and Ladder Diagram Program Support M3.

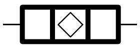
(5) Symbol
Indicates the graphical representation of the instruction, as used in WideField3, WideField2, WideField, and Ladder Diagram Program Support M3.

CAUTION

The parameters referred to in this document are the same as those used for WideField3, WideField2 and WideField, but differ from those used for Ladder Diagram Support Program M3. The difference between them only applies to long-word data instructions as follows:

Example of a symbol in FA-M3 Programming Tool, WideField3, WideField2 and WideField: 

F030102.VSD

and its equivalent for Ladder Diagram Support Program M3: 
F030103.VSD

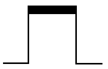
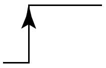
(6) Input Condition Required?

Indicates whether a contact needs to be specified as input condition. A check mark in the "Yes" column indicates that a contact must always be specified as input condition. A check mark in the "No" column indicates that no contact must be specified as input condition. An instruction for which a hyphen in the "Input Condition Required?" column spans over the "Yes" and "No" columns may or may not have a contact as input condition.

(7) Execution Condition

Contains the execution condition for the instruction which requires input condition.

Table 3.1.2 Execution Condition Symbols

Symbol	Description
	Represents an execute-while-ON instruction. The instruction continues to execute while the previous condition is ON. If execution of the instruction is suppressed if the previous condition is OFF
	Represents an execute-at-On instruction. The instruction is executed only once when the state of its precondition switches from OFF to ON. Subsequently the instruction is not executed even when its precondition is ON.
—	Represents an always-execute instruction. The instruction is executed regardless of whether its precondition is ON or OFF.

(8) Step Count

Indicates the number of steps required to execute the instruction. The step count varies according to the execution conditions.

(9) Processing Unit

Indicates the processing unit of the instruction. Instructions whose processing unit is 1 are intended for relays. Instructions whose processing unit is 16-, 32-, or 64-bits are intended for registers. 16 or 32 bits of relays, when combined, may be handled as data. Note that Input/output relays for which no Input/Output Setup is made are handled as being represented by binary data. Input/Output Setup is made using WideField3, WideField2, WideField, or Ladder Diagram Support Program M3.

(10) Carry

When an instruction identified by a check mark in the Input Condition column is executed, the state of the special relay (M188) may be changed to represent the carry state. See the individual instruction descriptions.

TIP

The input condition column indicates whether an input condition instruction such as Load (LD) or Compare (CMP) must be specified when an instruction is used.

TIP

The execution condition column indicates what operation result of the input condition (such as Load (LD) instruction) will trigger the execution of an instruction. There are four execution conditions corresponding to different operation results of the input condition.

■ Parameter

The parameters column indicates the parameters of an instruction. The symbols in this column have the following meanings:

s	: Identifies the source.
s1	: Identifies the first source of two or more sources.
s2	: Identifies the second source of two or more sources.
d	: Identifies the destination.
d1	: Identifies the first destination of two or more destinations.
d2	: Identifies the second destination of two or more destinations.
n	: Represents a numeric value or a device that represents a numeric value.
n1	: Identifies the first of two or more numeric values or devices representing a numeric value.
n2	: Identifies the second of two or more numeric values or devices representing a numeric value.
t	: Identifies the first device of the table.

Note:		
Note:	Source:	Data before the operation is performed
	Destination:	Data after the operation is performed

The relationship between a parameter of an instruction and available devices is given in the "■ Available Devices" table for that instruction.

■ Available Devices

Check marks in the available devices table indicate that the corresponding device is available. For instructions with two or more parameters, available devices are indicated for each of the parameters.

■ Function

Describes the function of the instruction.

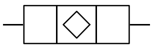
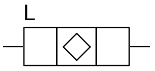
■ Programming Example

Shows sample codes which contain the instruction.

3.2 Comparison Instructions

3.2.1 Compare (CMP), Compare Long-word (CMP L)

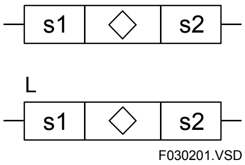
Table 3.2.1 Compare, Compare Long-word

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	10	Compare	CMP		—		—	3	16 bits	—
	10L	Compare Long-word	CMP L		—		—	3	32 bits	—

■ Parameter

Compare

Compare Long-word



s1, s2 : Comparison data or device numbers of the first devices to be compared as data
< > : Comparison operator (=, <>, >, >=, <, or <=)

■ Available Devices

Table 3.2.2 Devices Available for the Compare and Compare Long Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s1	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value (may not be used as a parameter for the Long-word)

*3: Counter current value (may not be used as a parameter for the Long-word)

■ Function

The Compare and Compare Long-word instructions perform a 16-bit and 32-bit comparison operation, respectively, and output the result as contact a. The possible results of comparison operations that can be performed by the Compare and Compare Long-word instructions are summarized below.

Table 3.2.3 Operators and Execution Results

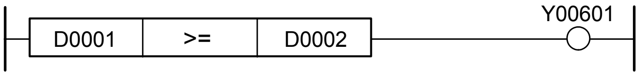
Operator (◇)	Condition and Execution Result			
	Condition	Execution Result	Condition	Execution Result
=	s1=s2	ON	s1≠s2	OFF
<>	s1≠s2		s1=s2	
>	s1>s2		s1≤s2	
>=	s1≥s2		s1<s2	
<	s1<s2		s1≥s2	
=<	s1≤s2		s1>s2	

The instructions can perform comparison operations on either binary or BCD operands, or both types of operands at the same time. They cannot compare 16-bit data with 32-bit data directly.

■ Programming Example

Compare, Long-word Compare

The sample code shown below sets Y00601 to ON if D0001 is greater than D0002.



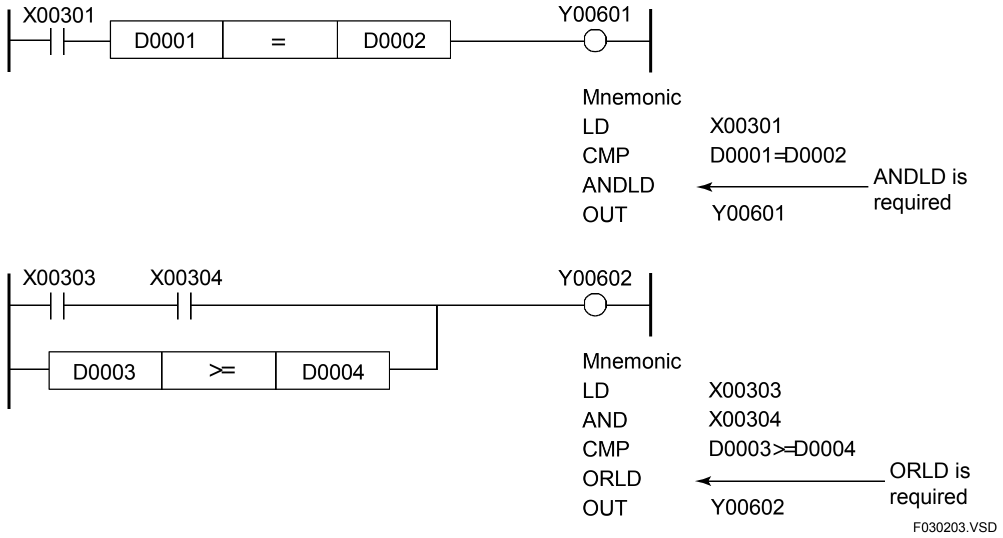
Line No.	Instruction	Operands				
0001	CMP	D0001	>=	D0002		
0002	OUT	Y00601				

F030202.VSD

Figure 3.2.1 Sample Code for the Compare Instruction

CAUTION

A Compare instruction, when represented by mnemonic, is handled as a single circuit element. Consequently, And Load and Or Load instructions need to be inserted to represent the following sample circuits by mnemonic:



```
Mnemonic
LD        X00301
CMP       D0001=D0002
ANDLD
OUT       Y00601
```

```
Mnemonic
LD        X00303
AND       X00304
CMP       D0003>=D0004
ORLD
OUT       Y00602
```

Figure 3.2.2 Example for the Compare Instruction in Mnemonic Representation

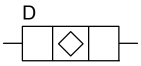
SEE ALSO

For details, see the descriptions for the And Load and Or Load instructions.

3.2.2 Compare Double Long-word (CMP D)

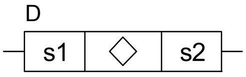
F3SP71
F3SP76

Table 3.2.4 Compare Long-word

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	10D	Compare Double Long-word	CMP D		—		—	5	64 bits	—

■ Parameter

Compare Double Long-word



F322001.VSD

s1, s2 : Comparison data or device numbers of the first devices to be compared as data
< > : Comparison operator (=, <>, >, >=, <, or <=)

■ Available Devices

Table 3.2.5 Devices Available for the Compare Double Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s1									✓	✓*1	✓*1	✓		✓		✓	Yes	Yes
s2									✓	✓*1	✓*1	✓		✓		✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Compare Double Long-word instruction performs a 64-bit comparison operation and outputs the result as contact a. The possible results of comparison operations that can be performed by the Compare Double Long-word instruction are summarized below.

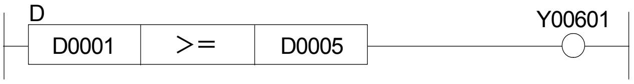
Table 3.2.6 Operators and Execution Results

Operator (◇)	Condition and Execution Result			
	Condition	Execution Result	Condition	Execution Result
=	s1=s2	ON	s1≠s2	OFF
<>	s1≠s2		s1=s2	
>	s1>s2		s1≤s2	
>=	s1≥s2		s1<s2	
<	s1<s2		s1≥s2	
=<	s1≤s2		s1>s2	

■ Programming Example

Compare Double Long-word

The sample code shown below sets Y00601 to ON if D0001 is greater than or equal to D0005.



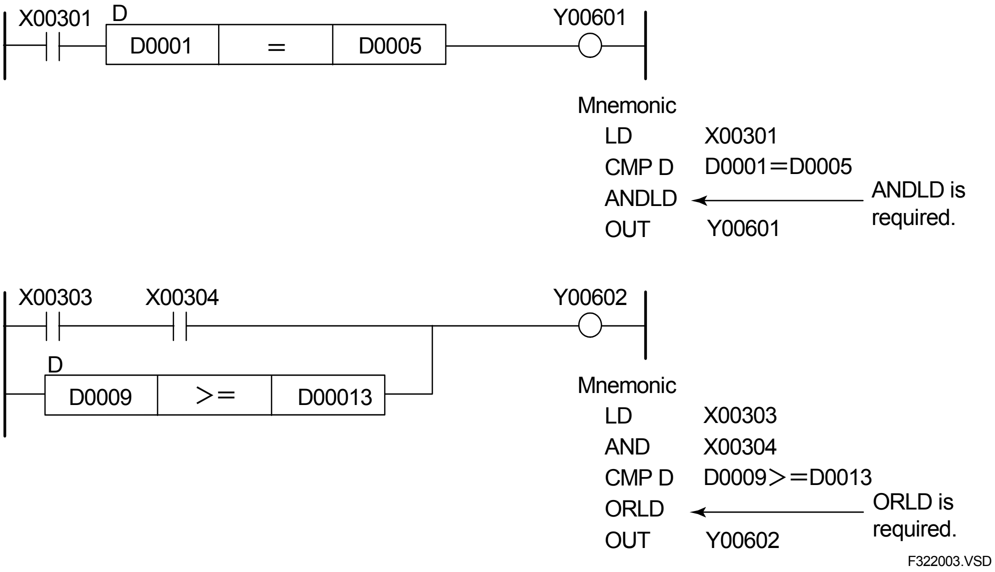
Line No.	Instruction	Operands				
0001	CMP D	D0001	>=	D0005		
0002	OUT	Y00601				

F322002.VSD

Figure 3.2.3 Sample Code for the Compare Instruction

CAUTION

A Compare instruction, when represented by mnemonic, is handled as a single circuit element. Consequently, And Load and Or Load instructions need to be inserted to represent the following sample circuits by mnemonic:



```
Mnemonic
  LD        X00301
  CMP D     D0001=D0005
  ANDLD  <--------------- ANDLD is
  OUT       Y00601        required.
```

```
Mnemonic
  LD        X00303
  AND       X00304
  CMP D     D0009>=D0013
  ORLD   <--------------- ORLD is
  OUT       Y00602        required.
```

F322003.VSD

Figure 3.2.4 Example for the Compare Instruction in Mnemonic Representation

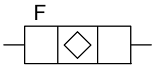
SEE ALSO

For details, see the descriptions for the And Load and Or Load instructions.

3.2.3 Compare Float (FCMP)

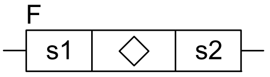
F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.2.7 Compare, Compare Float

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	904	Compare Float	FCMP		—		—	4	32 bits	—

■ Parameter

Compare Float



F030204.VSD

s1, s2 : Comparison data or device numbers of the first devices to be compared as data
◇ : Comparison operator (=, <>, >, >=, <, or <=)
Both s1 and s2 are represented in the IEEE single-precision floating-point format (32 bits)

■ Available Devices

Table 3.2.8 Devices Available for the Compare Float Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s1	✓	✓	✓	✓	✓	✓			✓	✓[*1]	✓[*1]	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓			✓	✓[*1]	✓[*1]	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Compare Float instruction compares two single-precision floating-point (32 bits) data and outputs the result as contact a. The single-precision floating-point data must be represented in the IEEE format.

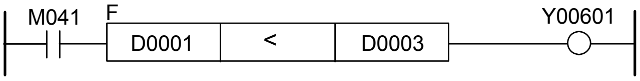
The possible results of comparison operations that can be performed by the FCMP instruction are summarized in the following table.

Table 3.2.9 Operators and Execution Results

Operator (◇)	Condition and Execution Result			
	Condition	Execution Result	Condition	Execution Result
=	s1=s2	ON	s1≠s2	OFF
<>	s1≠s2		s1=s2	
>	s1>s2		s1≤s2	
>=	s1≥s2		s1<s2	
<	s1<s2		s1≥s2	
=<	s1≤s2		s1>s2	

■ Programming Example

The sample code shown below sets Y00601 to ON if M041 is ON and floating-point data D0001 and D0002 are smaller than floating-point data D0003 and D0004.



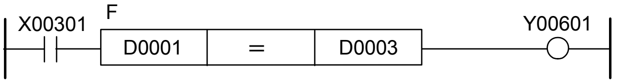
Line No.	Instruction	Operands				
0001	LD	M041				
0002	FCMP	D0001	<	D0003		
0003	ANDLD					
0004	OUT	Y00601				

F030205.VSD

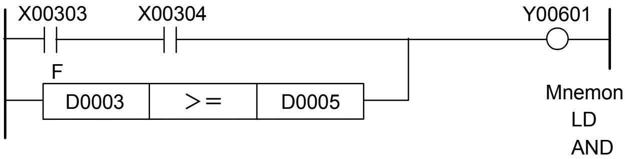
Figure 3.2.5 Sample Code for the Compare Float Instruction

CAUTION

A Compare Float instruction, when represented by mnemonic, is handled as a single circuit element. Consequently, And Load and Or Load need to be inserted to represent the following sample circuits by mnemonic:



```
Mnemonic
  LD       X00301
  FCMP     D0001=D0003      ANDLD is
  ANDLD  ←───────────────   required
  OUT      Y00601
```



```
Mnemonic
  LD       X00303
  AND      X00304
  FCMP     D0003>=D0005     ORLD is
  ORLD   ←───────────────   required
  OUT      Y00602
```

F030206.VSD

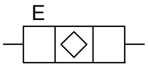
Figure 3.2.6 Sample Code for the Compare Float Instruction in Mnemonic Representation

SEE ALSO

For details, see the descriptions for the And Load and Or Load instructions.

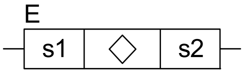
3.2.4 Compare Double-precision Float (FCMP E)

Table 3.2.10 Compare Double-precision Float

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	904E	Compare Double-precision Float	FCMP E		—		—	5	64 bits	—

■ Parameter

Compare Double-precision Float



F322004.VSD

s1, s2 : Comparison data or device numbers of the first devices to be compared as data
◇ : Comparison operator (=, <>, >, >=, <, or <=)
Both s1 and s2 are represented in the IEEE double-precision floating-point format (64 bits)

■ Available Devices

Table 3.2.11 Devices Available for the Compare Double-precision Float Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s1									✓	✓[*1]	✓[*1]	✓		✓		✓	Yes	Yes
s2									✓	✓[*1]	✓[*1]	✓		✓		✓	Yes	Yes

[*1]: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Compare Double-precision Float instruction compares two double-precision floating-point (64 bits) data and outputs the result as contact a. The double-precision floating-point data must be represented in the IEEE format.

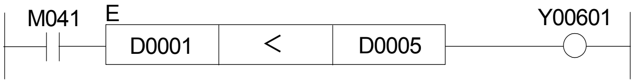
The possible results of comparison operations that can be performed by the FCMP E instruction are summarized in the following table.

Table 3.2.12 Operators and Execution Results

Operator (◇)	Condition and Execution Result			
	Condition	Execution Result	Condition	Execution Result
=	s1=s2	ON	s1≠s2	OFF
<>	s1≠s2		s1=s2	
>	s1>s2		s1≤s2	
>=	s1≥s2		s1<s2	
<	s1<s2		s1≥s2	
=<	s1≤s2		s1>s2	

■ Programming Example

The sample code shown below sets Y00601 to ON if M041 is ON and double-precision floating-point data D0001, D0002, D0003, and D0004 are smaller than double-precision floating-point data D0005, D0006, D0007, and D0008.



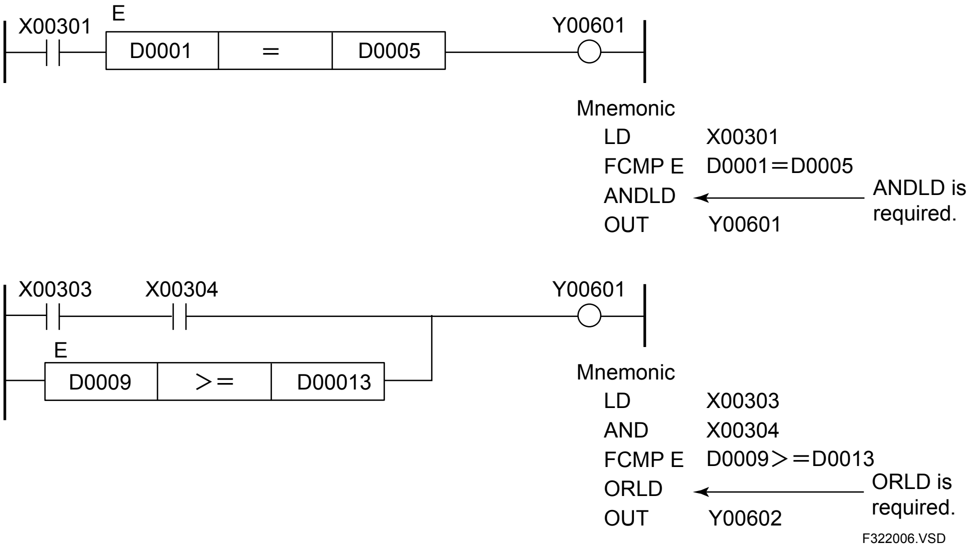
Line No.	Instruction	Operands				
0001	LD	M041				
0002	FCMP E	D0001	<	D0005		
0003	ANDLD					
0004	OUT	Y00601				

F322005.VSD

Figure 3.2.7 Sample Code for the Compare Double-precision Float Instruction

CAUTION

A Compare Double-precision Float instruction, when represented by mnemonic, is handled as a single circuit element. Consequently, And Load and Or Load need to be inserted to represent the following sample circuits by mnemonic:



```
Mnemonic
  LD        X00301
  FCMP E    D0001=D0005
  ANDLD  ←────────────── ANDLD is
                         required.
  OUT       Y00601
```

```
Mnemonic
  LD        X00303
  AND       X00304
  FCMP E    D0009>=D0013
  ORLD   ←────────────── ORLD is
                         required.
  OUT       Y00602
```

F322006.VSD

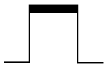
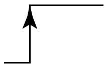
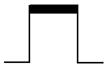
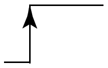
Figure 3.2.8 Sample Code for the Compare Double-precision Float Instruction in Mnemonic Representation

SEE ALSO

For details, see the descriptions for the And Load and Or Load instructions.

3.2.5 Table Compare (BCMP), Table Compare Long-word (BCMP L)

Table 3.2.13 Table Compare, Table Compare Long-word

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	111	Table Compare	BCMP	BCMP	✓	—		5	16 bit	—
	111P		↑BCMP	↑ BCMP				6		
	111L	Table Compare Long-word	BCMP L	L BCMP	✓	—		5	32 bit	—
	111LP		↑BCMP L	↑L BCMP				6		

■ Parameter

Table Compare

BCMP	s	t	n	d

Table Compare Long-word

L

BCMP	s	t	n	d

F030207.VSD

s : Device number of the first device storing the comparison data
t : Device number of the first device storing the upper-/lower-limit table to be searched
n : Maximum row number [*1] (0-999; row numbers start from 0)
d : Device number[*1] of the first device for storing the comparison result

*1: n and d are handled as words even for the 32-bit (long word) instruction.

■ Available Devices

Table 3.2.14 Devices Available for the Table Compare and Table Compare Long-word Instructions

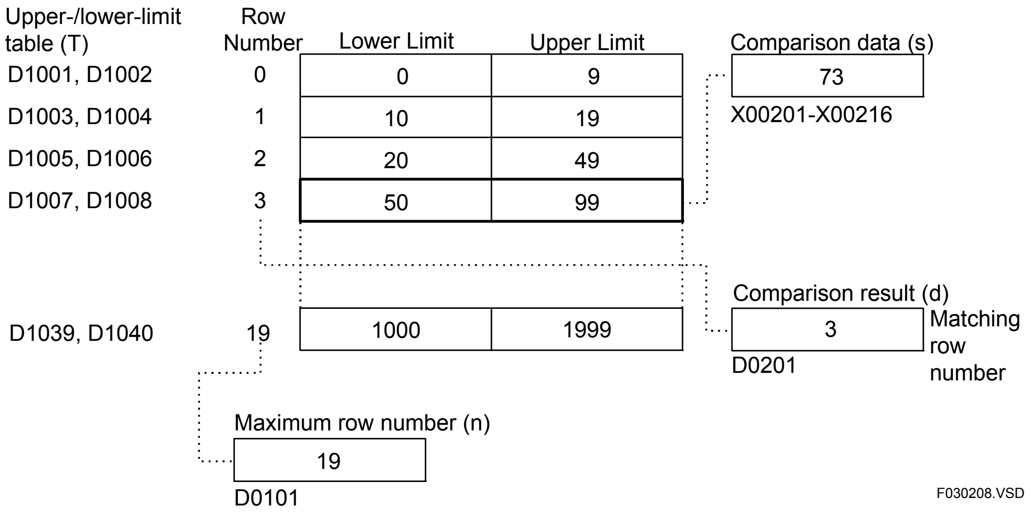
Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
t	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓		Yes	Yes
n	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value (may not be used as a parameter for the Long-word)
*3: Counter current value (may not be used as a parameter for the Long-word)

■ Function

The Table Compare and Table Compare Long-word instructions search an upper-/lower-limit table starting at the device designated by t and containing rows 0,1, 2,....n (n=maximum row No.) and load d with the row number that matches the comparison data s. If no match is found, -1 ($FFFF) is loaded in d.

● Upper-/Lower-limit table



Upper-/lower-limit table (T)	Row Number	Lower Limit	Upper Limit
D1001, D1002	0	0	9
D1003, D1004	1	10	19
D1005, D1006	2	20	49
D1007, D1008	3	50	99
D1039, D1040	19	1000	1999

Figure 3.2.9 Comparison Table

For the Table Compare Long-word instruction, the upper-/lower-limit table and comparison data are handled as long words and the maximum row number and comparison result are handled as words. Each row of the upper-lower-limit table contains 2 words or 2 long words that specify a pair of upper- and lower-limit values. The device number designating the lower-limit value must always be smaller than the device number designating the upper-limit value.

Word

	Lower limit	Upper limit
✓	D0001	D0002
×	D0002	D0001

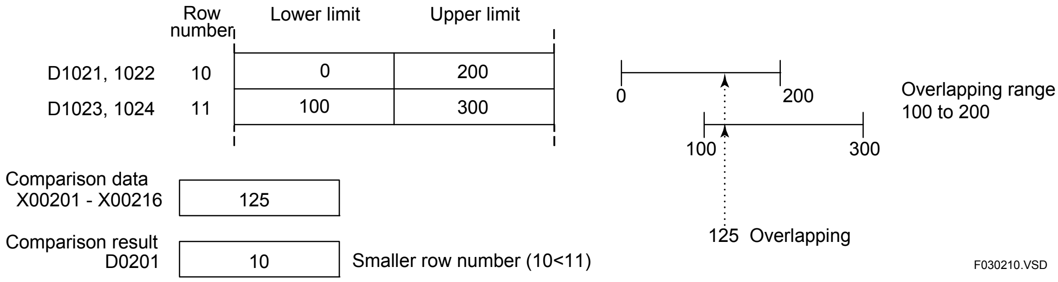
Long word

	Lower limit	Upper limit
✓	D0001, 2	D0003, 4
×	D0003, 4	D0001, 2

F030209.VSD

Figure 3.2.10 Upper-/Lower-limit Table

The upper-/lower-limit table must be specified with no overlapping ranges. If there is an overlap and the comparison data falls within the overlap, the smaller row number is stored as the comparison result.

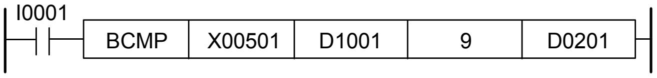


	Row number	Lower limit	Upper limit
D1021, 1022	10	0	200
D1023, 1024	11	100	300

Figure 3.2.11 Table Contains Overlapping Ranges

■ Programming Example

The sample code shown below searches an upper-/lower-limit table, starting at D1001 and containing 10 rows, for a row matching 16-bit comparison data starting at X00501 and loads the number of the matching row into D0201.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	BCMP	X00501	D1001	9	D0201	

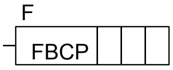
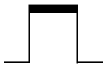
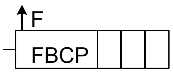
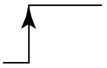
F030211.VSD

Figure 3.2.12 Sample Code for the Table Compare Instruction

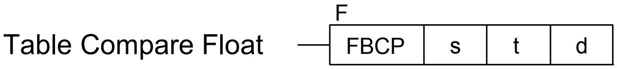
3.2.6 Table Compare Float (FBCP)

F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.2.15 Table Compare Float

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	919	Table Compare Float	FBCP		✓	—		5	32 bit	—
	919P		↑FBCP					6		

■ Parameter



F030212.VSD

s : Data to be compared or device number of the first device storing the data to be compared
t : Device number of the first device storing the upper-/lower-limit table (the first word contains an integer from 0 to 999 specifying the number of rows)
d : Device number of the first device for storing the comparison result (1-word integer)
Tables s and t must be represented in the IEEE single-precision floating-point format (32 bits).

■ Available Devices

Table 3.2.16 Devices Available for the Table Compare Float Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓			✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
t	✓	✓	✓	✓	✓	✓			✓	✓*1	✓*1	✓	✓	✓	✓		Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value
*3: Counter current value

■ Function

The Table Compare Float instruction searches an upper-/lower-limit table (the first word contains the number of rows) starting at the device designated by t and loads d with the row number that matches the comparison data s. If no match is found, -1 $(FFFF) is loaded in d.

● Upper-/Lower-limit table

The first word of the upper-/lower-limit table must be loaded with the maximum row number (row numbers start from 0) and the second and subsequent words with pairs of upper-/lower-limit values.

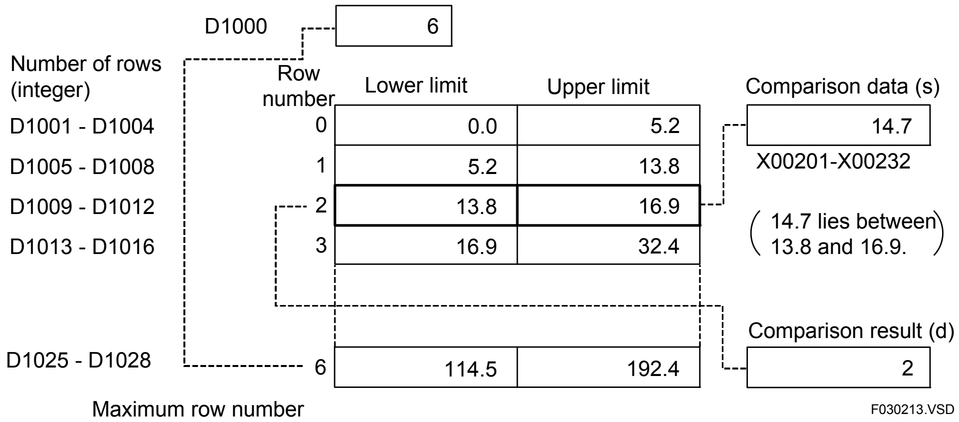
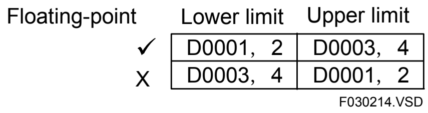


Figure 3.2.13 Comparison Table

Each row of the upper-lower-limit table contains 2 long words that specify a pair of upper- and lower-limit values. The device number designating the lower-limit value must always be smaller than the device number designating the upper-limit value.

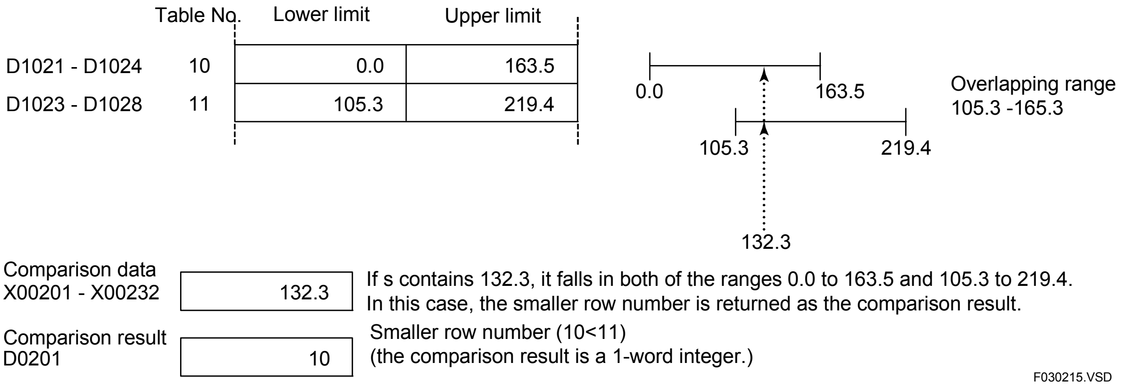


Floating-point	Lower limit	Upper limit
✓	D0001, 2	D0003, 4
X	D0003, 4	D0001, 2

F030214.VSD

Figure 3.2.14 Upper-/lower-limit Table

The upper-/lower-limit table must be specified with no overlapping ranges. If there is an overlap and the comparison data falls within the overlap, the smaller row number is stored as the comparison result.



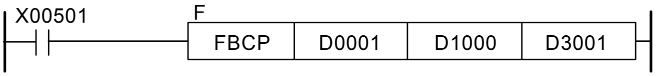
If s contains 132.3, it falls in both of the ranges 0.0 to 163.5 and 105.3 to 219.4. In this case, the smaller row number is returned as the comparison result.

Smaller row number (10<11)
(the comparison result is a 1-word integer.)

Figure 3.2.15 Table Contains Overlapping Ranges

■ Programming Example

The sample code shown below compares real (single-precision floating point) comparison data against an upper-/lower-limit table, starting at D1001 and containing 10 rows (the number of rows is stored in D1000), and loads the matching row number into D3001 if X00501 is ON.



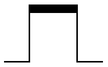
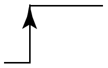
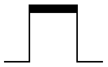
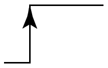
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FBCP	D0001	D1000	D3001		

F030216.VSD

Figure 3.2.16 Sample Code for the Table Compare Float Instruction

3.2.7 Table Search (TSRCH), Long-word Table Search (TSRCH L)

Table 3.2.17 Table Search, Long-word Table Search

Classi-fication	FUN No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	112	Table Search	TSRCH	TSRCH	✓	—		5	16 bit	—
	112P		↑TSRCH	↑ TSRCH				6		
	112L	Long-word Table Search	TSRCH L	L TSRCH	✓	—		5	32 bit	—
	112LP		↑TSRCH L	↑L TSRCH				6		

■ Parameter

Table Search	TSRCH	s	t	n	d
Long-word Table Search	L TSRCH	s	t	n	d

F030217.VSD

s : Device number of the first device storing the data to search for
t : Device number of the first device storing the table to be searched
n : Maximum row number [*1] (0-999; row numbers start from 0)
d : Device number of the first device for storing the search result [*1]

*1: n and d are handled as words even for the 32-bit (long word) instruction.

■ Available Devices

Table 3.2.18 Devices Available for the Table Search and Long-word Table Search Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
t	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓		Yes	Yes
n	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value (may not be used as a parameter for the Long-word)

*3: Counter current value (may not be used as a parameter for the Long-word)

■ Function

The Table Search or Table Search Long instruction searches the table, beginning with the device designated by t and containing a maximum of (n+1) rows, for search data s and if it finds a match, loads d with the matching row number. d is loaded with -1 ($FFFF) if no match is found.

● Search table

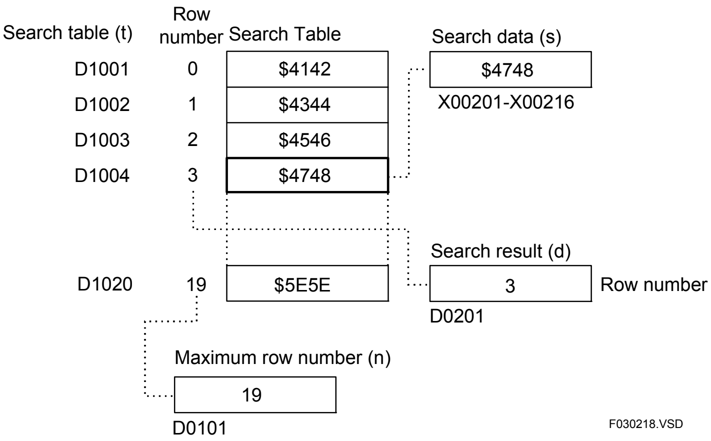


Figure 3.2.17 Search Table

In long-word table search, the search table and search data are handled as long words and the maximum row number and search result are handled as words. The values in the search table must be specified so that there are no duplicates. If there are duplicates and they match the search data, the smaller row number is loaded as the search result.

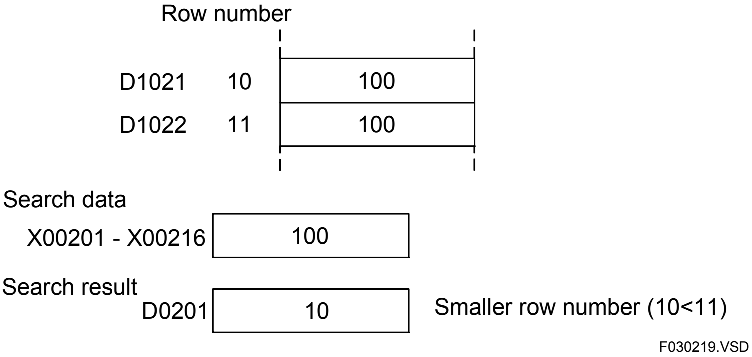
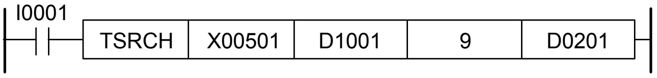


Figure 3.2.18 Table Contains Duplicate Values

■ Programming Example

The sample code shown below searches a search table containing ten 16-bit data items starting at D1001 for 16 bits of search data starting at X00501 and loads the row number of the matching data item into D0201.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	TSRCH	X00501	D1001	9	D0201	

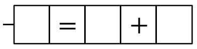
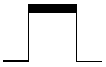
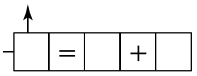
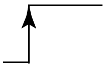
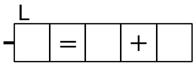
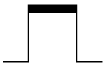
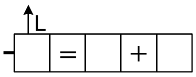
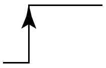
F030220.VSD

Figure 3.2.19 Example of a Table Search Program

3.3 Arithmetic Instructions

3.3.1 Add (CAL), Add Long-word (CAL L)

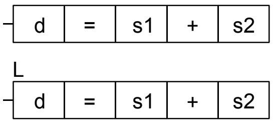
Table 3.3.1 Add, Add Long-word

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	20	Add	CAL		✓	—		4	16 bit	—
	20P		↑CAL					5		
	20L	Add Long-word	CAL L		✓	—		4	32 bit	—
	20LP		↑CAL L					5		

■ Parameter

Add

Add Long-word



F030301.VSD

d : Device number of the first device for storing the execution result
+ : Addition operator
s1, s2 : Data to be added or device numbers of the first devices to be added as data.

■ Available Devices

Table 3.3.2 Devices Available for the Add and Add Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
s1	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value (may not be used as a parameter for the Long-word)
*3: Counter current value (may not be used as a parameter for the Long-word)

■ Function

The Add and Add Long-word instructions perform an addition on 16- and 32-bit data, respectively, and place the result on the specified devices.

Use the Add instruction to add 16-bit data and the Add Long-word instruction to add 32-bit data. Neither Add nor Add Long-word instructions can perform an addition on a mixture of 16- and 32-bit data.

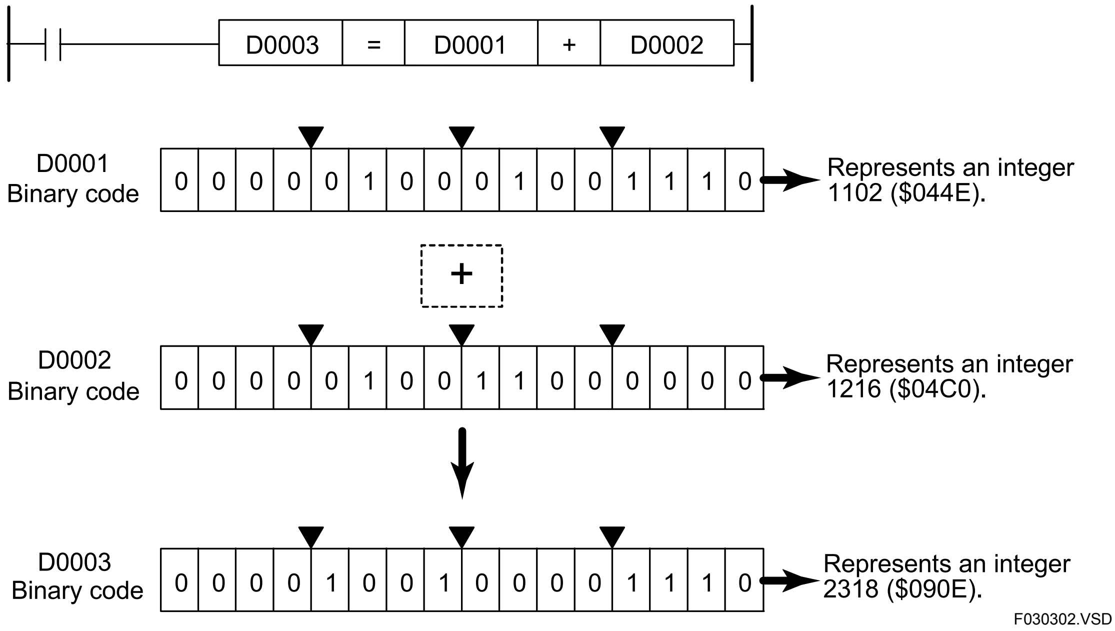
The numbers of bits in the execution results obtained through the Add and Add Long-word instructions are summarized in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.3.3 Numbers of Bits Resulting from of Additions

Specification Item	Instruction	
	Add (1-word instruction)	Add long-word (2-word instruction)
Number of bits in execution result	16bit	32 bits
Device where the execution result is placed	d	d+1, d

The operands on which an operation is to be performed can be either of binary, BCD, or a mixture of both types.

● Example of an addition


D0003
=
D0001
+
D0002
D0001
Binary code
Represents an integer 1102 ($044E).
+
D0002
Binary code
Represents an integer 1216 ($04C0).
D0003
Binary code
Represents an integer 2318 ($090E).
F030302.VSD

D0001: 0 0 0 0 0 1 0 0 0 1 0 0 1 1 1 0

D0002: 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0

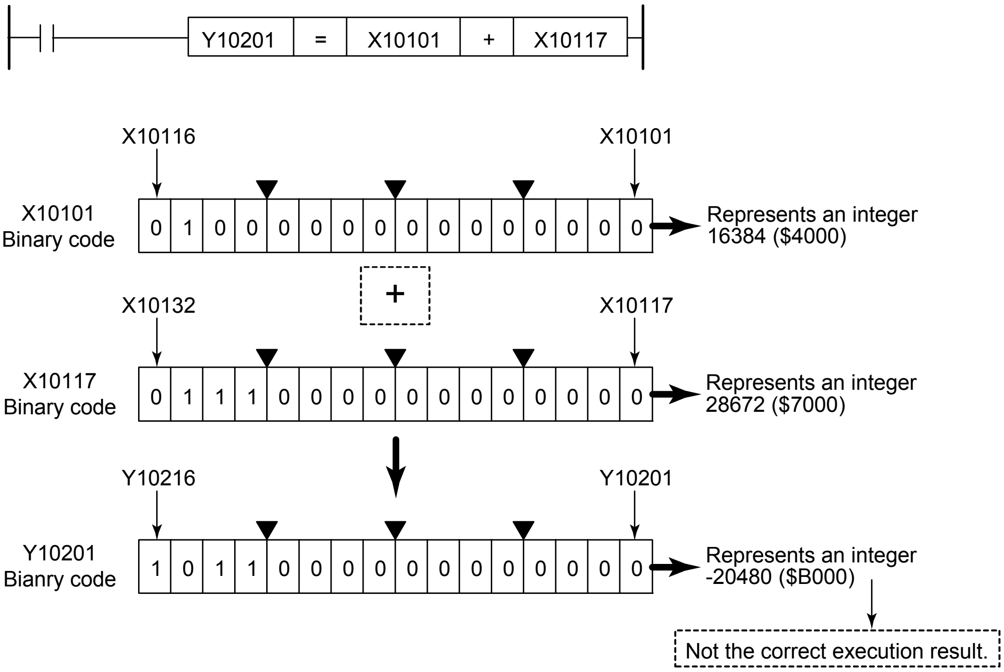
D0003: 0 0 0 0 1 0 0 1 0 0 0 0 1 1 1 0

Figure 3.3.1 Example of an Addition

The carry is not set to ON even if the execution result exceeds the valid value range of the data. The Add and Add Long-word instructions must be executed so that their execution result does not exceed the value range of the respective data type. If the execution result exceeds the value range of the data type, the destination device is loaded with a value but the value does not represent the correct execution result.

No arithmetic operation is executed if the operands of addition (s1 and s2) are defined in BCD code and their values exceed the valid value range of the BCD code. In this case, the value in d remains unchanged.

● Example of a calculation in which the result exceeds the valid value range of the data

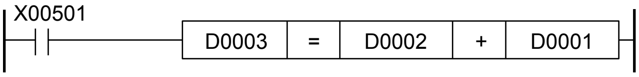


F030303.VSD

Figure 3.3.2 Example of a Calculation in which the Result Exceeds the Valid Value Range of the Data

■ Programming Example

The sample code shown below adds together the values in D0001 and D0002 and assigns the result to D0003 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	CAL	D0003	=	D0002	+	D0001

Note: The "=" operand need not be entered as it is automatically displayed when a CAL instruction is entered.

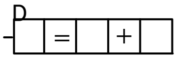
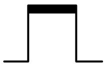
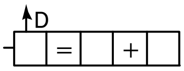
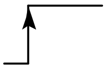
F030304.VSD

Figure 3.3.3 Example of an Addition Program

3.3.2 Add Double Long-word (CAL D)

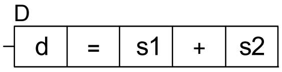
F3SP71
F3SP76

Table 3.3.4 Add Double Long-word

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	20D	Add Double Long-word	CAL D		✓	—		6	64 bit	—
	20DP		↑CAL D					7		

■ Parameter

Add Double Long-word



F332001.VSD

d : Device number of the first device for storing the calculation result
+ : Addition operator
s1, s2 : Data to be added or device numbers of the first devices to be added as data.

■ Available Devices

Table 3.3.5 Devices Available for the Add Double Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d									✓	✓*1	✓*1	✓*1		✓*1			Yes	Yes
s1									✓	✓*1	✓*1	✓		✓		✓	Yes	Yes
s2									✓	✓*1	✓*1	✓		✓		✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Add Double Long-word instruction performs an addition operation on 64-bit data and place the result on the specified devices.

The numbers of bits in the execution results obtained through the Add Double Long-word instructions are summarized in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.3.6 Numbers of Bits Resulting from Add Double Long-word

Specification Item	Instruction
	Add Double Long-word (4-word instruction)
Number of bits in execution result	64 bits
Devices where the execution result is placed	d+3, d+2, d+1, d

The operands on which the Add Double Long-word operation is to be performed can only be binary type data.

● Example of an addition

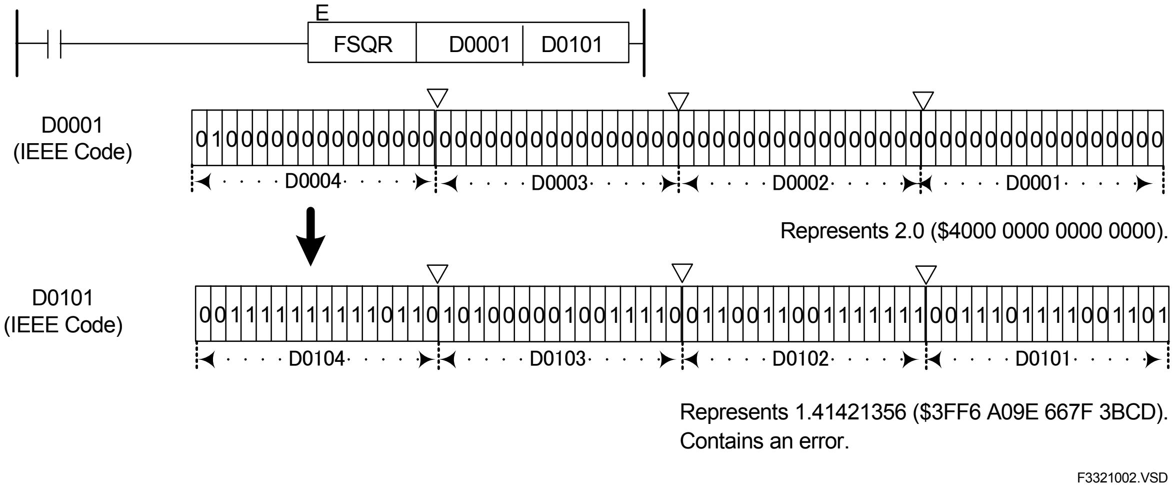


Figure 3.3.4 Example of an Add Double Long-word Instruction

The carry is not set to ON even if the execution result exceeds the valid value range of the data. The Add Double Long-word instruction must be executed so that its execution result does not exceed the value range of the data type. If the execution result exceeds the value range of the data type, the destination device is loaded with a value but the value does not represent the correct execution result.

- **Example of a calculation in which the result exceeds the valid value range of the data**

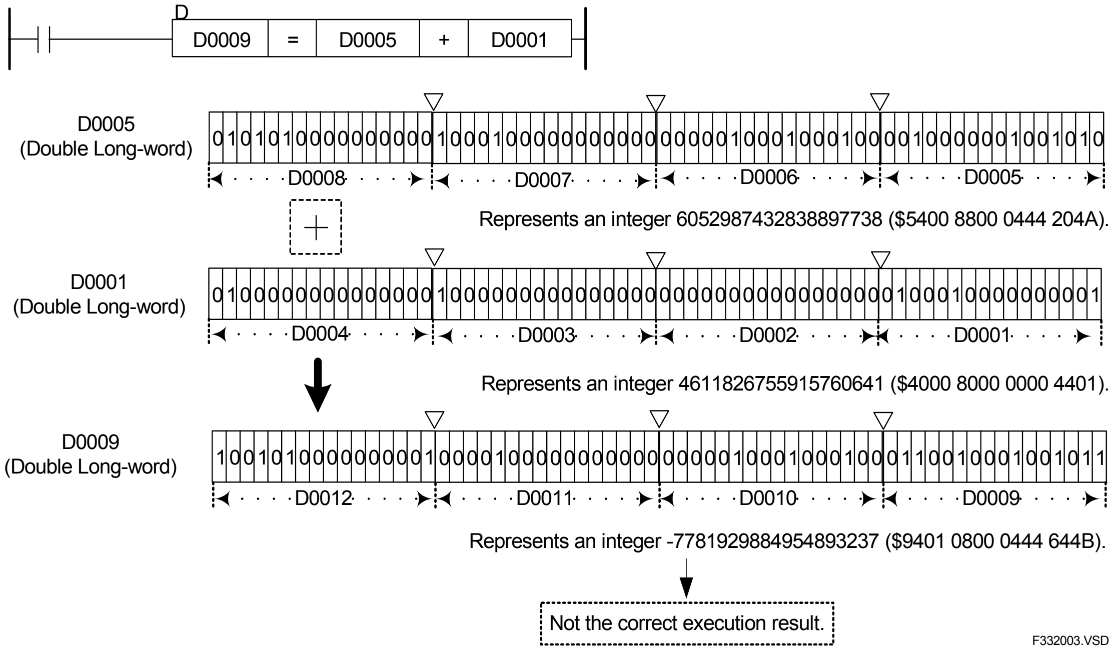
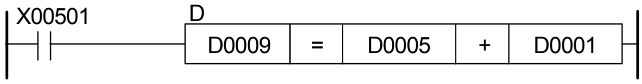


Figure 3.3.5 Example of a Calculation in which the Result Exceeds the Valid Value Range of the Data

■ Programming Example

The sample code shown below adds together the values in D0001 and D0005 and assigns the result to D0009 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	CAL D	D0009	=	D0005	+	D0001

Note: The "=" operand need not be entered as it is automatically displayed when a CAL instruction is entered.

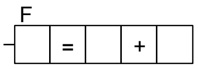
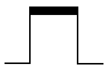
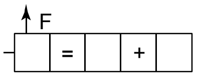
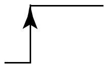
F332004.VSD

Figure 3.3.6 Example of a Double Long-word Addition Program

3.3.3 Add Float (FCAL)

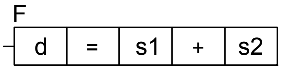
F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.3.7 Add Float

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	903	Add Float	FCAL		✓	—		5	32 bit	—
	903P		↑FCAL					6		

■ Parameter

Add Float



F030305.VSD

d : Device number of the first device for storing the execution result
+ : Addition operator
s1, s2 : Data to be added or device numbers of the first devices to be added as data.
d, s1, and s2 are all in single-precision, floating-point IEEE format (32 bits).

■ Available Devices

Table 3.3.8 Devices Available for the Add Float Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
s1	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Add Float instruction performs an addition on 32-bit data (floating-point data) and places the result on the specified devices.

The operands on which a floating-point addition is to be performed must be represented in the IEEE single-precision floating-point format (use ITOF for conversion or use the result of a floating-point operation).

SEE ALSO

For details on ITOF, see Subsection 3.8.6, "Integer to Float (ITOF), Long-word Integer to Float (ITOF L)."

The number of bits in the execution results obtained through the Add Float instruction is listed in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.3.9 Numbers of Bits Resulting from of Additions

Specification Item	Instruction
	Add Floating point (2-word instruction)
Number of bits in execution result	32 bits
Device where the execution result is placed	d+1, d

● Example of an addition

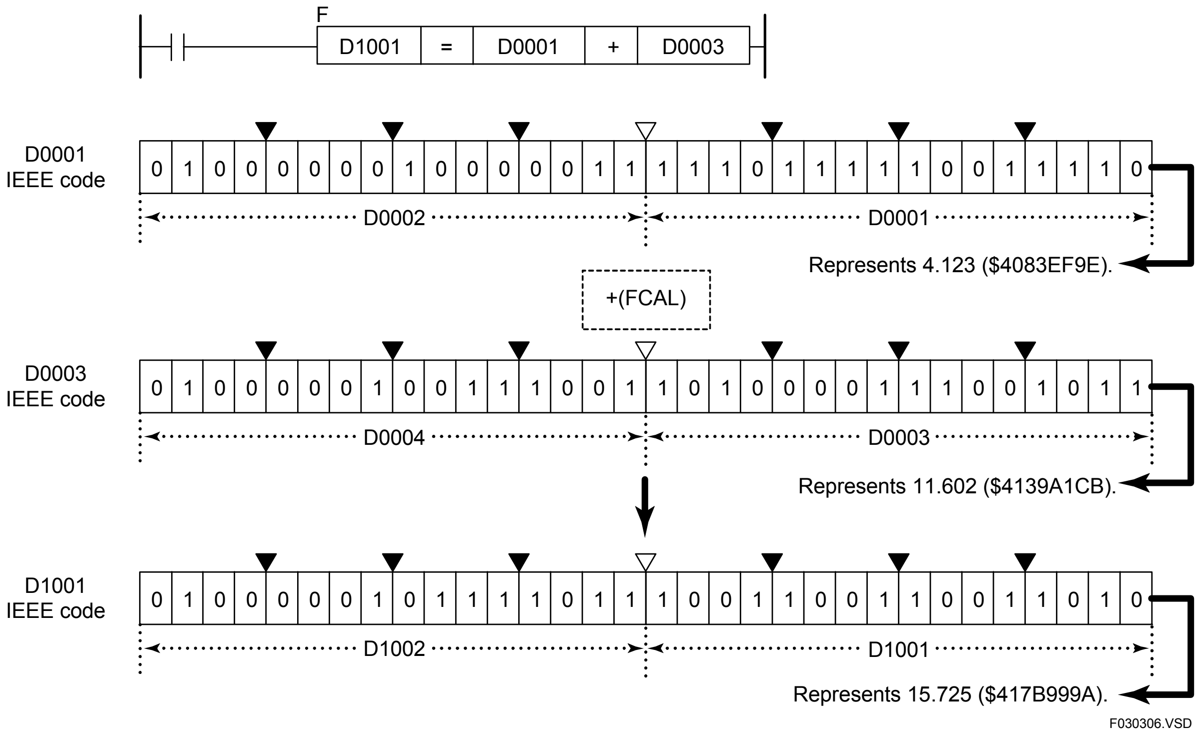
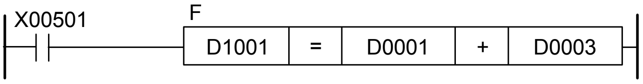


Figure 3.3.7 Example of a Floating-point Addition Instruction

■ Programming Example

The sample code shown below adds together the values stored in locations from D0001 to D002 and from D003 to D0004 and assigns the result to the location from D1001 to D1002 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FCAL	D1001	=	D0001	+	D0003

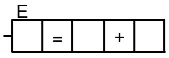
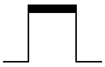
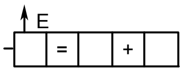
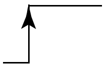
F030307.VSD

Figure 3.3.8 Example of a Floating-point Addition Program

3.3.4 Add Double-precision Float (FCAL E)

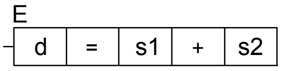
F3SP71
F3SP76

Table 3.3.10 Add Double-precision Float

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	903E	Add Double-precision Float	FCAL E		✓	—		6	64 bit	—
	903EP		↑FCAL E					7		

■ Parameter

Add Double-precision Float



F334001.VSD

d : Device number of the first device for storing the execution result
+ : Addition operator
s1, s2 : Data to be added or device numbers of the first devices to be added as data.
d, s1, and s2 are all in double-precision floating-point IEEE format (64 bits).

■ Available Devices

Table 3.3.11 Devices Available for the Add Double-precision Float Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d									✓	✓	✓*1	✓*1		✓*1			Yes	Yes
s1									✓	✓	✓*1	✓		✓		✓	Yes	Yes
s2									✓	✓	✓*1	✓		✓		✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Add Double-precision Float instruction performs an addition on 64-bit data (double-precision floating-point data) and places the result on the specified devices.

The operands on which a double-precision floating-point addition is to be performed must be represented in the IEEE double-precision floating-point format (use ITOE L and ITOE D for conversion or use the results of a double-precision floating-point operation).

SEE ALSO

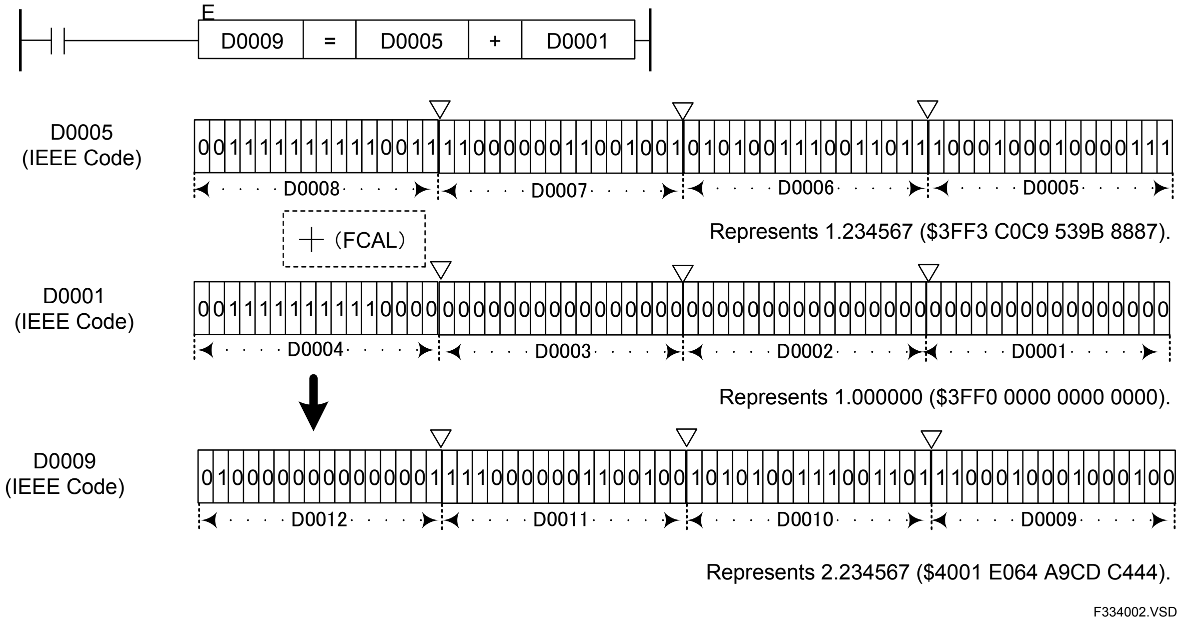
For details on ITOE L and ITOE D, see Subsection 3.8.7, "Long-word Integer to Double-precision Float (ITOE L), Double Long-word Integer to Double-precision Float (ITOE D)."

The number of bits in the execution results obtained through this instruction is listed in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.3.12 Numbers of Bits Resulting from of Addition

Specification Item	Instruction
	Add Double-precision Floating point (4-word instruction)
Number of bits in execution result	64 bits
Devices where the execution result is placed	d+3, d+2, d+1, d

- **Example of an Addition**



Represents 1.234567 ($3FF3 C0C9 539B 8887).

Represents 1.000000 ($3FF0 0000 0000 0000).

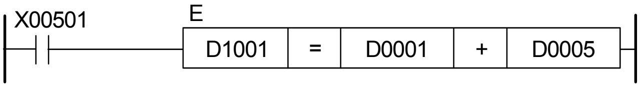
Represents 2.234567 ($4001 E064 A9CD C444).

F334002.VSD

Figure 3.3.9 Example of a Double-precision Floating-point Addition Instruction

■ Programming Example

The sample code shown below adds together the values stored in locations from D0001 to D004 and from D0005 to D0008, and assigns the result to the location from D1001 to D1004 if X00501 is on.



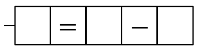
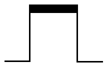
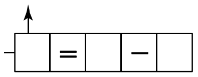
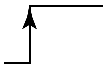
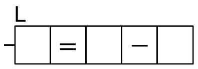
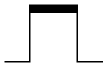
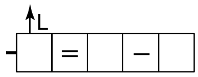
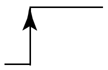
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FCAL E	D1001	=	D0001	+	D0005

F334003.VSD

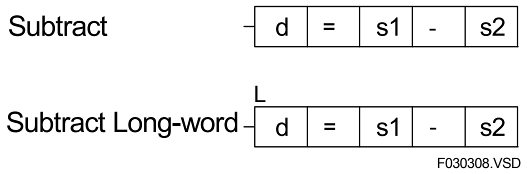
Figure 3.3.10 Example of a Double-precision Floating-point Addition Program

3.3.5 Subtract (CAL), Subtract Long-word (CAL L)

Table 3.3.13 Subtract, Subtract Long-word

Classifi-cation	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	20	Subtract	CAL		✓	—		4	16 bit	—
	20P		↑CAL					5		
	20L	Subtract Long-word	CAL L	 L	✓	—		4	32 bit	—
	20LP		↑CAL L	 L				5		

■ Parameter



d : Device number of the first device storing the execution result
- : Subtraction operator
s1 : Minuend or device number of the first device storing the minuend
s2 : Subtrahend or device number of the first device storing the subtrahend

■ Available Devices

Table 3.3.14 Devices Available for the Subtract and Subtract Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
s1	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value (may not be used as a parameter for the Long-word)
*3: Counter current value (may not be used as a parameter for the Long-word)

■ Function

The Subtract and Subtract Long-word instructions perform a Subtraction on 16- and 32-bit data, respectively, and place the result on the specified devices.

Use the Subtract instruction to subtract 16-bit data and the Subtract Long-word instruction to subtract 32-bit data. Neither Subtract nor Subtract Long-word instructions can perform a subtraction on a mixture of 16- and 32-bit data.

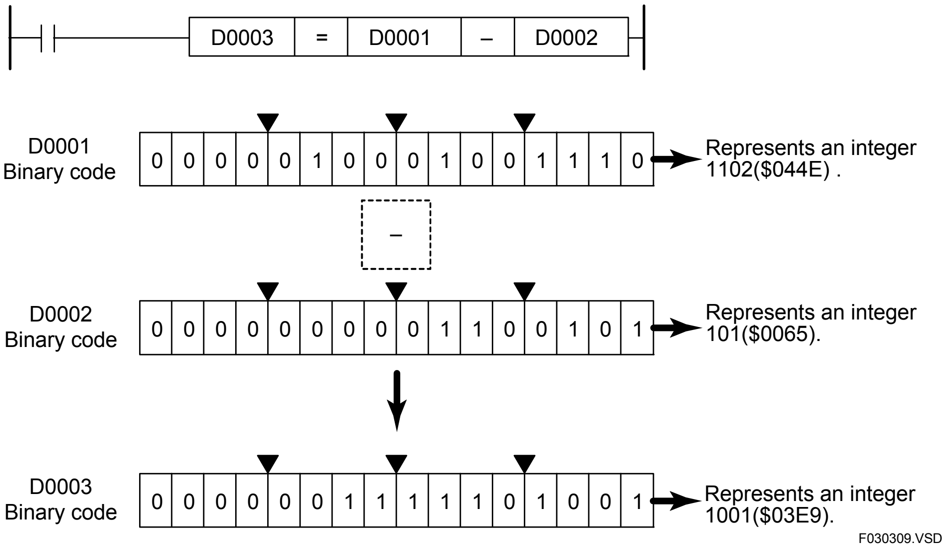
The numbers of bits in the execution results obtained through the Subtract and Subtract Long-word instructions are summarized in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.3.15 Numbers of Bits Resulting from of Subtractions

Specification Item	Instruction	
	Subtract (1-word instruction)	Subtract long-word (2-word instruction)
Number bits execution result	16bit	32 bits
Device where the execution result is placed	d	d+1, d

The operands on which an operation is to be performed can be either of binary, BCD, or a mixture of both types.

● Example of a Subtraction



D0001: 0 0 0 0 0 1 0 0 0 1 0 0 1 1 1 0

D0002: 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1

D0003: 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 1

Figure 3.3.11 Example of a Subtraction

The carry is not set to ON even if the execution result exceeds the valid value range of the data. The Subtract and Subtract Long-word instructions must be executed so that their execution result does not exceed the value range of the respective data type. If the execution result exceeds the value range of the data type, the destination devices are loaded with a value but the value does not represent the correct execution result.

No arithmetic operation is executed if the minuend (s1) or subtrahend (s2) is defined in BCD code and its value exceeds the valid value range of the BCD code. In this case, the value in d remains unchanged.

● Example of a calculation in which the result exceeds the valid value range of the data

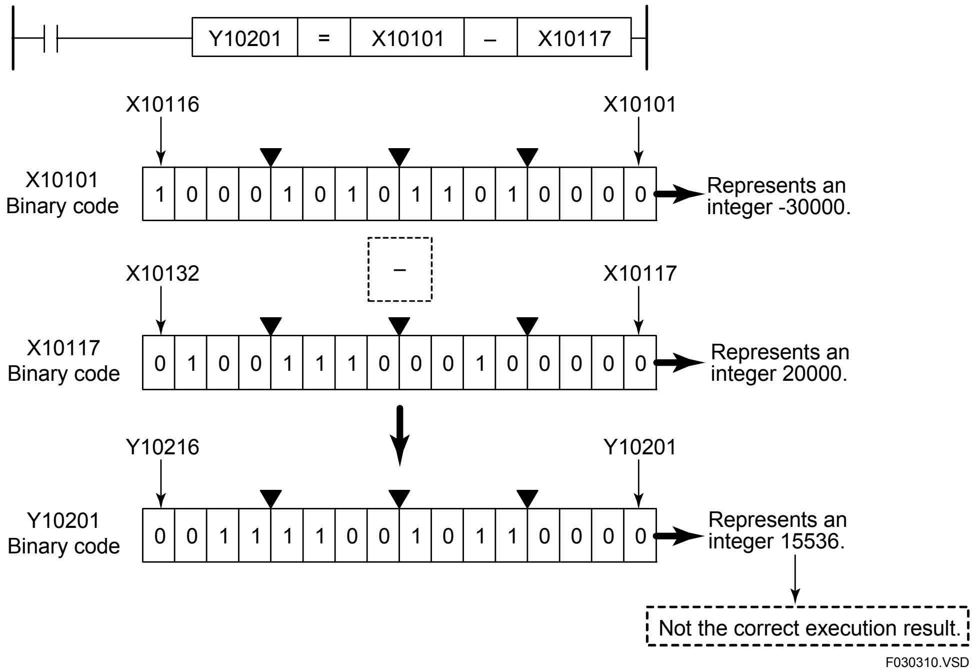
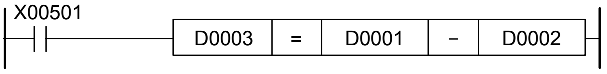


Figure 3.3.12 Example of a Calculation in which the Result Exceeds theValid Value Range of the Data

■ Programming Example

The sample code shown below subtracts the value in D0002 from the value in D0001 and places the result in D0003 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	CAL	D0003	=	D0001	–	D0002

Note: The "=" operand need not be entered as it is automatically displayed when a CAL instruction is entered.

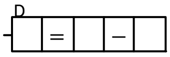
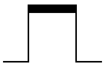
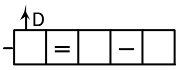
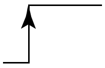
F030311.VSD

Figure 3.3.13 Example of a Subtraction Program

3.3.6 Subtract Double Long-word (CAL D)

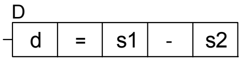
F3SP71
F3SP76

Table 3.3.16 Subtract Double Long-word

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	20D	Subtract Double Long-word	CAL D		✓	—		6	64 bit	—
	20DP		↑CAL D					7		

■ Parameter

Subtract Double Long-word



F336001.VSD

d : Device number of the first device storing the execution result
- : Subtraction operator
s1 : Minuend or device number of the first device storing the minuend
s2 : Subtrahend or device number of the first device storing the subtrahend

■ Available Devices

Table 3.3.17 Devices Available for the Subtract Double Long-word Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d									✓	✓*1	✓*1	✓*1		✓*1			Yes	Yes
s1									✓	✓*1	✓*1	✓		✓		✓	Yes	Yes
s2									✓	✓*1	✓*1	✓		✓		✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Subtract Double Long-word instruction performs a subtraction operation on 64-bit data and place the result on the specified devices.

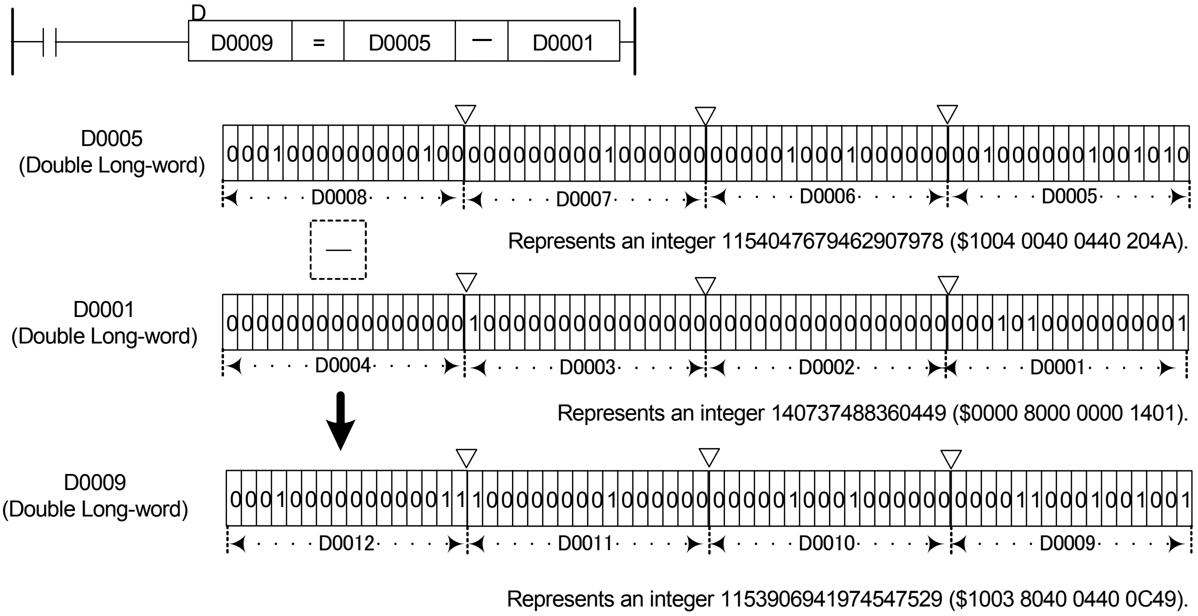
The number of bits in the execution results obtained through this instruction is listed in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.3.18 Numbers of Bits Resulting from of Subtractions

Specification Item	Instruction
	Subtract Double Long-word (4-word instruction)
Number of bits in execution result	64 bits
Devices where the execution result is placed	d+3, d+2, d+1, d

The operands on which the operation is to be performed can only be binary type data.

● Example of a Subtraction



Represents an integer 1154047679462907978 ($1004 0040 0440 204A).

Represents an integer 140737488360449 ($0000 8000 0000 1401).

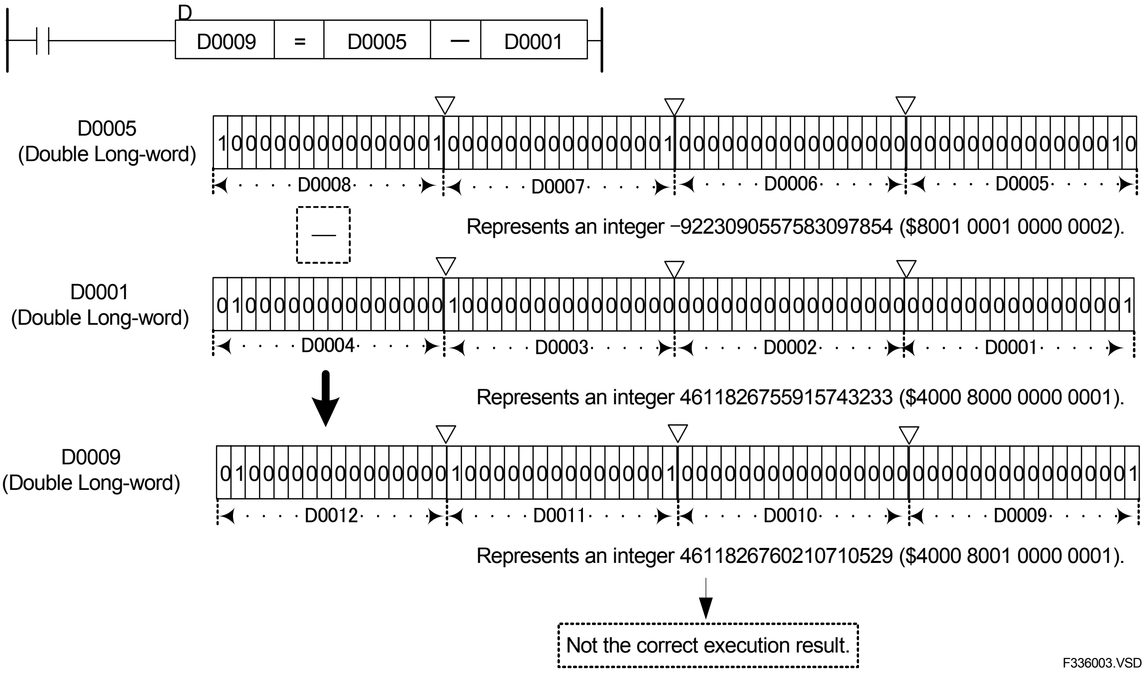
Represents an integer 1153906941974547529 ($1003 8040 0440 0C49).

F336002.VSD

Figure 3.3.14 Example of a Double Long-word Subtraction

The carry is not set to ON even if the execution result exceeds the valid value range of the data. This instruction must be executed so that its execution result does not exceed the value range of the data type. If the execution result exceeds the value range of the data type, the destination devices are loaded with a value but the value does not represent the correct execution result.

● Example of a calculation in which the result exceeds the valid value range of the data



Represents an integer −9223090557583097854 ($8001 0001 0000 0002).

Represents an integer 4611826755915743233 ($4000 8000 0000 0001).

Represents an integer 4611826760210710529 ($4000 8001 0000 0001).

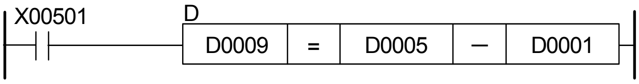
Not the correct execution result.

F336003.VSD

Figure 3.3.15 Example of a Double Long-word Subtraction Instruction

■ Programming Example

The sample code shown below subtracts the value in D0001 from the value in D0005 and places the result in D0009 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	CAL D	D0009	=	D0005	—	D0001

Note: The "=" operand need not be entered as it is automatically displayed when a CAL instruction is entered.

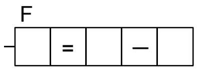
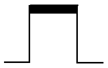
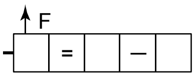
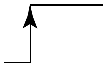
F336004.VSD

Figure 3.3.16 Example of a Double Long-word Subtraction Program

3.3.7 Subtract Float (FCAL)

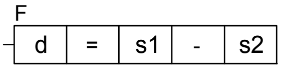
F3SP25
F3SP35 | F3SP22
F3SP28
F3SP38 | F3SP53
F3SP58
F3SP59 | F3SP66
F3SP67 | F3SP71
F3SP76

Table 3.3.19 Subtract Float

Classifi-cation	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	903	Subtract Float	FCAL		✓	—		5	32 bit	—
	903P		↑FCAL					6		

■ Parameter

Subtract Float



F030312.VSD

d : Device number of the first device storing the execution result
- : Subtraction operator
s1 : Minuend or device number of the first device storing the minuend
s2 : Subtrahend or device number of the first device storing the subtrahend
d, s1, and s2 are all in IEEE single-precision floating-point format (32 bits).

■ Available Devices

Table 3.3.20 Devices Available for the Subtract Float Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
s1	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Subtract Float instruction performs a subtraction on 32-bit data (floating-point data) and places the result on the specified devices.

The operands on which a floating-point subtraction is to be performed must be represented in the IEEE single-precision floating-point format (use ITOF for conversion or use the result of a floating-point operation).

SEE ALSO

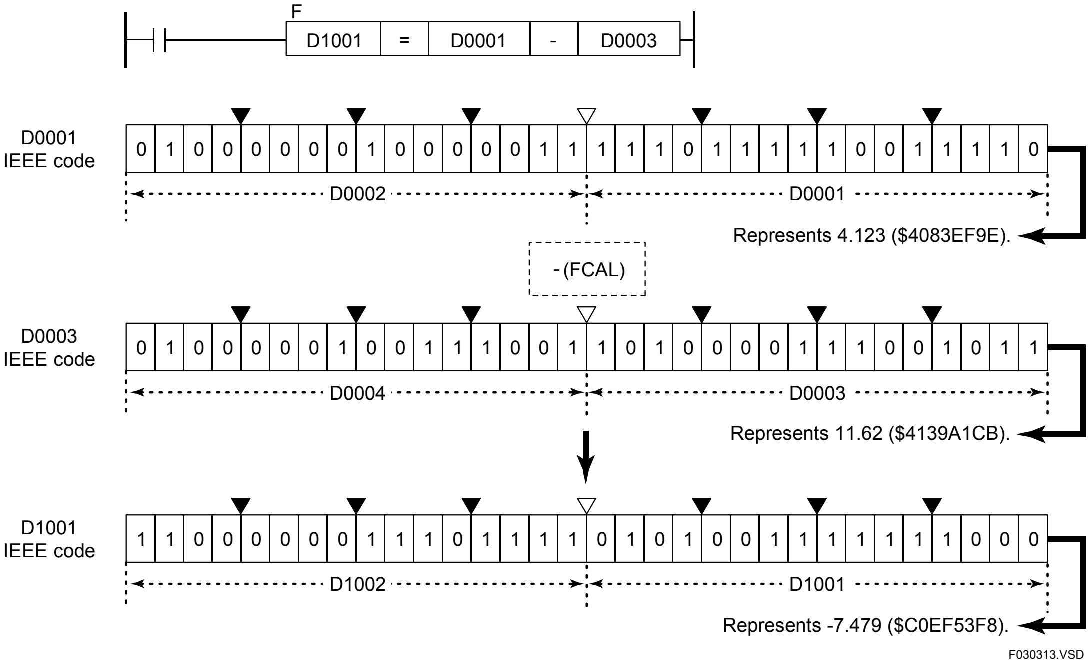
For details on ITOF, see Subsection 3.8.6, "Integer to Float (ITOF), Long-word Integer to Float (ITOF L)."

The number of bits in the execution results obtained through the Subtract Float instruction is listed in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.3.21 Numbers of Bits Resulting from of Subtractions

Specification Item	Instruction
	Subtract Float (2-word instruction)
Number of bits in execution result	32 bits
Device where the execution result is placed	d+1, d

● Example of a subtraction



D0001 IEEE code: 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1 1 0

Represents 4.123 ($4083EF9E).

D0003 IEEE code: 0 1 0 0 0 0 0 1 0 0 1 1 1 0 0 1 1 0 1 0 0 0 0 1 1 1 0 0 1 0 1 1

Represents 11.62 ($4139A1CB).

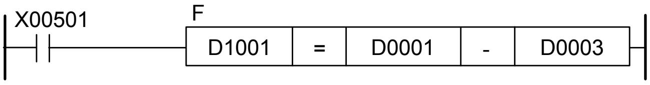
D1001 IEEE code: 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1 1 1 1 1 1 0 0 0

Represents -7.479 ($C0EF53F8).

Figure 3.3.17 Example of a Floating-point Subtraction

■ Programming Example

The sample code shown below subtracts the floating-point data in the location from D0003 to D0004 from the floating-point data in the location from D0001 to D0002 and assigns the result to the location from D1001 to D1002 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FCAL	D1001	=	D0001	—	D0003

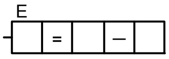
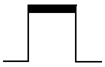
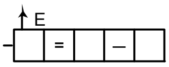
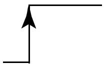
F030314.VSD

Figure 3.3.18 Example of a Floating-point Subtraction Program

3.3.8 Subtract Double-precision Float (FCAL E)

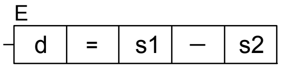
F3SP71
F3SP76

Table 3.3.22 Subtract Double-precision Float

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	903E	Subtract Double-precision Float	FCAL E		✓	—		6	64 bit	—
	903EP		↑FCAL E					7		

■ Parameter

Subtract Double-precision Float



F338001.VSD

d : Device number of the first device storing the execution result
- : Subtraction operator
s1 : Minuend or device number of the first device storing the minuend
s2 : Subtrahend or device number of the first device storing the subtrahend
d, s1, and s2 are all in double-precision floating-point IEEE format (64 bits).

■ Available Devices

Table 3.3.23 Devices Available for the Subtract Double-precision Float Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d									✓	✓	✓*1	✓*1		✓*1			Yes	Yes
s1									✓	✓	✓*1	✓		✓		✓	Yes	Yes
s2									✓	✓	✓*1	✓		✓		✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Subtract Double-precision Float instruction performs a subtraction operation on 64-bit data (double-precision floating-point data) and places the result on the specified devices.

The operands on which a double-precision floating-point addition is to be performed must be represented in the IEEE double-precision floating-point format (use ITOE L and ITOE D for conversion or use the results of a double-precision floating-point operation).

SEE ALSO

For details on ITOE L and ITOE D, see Subsection 3.8.7, "Long-word Integer to Double-precision Float (ITOE L), Double Long-word Integer to Double-precision Float (ITOE D)."

The number of bits in the execution results obtained through this instruction is listed in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.3.24 Numbers of Bits Resulting from of Subtractions

Specification Item	Instruction
	Subtract Double-precision Float (4-word instruction)
Number of bits in execution result	64 bits
Devices where the execution result is placed	d+3, d+2, d+1, d

● Example of a Subtraction

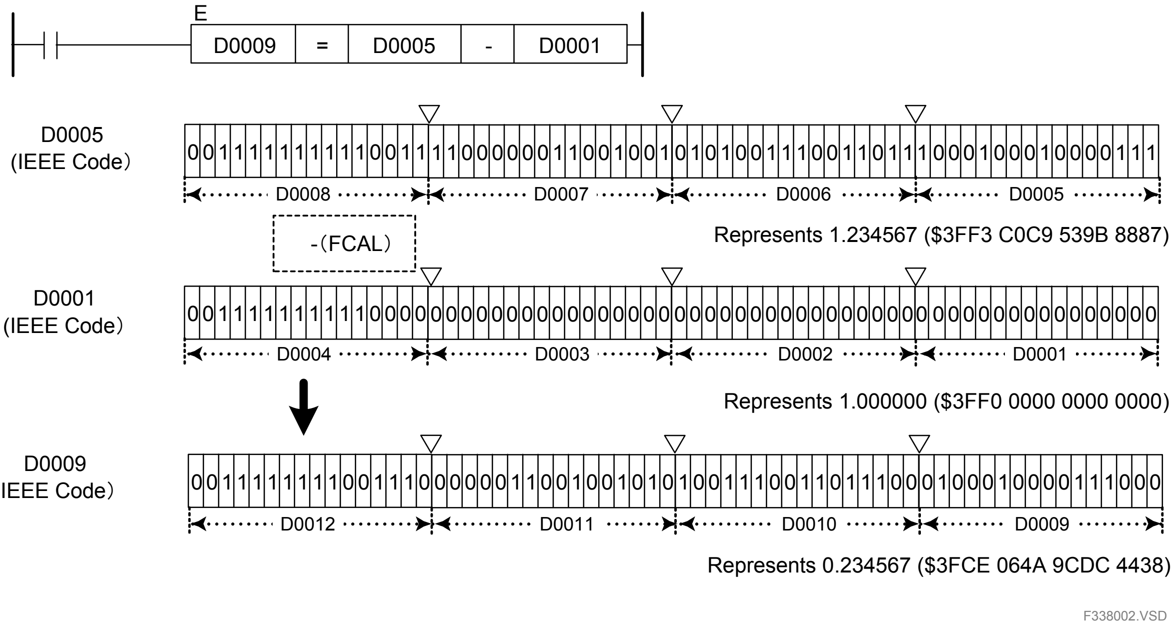
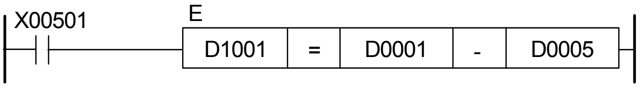


Figure 3.3.19 Example of a Double-precision Floating-point Subtraction Instruction

■ Programming Example

The sample code shown below subtracts the floating-point data in the location from D0005 to D0009 from the floating-point data in the location from D0001 to D0004, and assigns the result to the location from D1001 to D1004 if X00501 is on.



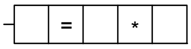
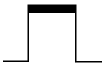
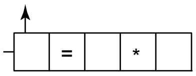
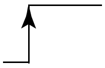
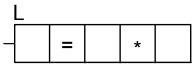
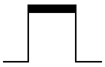
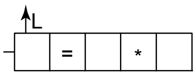
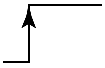
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FCAL E	D1001	=	D0001	-	D0005

F338003.VSD

Figure 3.3.20 Example of a Double-precision Floating-point Subtraction Program

3.3.9 Multiply (CAL), Multiply Long-word (CAL L)

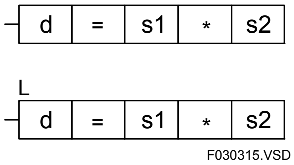
Table 3.3.25 Multiply, Multiply Long-word

Classifi-cation	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	20	Multiply	CAL		✓	—		4	16 bit	—
	20P		↑CAL					5		
	20L	Multiply Long	CAL L		✓	—		4	32 bit	—
	20LP		↑CAL L					5		

■ Parameter

Multiply

Multiply Long-word



d : Device number of the first device storing the execution result

* : Multiplication operator

s1, s2 : Data to be multiplied or device numbers of the first devices to be multiplied as data.

■ Available Devices

Table 3.3.26 Devices Available for the Multiply and Multiply Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1			✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
s1	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value (may not be used as a parameter for the Long-word)

*3: Counter current value (may not be used as a parameter for the Long-word)

■ Function

The Multiply and Multiply Long-word instructions perform a multiplication on 16- and 32-bit data, respectively, and place the result on the specified devices.

Use the Multiply instruction to Multiply 16-bit data and the Multiply Long-word instruction to Multiply 32-bit data. Neither Multiply nor Multiply Long-word instructions can perform a multiplication on a mixture of 16- and 32-bit data.

The numbers of bits in the execution results obtained through the Multiply and Multiply Long-word instructions are summarized in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.3.27   Numbers of Bits Resulting from of Multiplications

Specification Item	Instruction	
	Multiply (1-word instruction)	Multiply long-word (2-word instruction)
Number of bits in execution result	32 bits	64 bits
Device where the execution result is placed	d+1, d	d+3, d+2, d+1, d

The operands on which an operation is to be performed can be either of binary, BCD, or a mixture of both types.

● Example of a multiplication

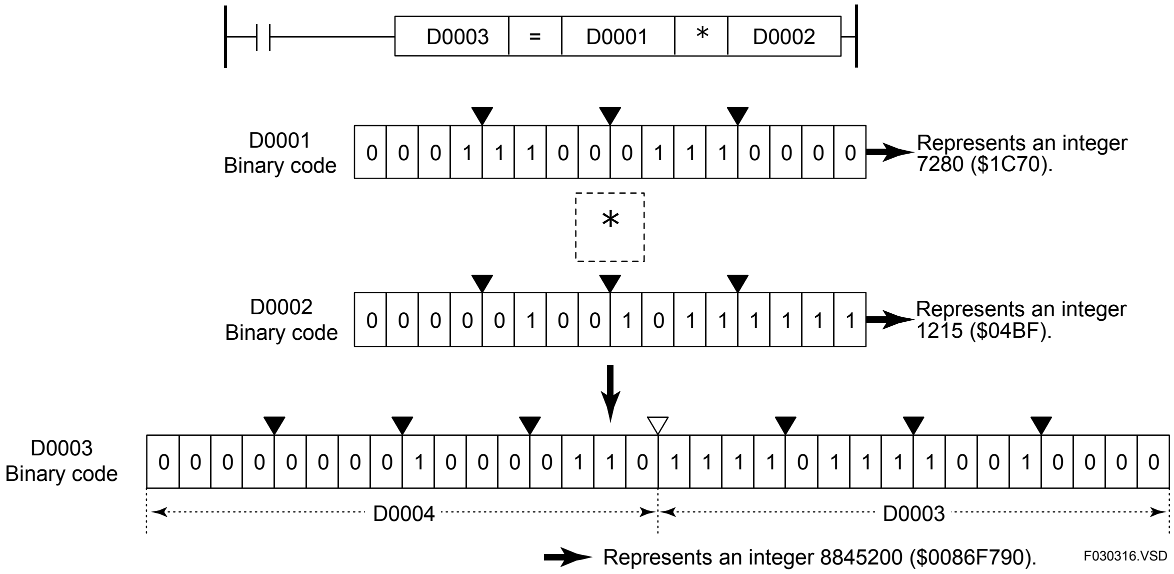
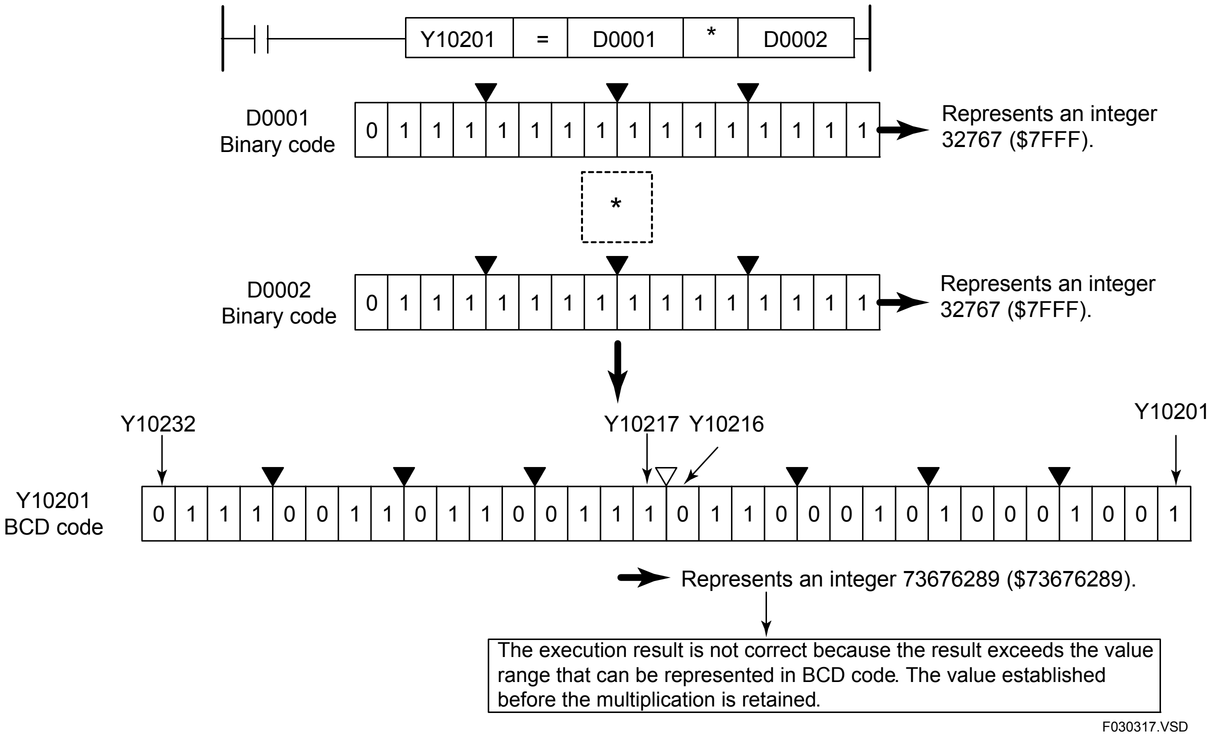


Figure 3.3.21   Example of a Multiplication

The carry is not set to ON even if the execution result exceeds the valid value range of the data. The Multiply and Multiply Long-word instructions must be executed so that their execution result does not exceed the value range of the respective data type (2 words (32 bits) for the 1-word instruction and 4 words (64 bits) for the 2-word instruction). If the execution result exceeds the value range of the data type, the destination device is loaded with a value but the value does not represent the correct execution result.

No arithmetic operation is executed if the operands of multiplication (s1 and s2) are defined in BCD code and their values exceed the valid value range of the BCD code. In this case, the value in d remains unchanged.

● Example of a calculation in which the result exceeds the valid value range of the data

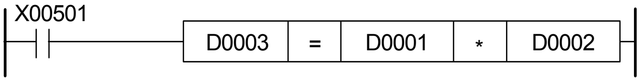


The execution result is not correct because the result exceeds the value range that can be represented in BCD code. The value established before the multiplication is retained.

Figure 3.3.22 Example of a Calculation in which the Result Exceeds the Valid Value Range of the Data

■ Programming Example

The sample code shown below multiplies together the values in D0001 and D0002 and assigns the result to D0003 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	CAL	D0003	=	D0001	*	D0002

Note: The "=" operand need not be entered as it is automatically displayed when a CAL instruction is entered.

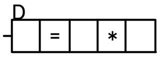
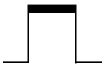
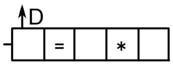
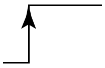
F030318.VSD

Figure 3.3.23 Example of a Multiplication Program

3.3.10 Multiply Double Long-word (CAL D)

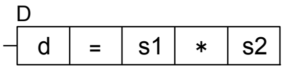
F3SP71
F3SP76

Table 3.3.28 Multiply Double Long-word

Classifi-cation	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	20D	Multiply Double Long-word	CAL D		✓	—		6	64 bit	—
	20DP		↑CAL D					7		

■ Parameter

Multiply Double Long-word



F3310001.VSD

d : Device number of the first device storing the execution result
* : Multiplication operator
s1, s2 : Data to be multiplied or device numbers of the first devices to be multiplied as data.

■ Available Devices

Table 3.3.29 Devices Available for the Multiply Double Long-word Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d									✓	✓*1	✓*1	✓*1		✓*1			Yes	Yes
s1									✓	✓*1	✓*1	✓		✓		✓	Yes	Yes
s2									✓	✓*1	✓*1	✓		✓		✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Multiply Double Long-word instruction performs a signed multiplication on 64-bit data and place the 8-word (128-bit data) result on the specified devices.

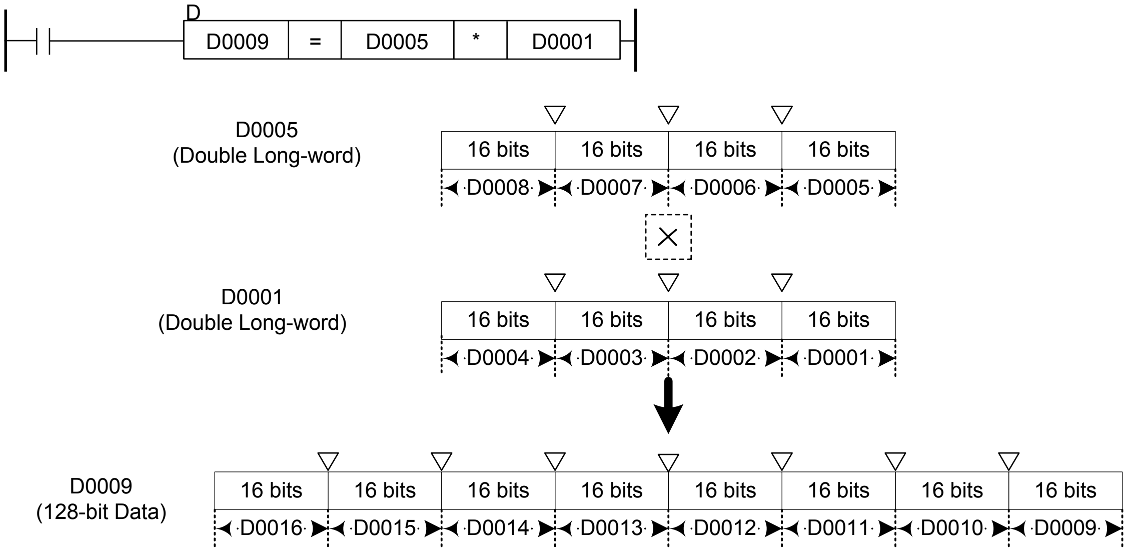
The number of bits in the execution results obtained through this instruction is listed in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.3.30 Numbers of Bits Resulting from of Multiplications

Specification Item	Instruction
	Multiply Double Long-word (4-word instruction)
Number of bits in execution result	128 bits
Devices where the execution result is placed	d+7, d+6, d+5, d+4, d+3, d+2, d+1, d

The operands on which the Multiply Double Long-word operation is to be performed can only be binary type data.

● Example of a Multiplication

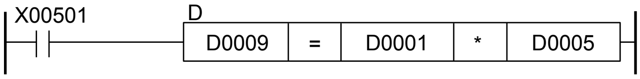


F3310002.VSD

Figure 3.3.24 Example of a Double Long-word Multiplication Instruction

■ Programming Example

The sample code shown below multiplies together the values stored in locations from D0001 to D0004 and from D0005 to D0008, and assigns the result to the locations from D0009 to D0016 if X00501 is on.



Line No.	Instruction	Operands					
0001	LD	X00501					
0002	CAL D	D0009	=	D0001	*	D0005	

Note: The "=" operand need not be entered as it is automatically displayed when a CAL instruction is entered.

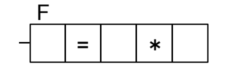
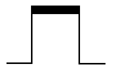
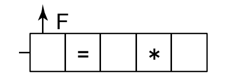
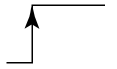
F3310003.VSD

Figure 3.3.25 Example of a Double Long-word Multiplication Program

3.3.11 Multiply Float (FCAL)

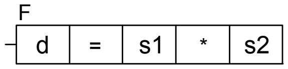
F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.3.31 Multiply Float

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	903	Multiply Float	FCAL		✓	—		5	32 bit	—
	903P		↑FCAL					6		

■ Parameter

Multiply Float



F030319.VSD

d : Device number of the first device storing the execution result
* : Multiplication operator
s1, s2 : Data to be multiplied or device numbers of the first devices to be multiplied as data.
d, s1, and s2 are all in single-precision, floating-point IEEE format (32 bits).

■ Available Devices

Table 3.3.32 Devices Available for the Multiply Float Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
s1	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Multiply Float instruction performs a multiplication on 32-bit data (floating-point data) and places the result on the specified devices.

The operands on which a floating-point multiplication is to be performed must be represented in the IEEE single-precision floating-point format (use ITOF for conversion or use the result of a floating-point operation).

SEE ALSO

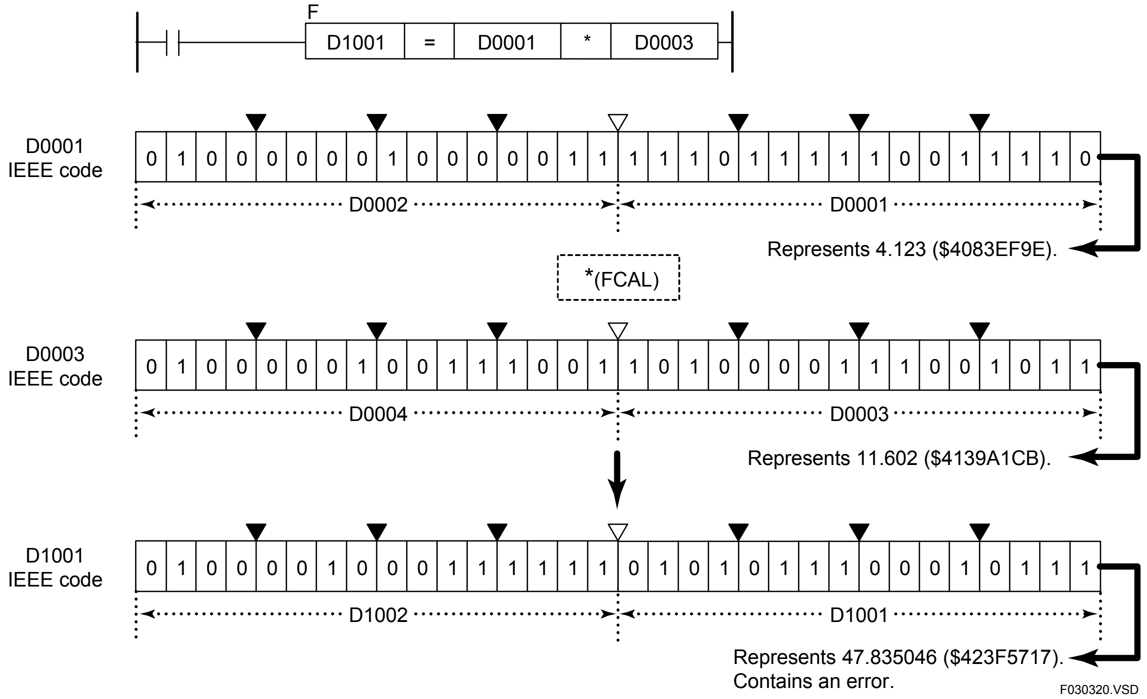
For details on ITOF, see Subsection 3.8.6, "Integer to Float (ITOF), Long-word Integer to Float (ITOF L)."

The number of bits in the execution results obtained through the Multiply Float instruction is listed in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.3.33 Numbers of Bits Resulting from of Multiplication

Specification Item	Instruction
	Multiply Float (2-word instruction)
Number of bits in execution result	32 bits
Device where the execution result is placed	d +1, d

● Example of a Multiplication



D0001 IEEE code: 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1 1 0

D0003 IEEE code: 0 1 0 0 0 0 0 1 0 0 1 1 1 0 0 1 1 0 1 0 0 0 0 1 1 1 0 0 1 0 1 1

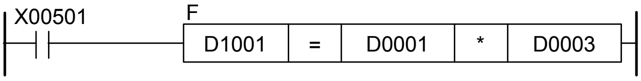
D1001 IEEE code: 0 1 0 0 0 0 1 0 0 0 1 1 1 1 1 1 0 1 0 1 0 1 1 1 0 0 0 1 0 1 1 1

F030320.VSD

Figure 3.3.26 Example of a Floating-point Multiplication

■ Programming Example

The sample code shown below multiplies together the values stored in locations from D0001 to D002 and from D003 to D0004 and assigns the result to the location from D1001 to D1002 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FCAL	D1001	=	D0001	∗	D0003

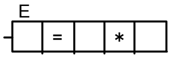
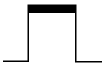
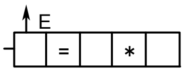
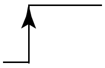
F030321.VSD

Figure 3.3.27 Example of a Floating-point Multiplication Program

3.3.12 Multiply Double-precision Float (FCAL E)

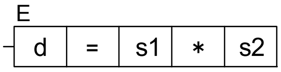
F3SP71
F3SP76

Table 3.3.34 Multiply Double-precision Float

Classifi-cation	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	903E	Multiply Double-precision Float	FCAL E		✓	—		6	64 bit	—
	903EP		↑FCAL E					7		

■ Parameter

Multiply
Double-precision Float



F3312001.VSD

d : Device number of the first device storing the execution result
* : Multiplication operator
s1, s2 : Data to be multiplied or device numbers of the first devices to be multiplied as data.
d, s1, and s2 are all in double-precision floating-point IEEE format (64 bits).

■ Available Devices

Table 3.3.35 Devices Available for the Multiply Double-precision Float Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d									✓	✓	✓*1	✓*1		✓*1			Yes	Yes
s1									✓	✓	✓*1	✓		✓		✓	Yes	Yes
s2									✓	✓	✓*1	✓		✓		✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Multiply Double-precision Float instruction performs a multiplication operation on 64-bit data (double-precision floating-point data) and places the result on the specified devices.

The operands on which a double-precision floating-point addition is to be performed must be represented in the IEEE double-precision floating-point format (use ITOE L and ITOE D for conversion or use the results of a double-precision floating-point operation).

SEE ALSO

For details on ITOE L and ITOE D, see Subsection 3.8.7, "Long-word Integer to Double-precision Float (ITOE L), Double Long-word Integer to Double-precision Float (ITOE D)."

The number of bits in the execution results obtained through this instruction is listed in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.3.36 Numbers of Bits Resulting from of Multiplication

Specification Item	Instruction
	Multiply Double-precision Float (4-word instruction)
Number of bits in execution result	64 bits
Devices where the execution result is placed	d+3, d+2, d +1, d

● . Example of a Multiplication

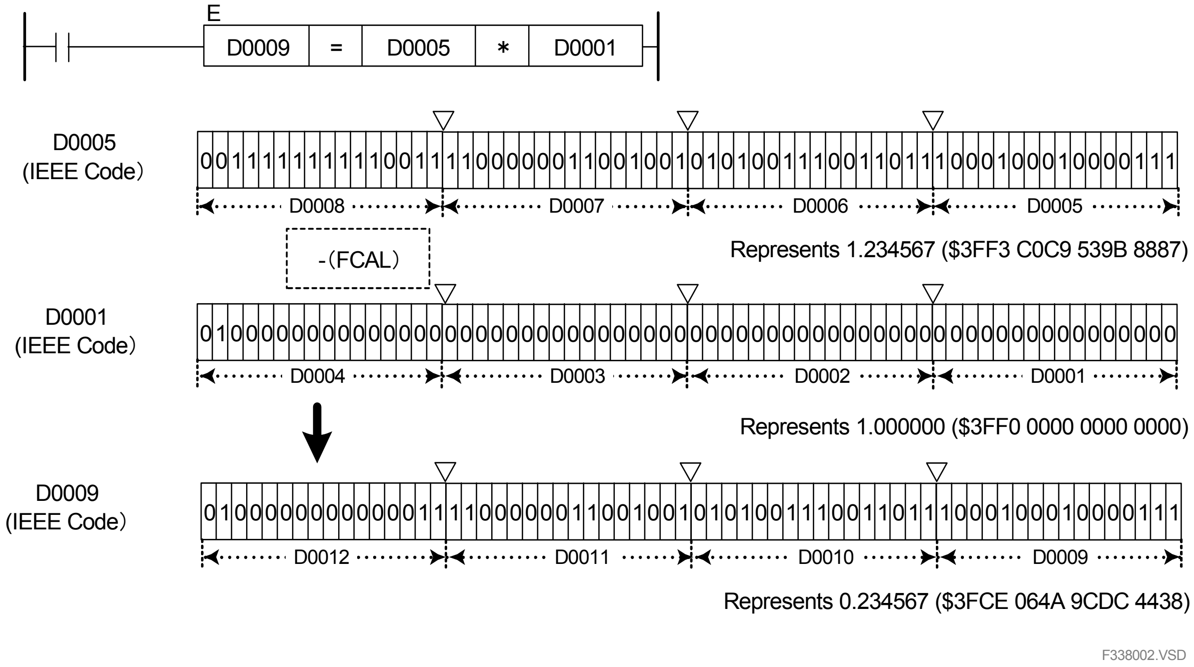
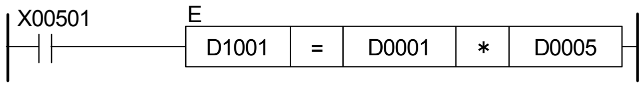


Figure 3.3.28 Example of a Double-precision Floating-point Multiplication

■ Programming Example

The sample code shown below multiplies together the values stored in locations from D0001 to D002 and from D005 to D0008, and assigns the result to the location from D1001 to D1004 if X00501 is on.



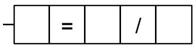
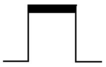
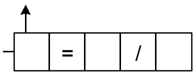
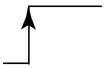
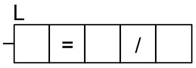
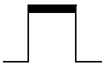
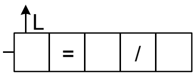
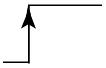
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FCAL E	D1001	=	D0001	*	D0005

F3312003.VSD

Figure 3.3.29 Example of a Double-precision Floating-point Multiplication Program

3.3.13 Divide (CAL), Divide Long-word (CAL L)

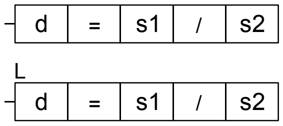
Table 3.3.37 Divide, Divide Long-word

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	20	Divide	CAL		✓	—		4	16 bit	—
	20P		↑CAL					5		
	20L	Divide Long-word	CAL L		✓	—		4	32 bit	—
	20LP		↑CAL L					5		

■ Parameter

Divide

Divide Long-word



F030322.VSD

d : Device number of the first device storing the execution result
/ : Division operator
s1 : Dividend or device number of the first device storing the dividend
s2 : Divisor or device number of the first device storing the divisor

■ Available Devices

Table 3.3.38 Devices Available for the Divide and Divide Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1			✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
s1	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value (may not be used as a parameter for the Long-word)

*3: Counter current value (may not be used as a parameter for the Long-word)

■ Function

The Divide and Divide Long-word instructions perform a division on 16- and 32-bit, respectively, data and place the result on the specified devices.

Use the Divide instruction to divide 16-bit data and the Divide Long-word instruction to divide 32-bit data. Neither Divide nor Divide Long-word instructions can perform a division on a mixture of 16- and 32-bit data.

The numbers of bits in the execution results obtained through the Divide and Divide Long-word instructions are summarized in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.3.39 Numbers of Bits Resulting from of Divisions

Specification Item	Instruction	
	Divide (1-word instruction)	Divide long-word (2-word instruction)
Number of bits in execution result	32 bits	64 bits
Device where the execution result is placed	Quotient : d Remainder : d+1	Quotient : d+1, d Remainder : d+3, d+2

The operands on which an operation is to be performed can be either of binary, BCD, or a mixture of both types.

● Example of a division

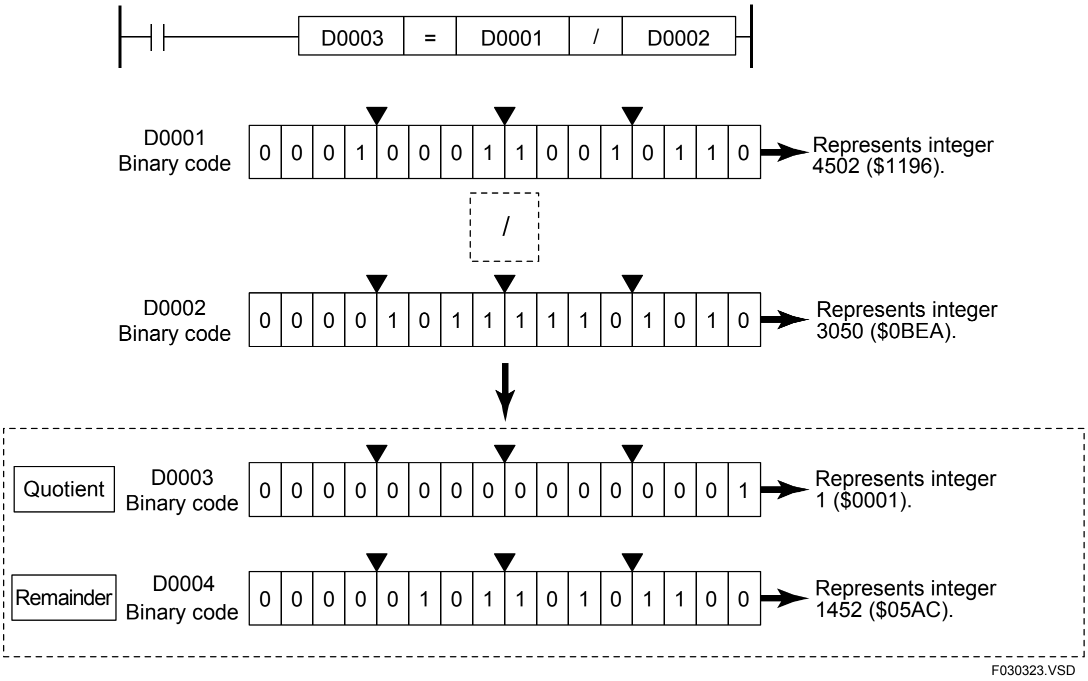


Figure 3.3.30 Example of a Division

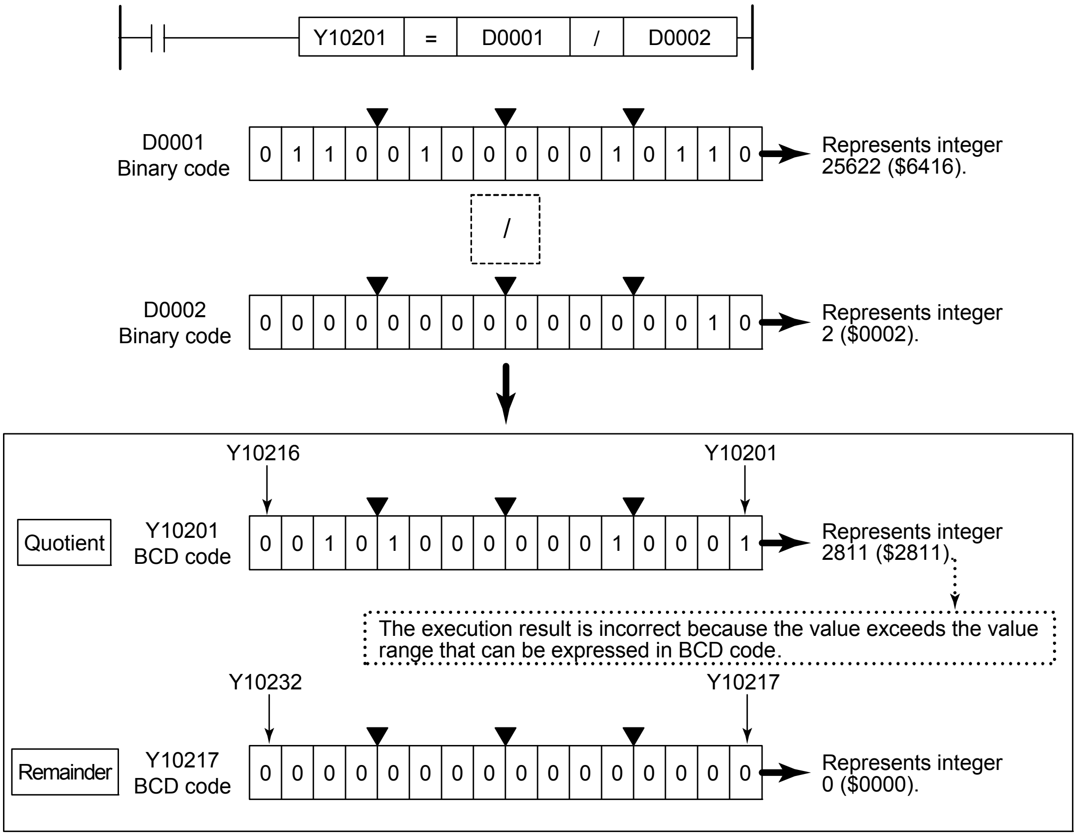
The carry is not set to ON even if the execution result exceeds the valid value range of the data. The Divide and Divide Long-word instructions must be executed so that their execution result does not exceed the value range of the respective data type. If the execution result exceeds the value range of the data type, the destination devices are loaded with a value but the value does not represent the correct execution result.

No arithmetic operation is executed if the dividend (s1) or divisor (s2) is defined in BCD code and its value exceeds the valid value range of the BCD code. In this case, the value in d remains unchanged.

CAUTION

When the divisor (s2) is 0, the special relay M201 is set to ON to signal an instruction error and the division is not executed.

● Example of a calculation in which the result exceeds the valid value range of the data

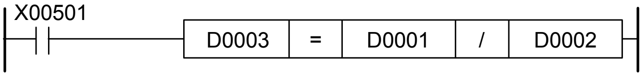


F030324.VSD

Figure 3.3.31 Example of a Calculation in which the Result Exceeds the Valid Value Range of the Data

■ Programming Example

The sample code shown below divides the value in D0001 by the value in D0002 and places the result in D0003 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	CAL	D0003	=	D0001	/	D0002

Note:The "=" operand needs not be entered as it is automatically displayed when a CAL instruction is entered.

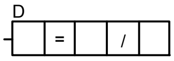
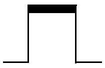
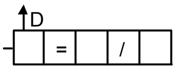
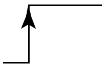
F030325.VSD

Figure 3.3.32 Example of a Division Program

3.3.14 Divide Double Long-word (CAL D)

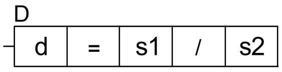
F3SP71
F3SP76

Table 3.3.40 Divide Double Long-word

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	20D	Divide Double Long-word	CAL D		✓	—		6	64 bit	—
	20DP		↑CAL D					7		

■ Parameter

Divide Double
Long-word



F3314001.VSD

d : Device number of the first device storing the execution result
/ : Division operator
s1 : Dividend or device number of the first device storing the dividend
s2 : Divisor or device number of the first device storing the dvisor

■ Available Devices

Table 3.3.41 Devices Available for the Divide Double Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d									✓	✓*1	✓*1	✓*1		✓*1			Yes	Yes
s1									✓	✓*1	✓*1	✓		✓		✓	Yes	Yes
s2									✓	✓*1	✓*1	✓		✓		✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Divide Double Long-word instruction performs a division operation on 64-bit data and place the result on the specified devices.

The number of bits in the execution results obtained through this instruction is listed in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.3.42 Numbers of Bits Resulting from of Divisions

Specification Item	Instruction
	Divide Double Long-word (4-word instruction)
Number of bits in execution result	128 bits
Devices where the execution result is placed	Quotient : d+3, d+2, d+1, d Remainder : d+7, d+6, d+5, d+4

The operands on which the double long-word operation is to be performed can only be binary type data.

● Example of a Division

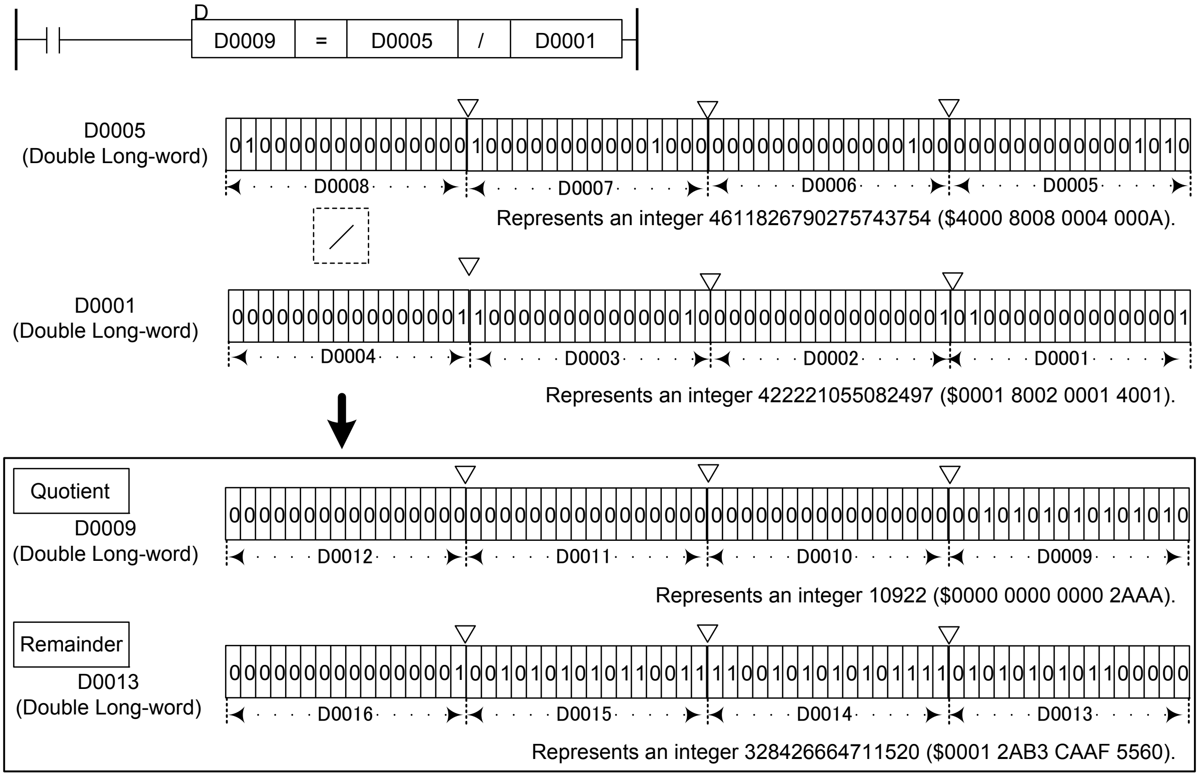


Figure 3.3.33 Example of a Double Long-word Division

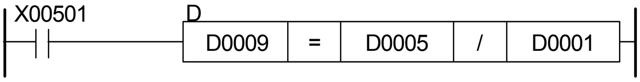
The carry is not set to ON even if the execution result exceeds the valid value range of the data. The Divide Double Long-word instruction must be executed so that its execution result does not exceed the value range of the respective data type. If the execution result exceeds the value range of the data type, the destination devices are loaded with a value but the value does not represent the correct execution result.

CAUTION

When the divisor (s2) is 0, the special relay M201 is set to ON to signal an instruction error and the division is not executed.

■ Programming Example

The sample code shown below divides the value stored in locations from D0005 to D0008 by the value stored in locations from D0001 to D0004, and places the resulting quotient to the locations from D0009 to D0012 and the remainder from D0013 to D0016 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	CAL D	D0009	=	D0005	/	D0001

Note: The "=" operand need not be entered as it is automatically displayed when a CAL instruction is entered.

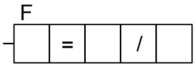
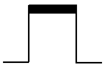
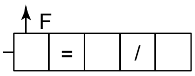
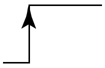
F3314003.VSD

Figure 3.3.34 Example of a Double Long-word Division Program

3.3.15 Divide Float (FCAL)

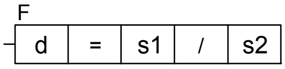
F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.3.43 Divide Float

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	903	Divide Float	FCAL	 F = /	✓	—		5	32 bit	—
	903P		↑FCAL	 ↑ F = /				6		

■ Parameter

Divide Float



F030326.VSD

d : Device number of the first device storing the execution result
/ : Division operator
s1 : Dividend or device number of the first device storing the dividend
s2 : Divisor or device number of the first device storing the divisor
d, s1, and s2 are all in single-precision, floating-point IEEE format (32 bits).

■ Available Devices

Table 3.3.44 Devices Available for the Divide Float Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
s1	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Divide Float instruction performs a division on 32-bit data (floating-point data) and places the result on the specified devices.

The operands on which a floating-point division is to be performed must be represented in the IEEE single-precision floating-point format (use ITOF for conversion or use the result of a floating-point operation).

SEE ALSO

For details on ITOF, see Subsection 3.8.6, "Integer to Float (ITOF), Integer to Float Long (ITOF L)."

The number of bits in the execution results obtained through the Divide Float instruction is listed in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.3.45 Numbers of Bits Resulting from of Divisions

Specification Item	Instruction
	Add Floating point (2-word instruction)
Number of bits execution result	32 bits
Device where the execution result is placed	d+1, d

● Example of a division

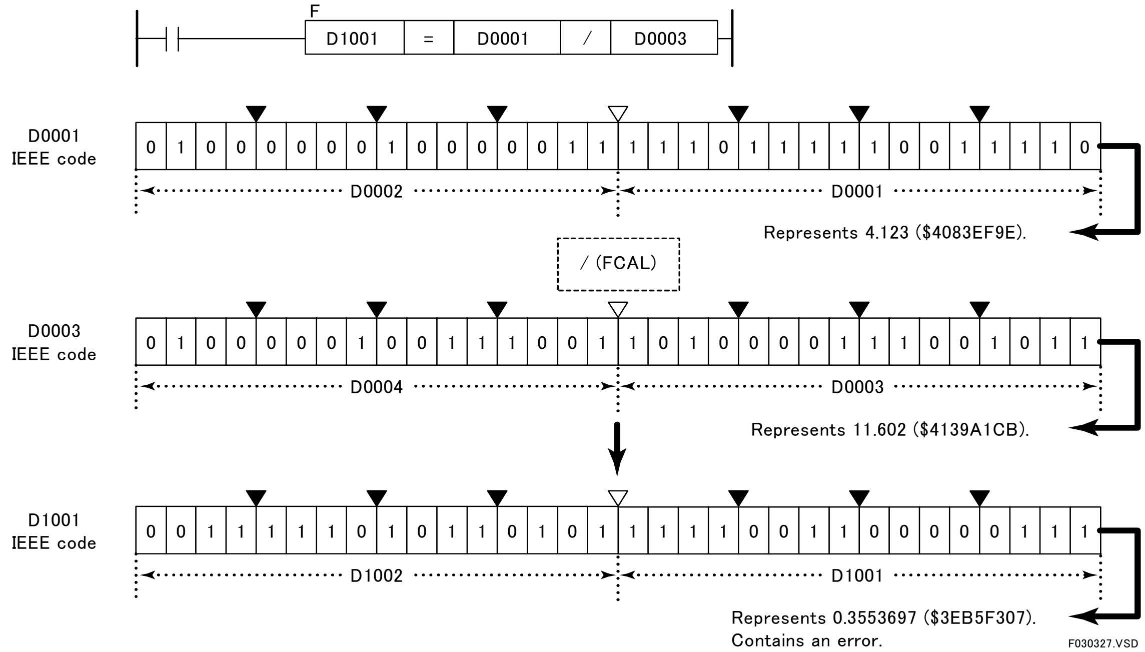


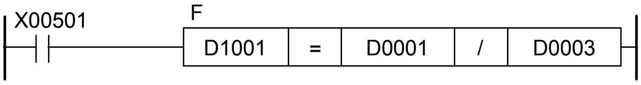
Figure 3.3.35 Example of a Floating-point Division

CAUTION

If the divisor (s2) is 0, the special relay M201 is set to ON to signal an instruction error and the division is not executed.

■ Programming Example

The sample code shown below divides the floating-point data in the location from D0001 to D0002 by the floating-point data in the location from D0003 to D0004 and assigns the result to the location from D1001 to D1002 if X00501 is on.



Line No.	Instruction	Operands					
0001	LD	X00501					
0002	FCAL	D1001	=	D0001	/	D0003	

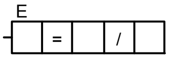
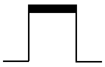
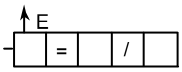
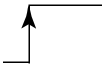
F030328.VSD

Figure 3.3.36 Example of a Floating-point Division Program

3.3.16 Divide Double-precision Float (FCAL E)

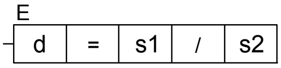
F3SP71
F3SP76

Table 3.3.46 Divide Double-precision Float

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	903E	Divide Double-precision Float	FCAL E		✓	—		6	64 bit	—
	903EP		↑FCAL E					7		

■ Parameter

Divide Double-precision Float



F3316001.VSD

d : Device number of the first device storing the execution result
/ : Division operator
s1 : Dividend or device number of the first device storing the dividend
s2 : Divisor or device number of the first device storing the divisor
d, s1, and s2 are all in double-precision floating-point IEEE format (64 bits).

■ Available Devices

Table 3.3.47 Devices Available for the Divide Double-precision Float Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d									✓	✓	✓*1	✓*1		✓*1			Yes	Yes
s1									✓	✓	✓*1	✓		✓		✓	Yes	Yes
s2									✓	✓	✓*1	✓		✓		✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Divide Double-precision Float instruction performs a division on 64-bit data (double-precision floating-point data) and places the result on the specified devices.

The operands on which a double-precision floating-point division is to be performed must be represented in the IEEE double-precision floating-point format (use ITOE L and ITOE D for conversion or use the results of a double-precision floating-point operation).

SEE ALSO

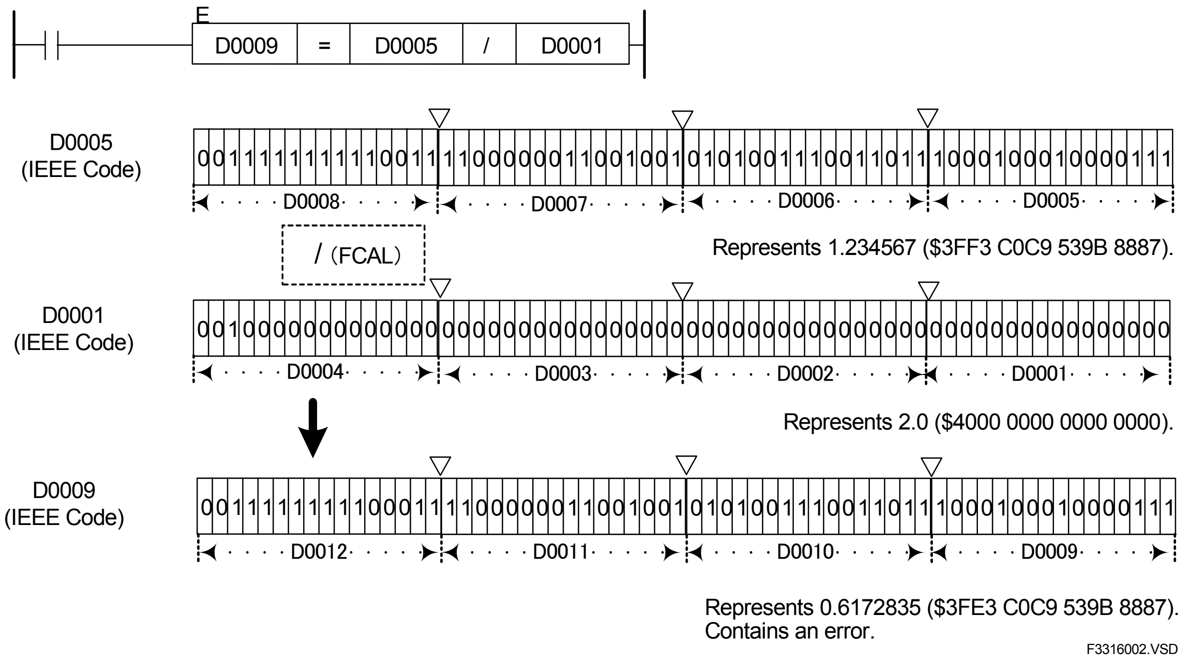
For details on ITOE L and ITOE D, see Subsection 3.8.7, "Long-word Integer to Double-precision Float (ITOE L), Double Long-word Integer to Double-precision Float (ITOE D)."

The number of bits in the execution results obtained through this instruction is listed in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.3.48 Numbers of Bits Resulting from of Double-precision Floating-Point Divisions

Specification Item	Instruction
	Divide Double-precision Floating-point (4-word instruction)
Number of bits in execution result	64 bits
Devices where the execution result is placed	d+3, d+2, d+1, d

● Example of a Division



Represents 1.234567 ($3FF3 C0C9 539B 8887).

Represents 2.0 ($4000 0000 0000 0000).

Represents 0.6172835 ($3FE3 C0C9 539B 8887).
Contains an error.

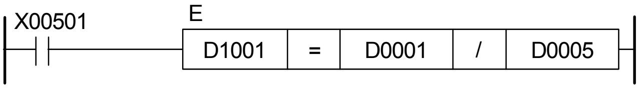
Figure 3.3.37 Example of a Double-precision Floating-point Division

CAUTION

If the divisor (s2) is 0, the special relay M201 is set to ON to signal an instruction error and the division is not executed.

■ Programming Example

The sample code shown below divides the double-precision floating-point data in the location from D0001 to D0004 by the double-precision floating-point data in the location from D0005 to D0008, and assigns the result to the location from D1001 to D1004 if X00501 is on.



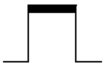
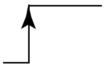
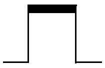
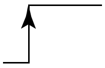
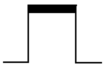
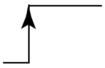
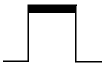
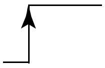
Line No.	Instruction	Operands					
0001	LD	X00501					
0002	FCAL E	D1001	=	D0001	/	D0005	

F3316003.VSD

Figure 3.3.38 Example of a Double-precision Floating-point Division Program

3.3.17 Increment (INC), Increment Long-word (INC L), Decrement (DEC), Decrement Long-word (DEC L)

Table 3.3.49 Increment, Decrement

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Application Instruction	120	Increment	INC	INC	✓	—		2	16 bit	—
	120P		↑INC	↑ INC				3		
	120L	Increment Long-word	INC L	L INC	✓	—		2	32 bit	—
	120LP		↑INC L	↑L INC				3		
	121	Decrement	DEC	DEC	✓	—		2	16 bit	—
	121P		↑DEC	↑ DEC				3		
	121L	Decrement Long-word	DEC L	L DEC	✓	—		2	32 bit	—
	121LP		↑DEC L	↑L DEC				3		

■ Parameter

Increment	INC d
Increment Long-word	L INC d
Decrement	DEC d
Decrement Long-word	L DEC d

F030329.VSD

d : Device number of the first device storing the data to be incremented or decremented

■ Available Devices

Table 3.3.50 Devices Available for the Increment and Decrement Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value (may not be used as a parameter for the Long-word)
*3: Counter current value (may not be used as a parameter for the Long-word)

■ Function

(1) Incrementing

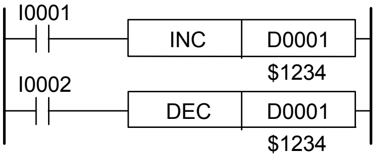
The Increment and Increment Long-word instructions increment 16- and 32-bit data d by 1, respectively. Use the Increment instruction to increment 16-bit data and the Increment Long-word instruction to increment 32-bit data.

(2) Decrementing

The Decrement and Decrement Long-word instructions decrement 16- and 32-bit data d by 1, respectively. Use the Decrement instruction to decrement 16-bit data and the Decrement Long-word instruction to decrement 32-bit data.

■ Programming Example

The sample code shown below increments the data in D0001 ($1234) if I0001 is on and decrements the data if I0002 is ON.



Before executing	$1234
After INC is executing	$1235
After DEC is executing	$1233

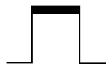
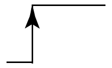
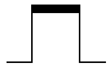
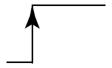
Line No.	Instruction	Operands				
0001	LD	I0001				
0002	INC	D0001				
0003	LD	I0002				
0004	DEC	D0001				

F030330.VSD

Figure 3.3.39 Example of an Increment/Decrement Program

3.3.18 Square Root (SQR), Long-word Square Root (SQR L)

Table 3.3.51 Square Root, Long-word Square Root

Classifi-cation	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	122	Square Root	SQR	SQR	✓	—		2	16 bits	—
	122P		↑SQR	↑ SQR				3		
	122L	Long-word Square Root	SQR L	L SQR	✓	—		2	32 bits	—
	122LP		↑SQR L	↑L SQR				3		

■ Parameter

Square Root — SQR | d

Long-word Square Root — L SQR | d

F030331.VSD

d : Device number of the first device storing the data whose square root is to be calculated

■ Available Devices

Table 3.3.52 Devices Available for the Square Root and Long-word Square Root Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value (may not be used as a parameter for the Long-word)
*3: Counter current value (may not be used as a parameter for the Long-word)

■ Function

The Square Root and Long-word Square Root instructions calculate the square root of the 16- and 32-bit data d, respectively, and places the result on the specified devices.

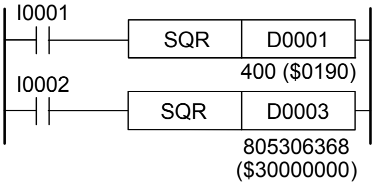
Use the Square Root instruction to calculate the square root of 16-bit data and the Long-word Square Root instruction to calculate the square root of 32-bit data. The fractional part of the result is truncated.

CAUTION

If d is a negative number, the special relay M201 is set to ON to signal an instruction processing error and the instruction is not executed.

■ Programming Example

The sample code shown below calculates the square roots of the value 400 ($0190) in location D0001 if I0001 is on and calculates the square root of the value 805306368 ($30000000) if I0002 is on.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	SQR	D0001				
0003	LD	I0002				
0004	SQR L	D0003				

	Square Root (D0001)	Square Root Long-word (D0004, D0003)
Before execution	400($0190)	805306368($30000000)
After execution	20($0014)	28377 ($00006ED9)

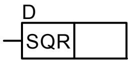
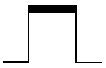
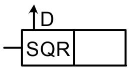
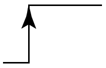
F030332.VSD

Figure 3.3.40 Example of a Square Root Program

3.3.19 Double Long-word Square Root (SQR D)

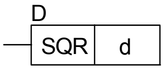
F3SP71
F3SP76

Table 3.3.53 Double Long-word Square Root

Classifi-cation	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	122D	Double Long-word Square Root	SQR D		✓	—		3	64 bits	—
	122DP		↑SQR D					4		

■ Parameter

Double Long-word Square Root



F3319001.VSD

d : Device number of the first device storing the data whose square root is to be calculated in double long-word

■ Available Devices

Table 3.3.54 Devices Available for the Double Long-word Square Root Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d									✓	✓*1	✓*1	✓		✓*1			Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

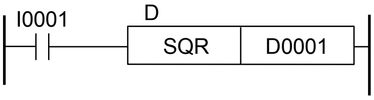
The Double Long-word Square Root instruction calculates the square root of the 64-bit data and places the result in double long-word data. The fractional part of the result is truncated.

CAUTION

If d is a negative number, the special relay M201 is set to ON to signal an instruction processing error and the instruction is not executed.

■ Programming Example

The sample code shown below calculates the square root of the value 1152921504606846976 ($1000000000000000) in locations from D0001 to D0004 if I0001 is on.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	SQR D	D0001				

	Double Long-word Square Root D0004, D0003, D0002, D0001
Before execution	1152921504606846976（$1000 0000 0000 0000）
After execution	1073741824（$0000 0000 4000 0000）

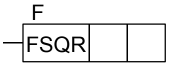
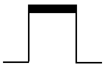
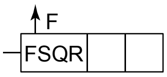
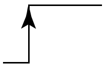
F3319002.VSD

Figure 3.3.41 Example of a Double Long-word Square Root Program

3.3.20 Square Root Float (FSQR)

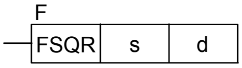
F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.3.55 Square Root Float

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	915	Square Root Float	FSQR		✓	—		4	32 bit	—
	915P		↑FSQR					5		

■ Parameter

Square Root Float



F030333.VSD

s : Data or device number of the first device storing the data whose square root is to be calculated
d : Device number of the first device storing the execution result
Both s and d are represented in the IEEE single-precision floating-point format (32 bits).

■ Available Devices

Table 3.3.56 Devices Available for the Square Root Float Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

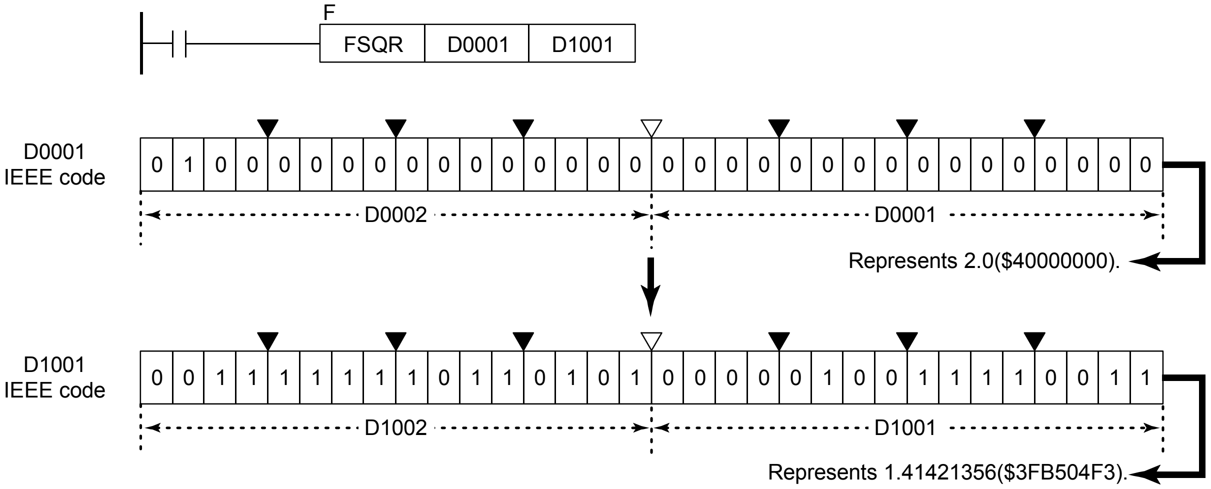
*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Square Root Float instruction calculates the square root of 32-bit real data s (single-precision floating-point data) and places the result in d.

The operand s (single-precision floating-point) must be represented in the IEEE format. The result d (single-precision floating-point) is also represented in the IEEE format.

● Example of a floating-point square root operation



F030334.VSD

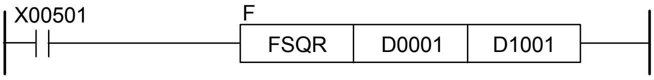
Figure 3.3.42 Example of a Floating-point Square Root Operation

CAUTION

If s is a negative number, the special relay M201 is set to ON to signal an instruction processing error and the instruction is not executed.

■ Programming Example

The sample code shown below calculates the square root of the real (single-precision floating point) data in location D0001 to D0002 and loads the result into the location from D1001 to D1002 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FSQR	D0001	D1001			

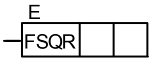
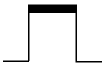
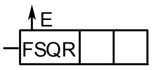
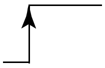
F030335.VSD

Figure 3.3.43 Example of a Floating-point Square Root Program

3.3.21 Square Root Double-precision Float (FSQR E)

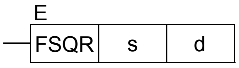
F3SP71
F3SP76

Table 3.3.57 Square Root Double-precision Float

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	915E	Square Root Double-precision Float	FSQR E		✓	—		4	64 bit	—
	915EP		↑FSQR E					5		

■ Parameter

Square Root Double-precision Float



F3321001.VSD

s : Data or device number of the first device whose square root is to be calculated in double-precision floating-point format

d : Device number of the first device storing the result of the square root in double-precision floating-point format

Both s and d are represented in the IEEE double-precision floating-point format (64 bits).

■ Available Devices

Table 3.3.58 Devices Available for the Square Root Double-precision Float Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s									✓	✓	✓*1	✓		✓		✓	Yes	Yes
d									✓	✓	✓*1	✓*1		✓*1			Yes	Yes

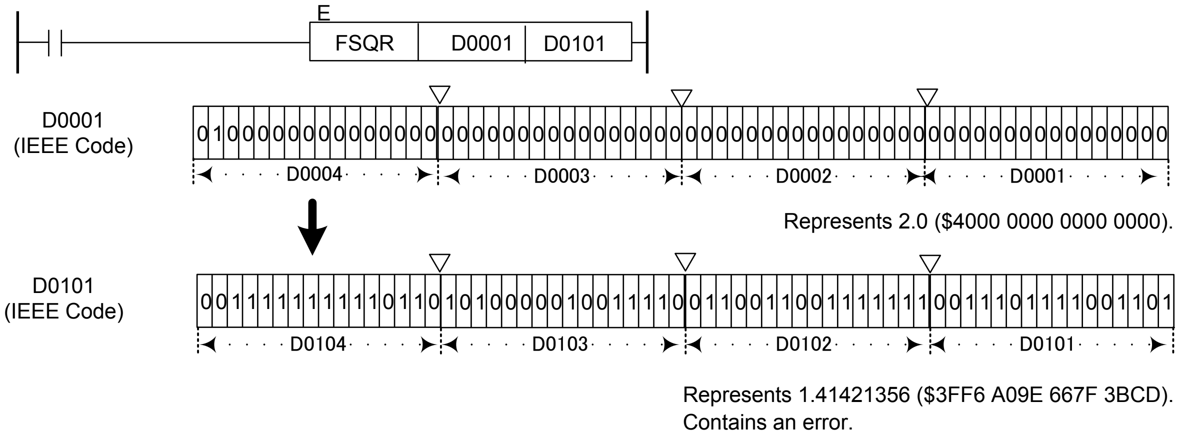
*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Square Root Double-precision Float instruction calculates the square root of 64-bit real data s (double-precision floating-point data) and places the result in d.

The operand s (double-precision floating-point) must be represented in the IEEE format. The result d (double-precision floating-point) is also represented in the IEEE format.

● Example of a Double-precision Floating-point Square Root Operation



Represents 2.0 ($4000 0000 0000 0000).

Represents 1.41421356 ($3FF6 A09E 667F 3BCD).
Contains an error.

F3321002.VSD

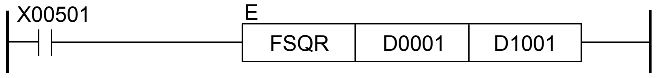
Figure 3.3.44 Example of a Double-precision Floating-point Square Root Operation

CAUTION

If s is a negative number, the special relay M201 is set to ON to signal an instruction processing error and the instruction is not executed.

■ Programming Example

The sample code shown below calculates the square root of the real (double-precision floating point) data in location D0001 to D0004 and loads the result into the location from D1001 to D1004 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FSQR E	D0001	D1001			

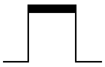
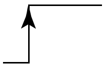
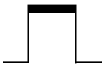
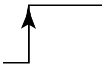
F3321003.VSD

Figure 3.3.45 Example of a Double-precision Floating-point Square Root Program

3.3.22 SIN (FSIN), SIN^{-1} (FASIN)

F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

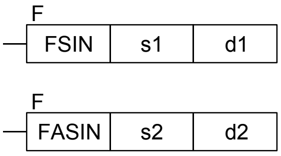
Table 3.3.59 Sine, Arc Sine

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	907	SIN (Sine)	FSIN	F FSIN	✓	—		4	32 bit	—
	907P		↑FSIN	↑F FSIN				5		
	910	SIN^{-1} (Arc Sine)	FASIN	F FASIN	✓	—		4	32 bit	—
	910P		↑FASIN	↑F FASIN				5		

■ Parameter

SIN(Sine)

SIN^{-1}(Arc Sine)



F030336.VSD

s1 : Angle data (in radians) whose sine is to be calculated or device number of the first device storing the angle data whose sine is to be calculated

d1 : Device number of the first device storing the execution result (sine)

Both s1 and d1 are represented in the IEEE single-precision floating-point format (32 bits).

s2 : Data whose arc sine is to be calculated or device number of the first device storing the data whose arc sine is to be calculated

d2 : Device number of the first device storing the execution result (arc sine)

Both s2 and d2 are in IEEE single-precision floating-point format (32 bits).

■ Available Devices

Table 3.3.60 Devices Available for the Sine and Arc Sine Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s1, s2	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d1, d2		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

(1) SIN (Sine)

The Sine instruction calculates the sine (single-precision floating-point) of given angle data (single-precision floating-point) specified in radians. The equation for and outline of the sine calculation is shown in the figure below.

d1 = SIN (s1) s1 : Angle whose sine is to be calculated (in radians) $(-\frac{\pi}{2} \leqq s1 \leqq \frac{\pi}{2})$

d1 : Execution result $(-1 \leqq d1 \leqq 1)$

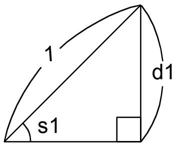


Figure 3.3.46 SIN (Sine)

The single-precision floating-point numbers are represented in the IEEE format.

(2) SIN^{-1} (Arc Sine)

The Arc Sine instruction calculates the arc sine of a given real number (single-precision floating-point) in radians (single-precision floating-point). The equation for and outline of the arc sine calculation is shown in the figure below.

d2 = SIN^{-1} (s2) s2 : Real number data whose arc sine is to be calculated $(-1 \leqq s2 \leqq 1)$

d2 : Execution result (in radians) $(-\frac{\pi}{2} \leqq d2 \leqq \frac{\pi}{2})$

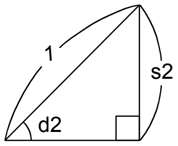


Figure 3.3.47 SIN^{-1} (Arc Sine)

The single-precision floating-point numbers are represented in the IEEE format.

● Example

$$SIN(60°)=SIN\left(\frac{\pi}{3}\right)=\left(\frac{\sqrt{3}}{2}\right)$$

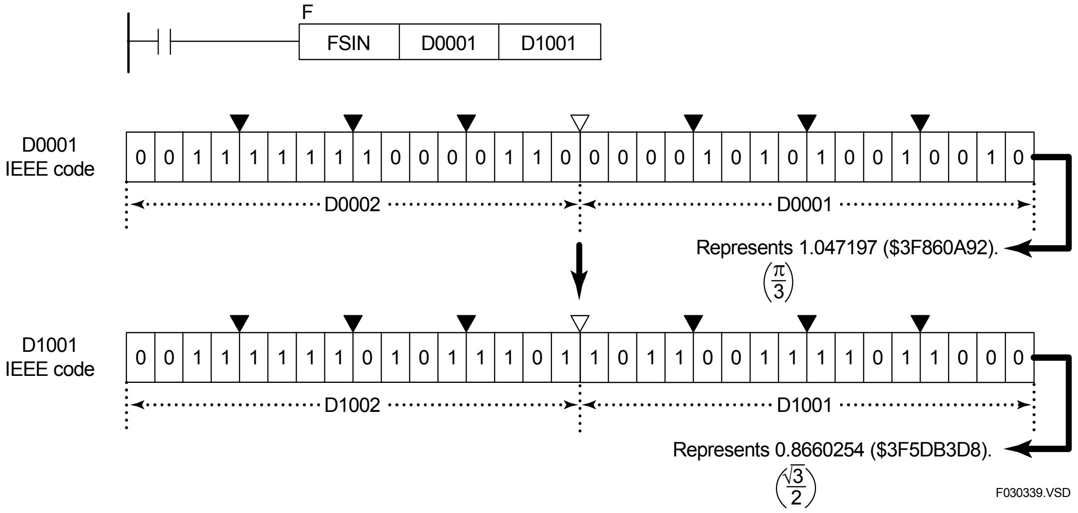
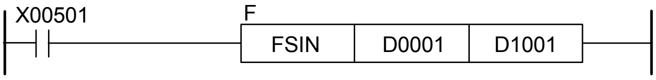


Figure 3.3.48 Example of a Sine Calculation

■ Programming Example

(1) SIN

The sample code shown below calculates the sine of the angle data (in radians) in location D0001 to D0002 and loads the execution result into the location D1001 to D1002 if X00501 is on.



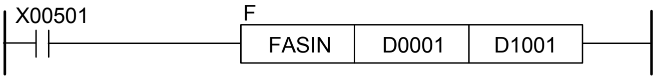
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FSIN	D0001	D1001			

F030340.VSD

Figure 3.3.49 Example of a Sine Program

(2) SIN^{-1}

The sample code shown below calculates the arc sine of the real data in location D0001 to D0002 and loads the execution result into the location D1001 to D1002 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FASIN	D0001	D1001			

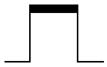
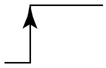
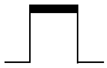
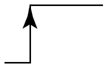
F030341.VSD

Figure 3.3.50 Example of an Arc Sine Program

3.3.23 COS (FCOS), COS^{-1} (FACOS)

F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

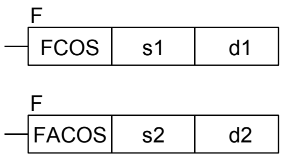
Table 3.3.61 Cosine, Arc Cosine

Classifi-cation	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	908	COS (Cosine)	FCOS	F FCOS	✓	—		4	32 bit	—
	908P		↑FCOS	↑F FCOS				5		
	911	COS^{-1} (Arc Cosine)	FACOS	F FACOS	✓	—		4	32 bit	—
	911P		↑FACOS	↑F FACOS				5		

■ Parameter

COS(Cosine)

COS^{-1}(Arc Cosine)



F030342.VSD

s1 : Angle data (in radians) whose cosine is to be calculated or device number of the first device storing the angle data whose cosine is to be calculated
d1 : Device number of the first device storing the execution result (cosine)
Both s1 and d1 are represented in the IEEE single-precision floating-point format (32 bits).

s2 : Data whose arc cosine is to be calculated or device number of the first device storing the data whose arc cosine is to be calculated
d2 : Device number of the first device storing the execution result (arc cosine)
Both s2 and d2 are represented in the IEEE single-precision floating-point format (32 bits).

■ Available Devices

Table 3.3.62 Devices Available for the Cosine and Arc Cosine Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s1, s2	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d1, d2		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

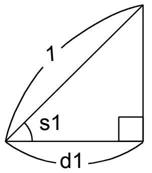
■ Function

(1) COS (Cosine)

The Cosine instruction calculates the cosine (single-precision floating-point) of given angle data (single-precision floating-point) specified in radians. The equation for and outline of the cosine calculation is shown in the figure below.

d1 = COS (s1)   s1 : Angle whose cosine is to be calculated (in radians) $(0 \leqq s1 \leqq \pi)$

d1 : Execution result $(-1 \leqq d1 \leqq 1)$



F030343.VSD

Figure 3.3.51 COS (Cosine)

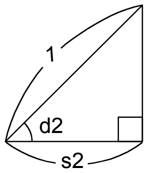
The single-precision floating-point numbers are represented in the IEEE format.

(2) COS^{-1} (Arc Cosine)

The Arc Cosine instruction calculates the arc cosine of a given real number (single-precision floating-point) in radians (single-precision floating-point). The equation for and outline of the arc cosine calculation is shown in the figure below.

d2 = COS^{-1} (s2)   s2 : Real number data whose arc cosine is to be calculated $(-1 \leqq s2 \leqq 1)$

d2 : Execution result (in radians) $(0 \leqq d2 \leqq \pi)$



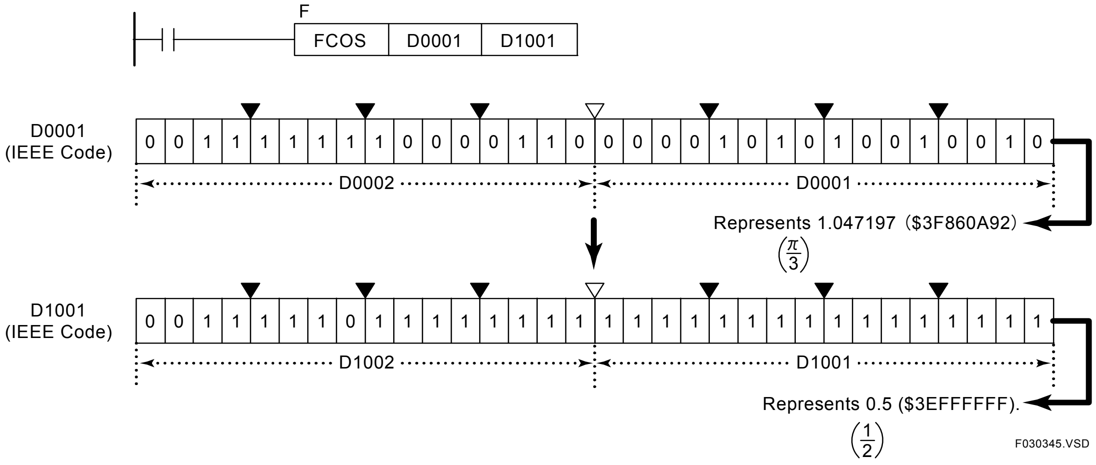
F030344.VSD

Figure 3.3.52 COS^{-1} (Arc Cosine)

The single-precision floating-point numbers are represented in the IEEE format.

● Example

$$\mathrm{COS}\,(60°) = \mathrm{COS}\left(\frac{\pi}{3}\right) = \left(\frac{1}{2}\right)$$



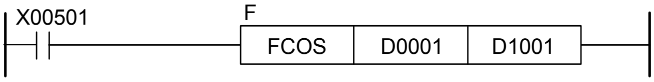
F030345.VSD

Figure 3.3.53 Example of a Cosine Calculation

■ Programming Example

(1) COS

The sample code shown below calculates the cosine of the angle data (in radians) in location D0001 to D0002 and loads the execution result into the location D1001 to D1002 if X00501 is on.



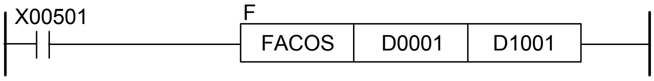
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FCOS	D0001	D1001			

F030346.VSD

Figure 3.3.54 Example of a Cosine Program

(2) COS^{-1}

The sample code shown below calculates the arc cosine of the real data in location D0001 to D0002 and loads the execution result into the location D1001 to D1002 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FACOS	D0001	D1001			

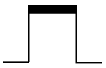
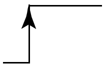
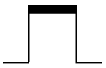
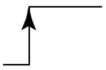
F030347.VSD

Figure 3.3.55 Example of an Arc Cosine Program

3.3.24 TAN (FTAN), TAN^{-1} (FATAN)

F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.3.63 Tangent, Arc Tangent

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Application Instruction	909	TAN (Tangent)	FTAN	F FTAN	✓	—		4	32 bit	—
	909P		↑FTAN	↑F FTAN				5		
	912	TAN^{-1} (Arc Tangent)	FATAN	F FATAN	✓	—		4	32 bit	—
	912P		↑FATAN	↑F FATAN				5		

■ Parameter

TAN(Tangent)

F
| FTAN | s1 | d1 |

TAN^{-1}(Arc Tangent)

F
| FATAN | s2 | d2 |

F030348.VSD

s1 : Angle data (in radians) whose tangent is to be calculated or device number of the first device storing the angle data whose tangent is to be calculated
d1 : Device number of the first device storing the execution result (tangent)
Both s1 and d1 are represented in the IEEE single-precision floating-point format (32 bits).

s2 : Data whose arc tangent is to be calculated or device number of the first device storing the data whose arc tangent is to be calculated
d2 : Device number of the first device storing the execution result (arc tangent)
Both s2 and d2 are represented in the IEEE single-precision floating-point format (32 bits).

■ Available Devices

Table 3.3.64 Devices Available for the Tangent and Arc Tangent Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s1, s2	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d1, d2		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

(1) TAN (Tangent)

The Tangent instruction calculates the tangent (single-precision floating-point) of given angle data (single-precision floating-point) specified in radians. The equation for and outline of the Tangent calculation is shown in the figure below.

d1 = TAN (s1)

s1 : Angle whose tangent is to be calculated (in radians) $\left(-\frac{\pi}{2} \leqq s1 \leqq \frac{\pi}{2}\right)$

d1 : Execution result $(-\infty \leqq d1 \leqq \infty)$

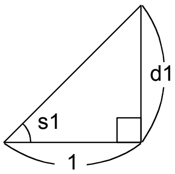


Figure 3.3.56 TAN (Tangent)

The single-precision floating-point numbers are represented in the IEEE format.

(2) TAN^{-1} (Arc Tangent)

The Arc Tangent instruction calculates the arc tangent of a given real number (single-precision floating-point) in radians (single-precision floating-point). The equation for and outline of the arc tangent calculation is shown in the figure below.

d2 = TAN^{-1} (s2)

s2 : Real number data whose arc tangent is to be calculated $(-\infty \leqq s2 \leqq \infty)$

d2 : Execution result (in radians) $\left(-\frac{\pi}{2} \leqq d2 \leqq \frac{\pi}{2}\right)$

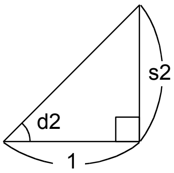


Figure 3.3.57 TAN^{-1} (Arc Tangent)

The single-precision floating-point numbers are represented in the IEEE format.

CAUTION

- The error of a tangent calculation is greater near $\pi/2$ and $-\pi/2$.
- The value of arc tangent calculation is greater near -2^{128} and 2^{128}.

● Example

$$\mathrm{TAN}(60°) = \mathrm{TAN}\left(\frac{\pi}{3}\right) = \sqrt{3}$$

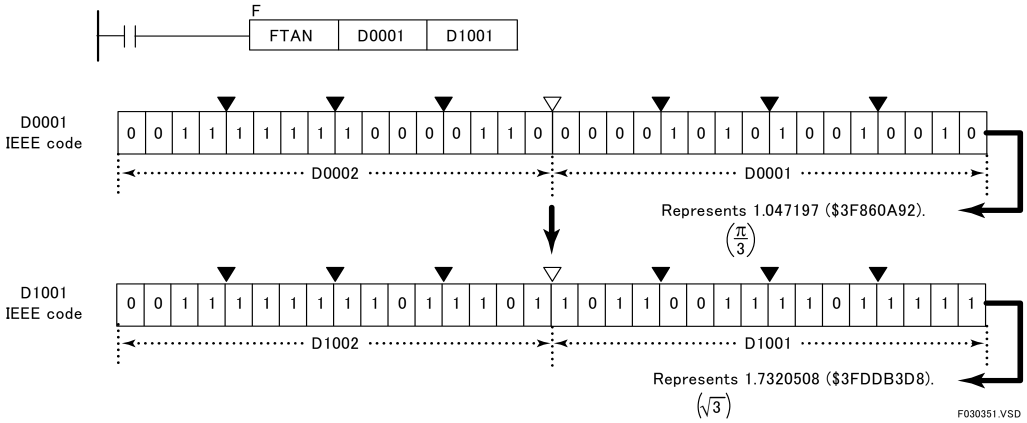
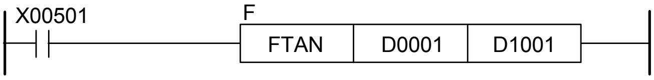


Figure 3.3.58 Example of a Tangent Calculation

■ Programming Example

(1) TAN

The sample code shown below calculates the tangent of the angle data (in radians) in location D0001 to D0002 and loads the execution result into the location D1001 to D1002 if X00501 is on.



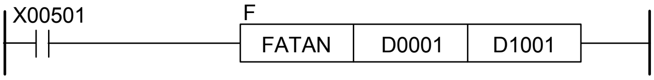
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FTAN	D0001	D1001			

F030352.VSD

Figure 3.3.59 Example of a Tangent Program

(2) TAN^{-1}

The sample code shown below calculates the arc tangent of the real data in location D0001 to D0002 and loads the execution result into the location D1001 to D1002 when if X00501 is ON.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FATAN	D0001	D1001			

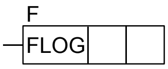
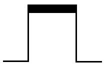
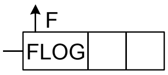
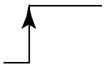
F030353.VSD

Figure 3.3.60 Example of an Arc Tangent Program

3.3.25 LOG (FLOG)

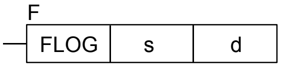
F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.3.65 Logarithm

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	913	LOG (Logarithm)	FLOG		✓	—		4	32 bit	—
	913P		↑FLOG					5		

■ Parameter

LOG(Logarithm)



F030354.VSD

s : Data whose logarithm is to be calculated or device number of the first device storing the data whose logarithm is to be calculated

d : Device number of the first device storing the execution result (logarithm)

Both s and d are represented in the IEEE single-precision floating-point format (32 bits).

■ Available Devices

Table 3.3.66 Devices Available for the Logarithm Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Logarithm instruction calculates the natural logarithm (logarithm to the base e) of a given real number (single-precision floating point). The equation for and outline of the logarithm calculation is shown in the figure below.

d = $LOG_e s$

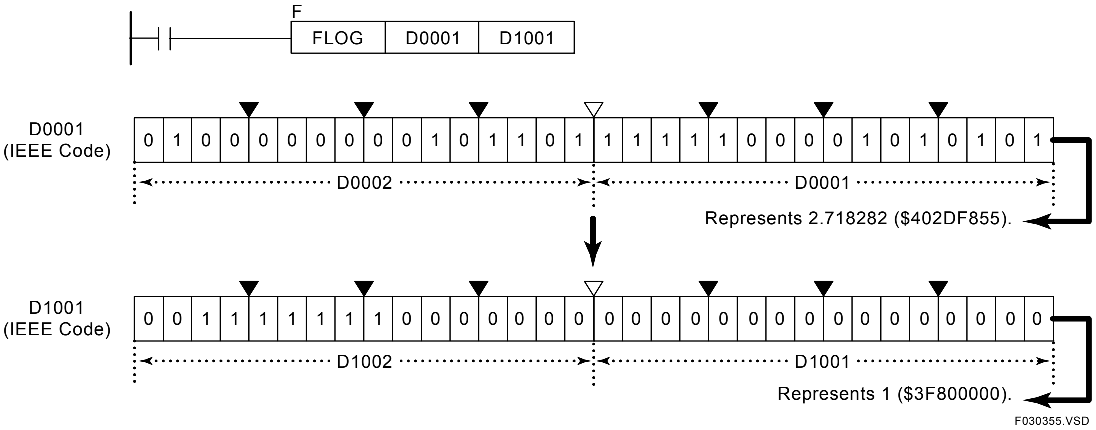
s : Real data whose logarithm is to be calculated

d : Execution result

The single-precision floating-point numbers are represented in the IEEE format.

● Example

LOG_e 2.718282=1



D0001 (IEEE Code)

0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 1 1 1 1 1 1 0 0 0 0 1 0 1 0 1 0 1

D0002 | D0001

Represents 2.718282 ($402DF855).

D1001 (IEEE Code)

0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

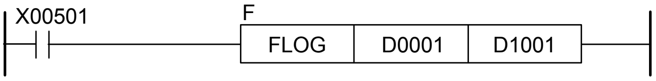
D1002 | D1001

Represents 1 ($3F800000).

Figure 3.3.61 Example of a Logarithm Calculation

■ Programming Example

The sample code shown below calculates the natural logarithm (logarithm to the base e) of the real data in location D0001 to D0002 and loads the execution result into the location D1001 to D1002 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FLOG	D0001	D1001			

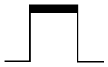
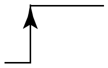
F030356.VSD

Figure 3.3.62 Example of a Logarithm Program

3.3.26 EXP (FEXP)

F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.3.67 Exponent

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	914	EXP (Exponent)	FEXP	F FEXP	✓	—		4	32 bit	—
	914P		↑FEXP	↑F FEXP				5		

■ Parameter

EXP(Exponent)

F

FEXP	s	d

F030357.VSD

s : Data whose exponent is to be calculated or device number of the first device storing the data whose exponent is to be calculated

d : Device number of the first device storing the execution result (exponent)

Both s and d are represented in the IEEE single-precision floating-point format (32 bits).

■ Available Devices

Table 3.3.68 Devices Available for the Exponent Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Exponent instruction calculates the exponent (e to the power of s) of a given real number (single-precision floating point). The equation for and outline of the exponent calculation is shown in the figure below.

$d = e^{s}$

s : Real data whose exponent is to be calculated

d : Execution result

The single-precision floating-point numbers are represented in the IEEE format.

● Example

e^{1}=2.718282

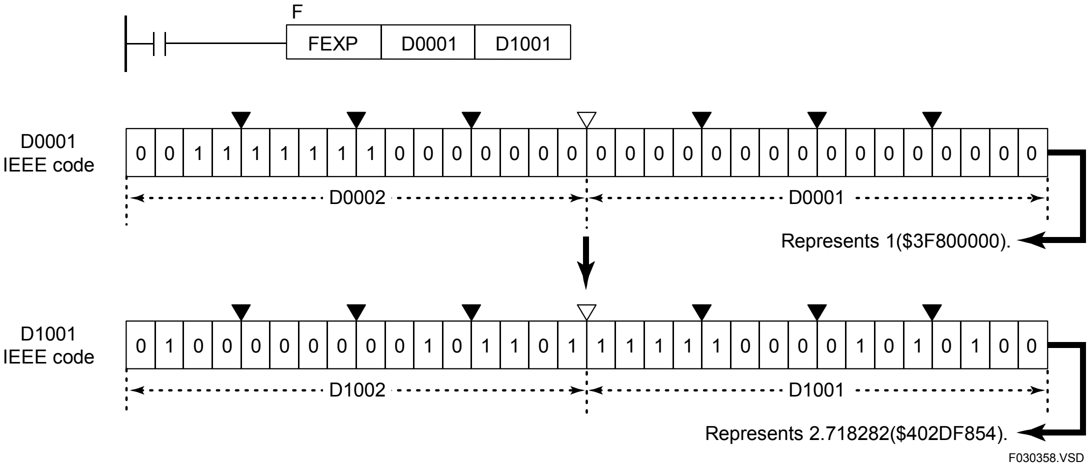
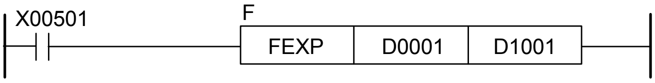


Figure 3.3.63 Example of an Exponent Calculation

■ Programming Example

The sample code shown below calculates the exponent (to the base e) of the real data in location D0001 to D0002 and loads the execution result into the location D1001 to D1002 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FEXP	D0001	D1001			

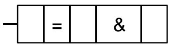
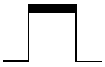
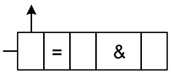
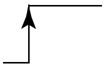
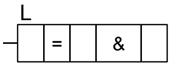
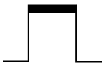
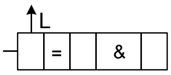
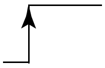
F030359.VSD

Figure 3.3.64 Example of an Exponent Program

3.4 Logical Instructions

3.4.1 Logical AND (CAL), Logical AND Long-word (CAL L)

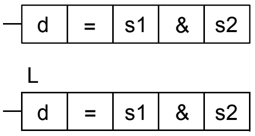
Table 3.4.1 Logical AND, Logical AND Long-word

Classifi-cation	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	20	Logical AND	CAL		✓	—		4	16 bit	—
	20P		↑CAL					5		
	20L	Logical AND Long-word	CAL L		✓	—		4	32 bit	—
	20LP		↑CAL L					5		

■ Parameter

Logical AND

Logical AND Long-word



F030360.VSD

d : Device number of the first device for storing the execution result
& : Logical AND operator
s1, s2 : Operand data or device number of the first device storing the operand data

■ Available Devices

Table 3.4.2 Devices Available for the Logical AND and Logical AND Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
s1	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value (may not be used as a long-word parameter)

*3: Counter current value (may not be used as a long-word parameter)

■ Function

The Logical AND and Logical AND Long-word instructions perform a logical AND operation on 16- and 32-bit data, respectively, and load the result into the specified devices. Use the Logical AND instruction to perform the logical AND on 16-bit data and the Logical AND Long-word instruction to perform the logical AND on 32-bit data. Neither Logical AND nor Logical AND Long-word instructions can perform a logical AND operation on a mixture of 16- and 32-bit data.

The numbers of bits in the execution results obtained through the Logical AND and Logical AND Long-word instructions are summarized in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.4.3 Numbers of Bits Resulting from of Logical AND Operations

Specification Item	Instruction	
	Logical AND (1-word instruction)	Logical AND long-word (2-word instruction)
Number of bits in execution result	16 bits	32 bits
Device where the execution result is placed	d	d+1, d

The operands on which a logical operation is to be performed may be either of binary, BCD, or a mixture of both types. In either case, the operands are handled as collections of bit data. Consequently, no BCD range check is made before and after the logical operation.

● Example of a logical AND operation

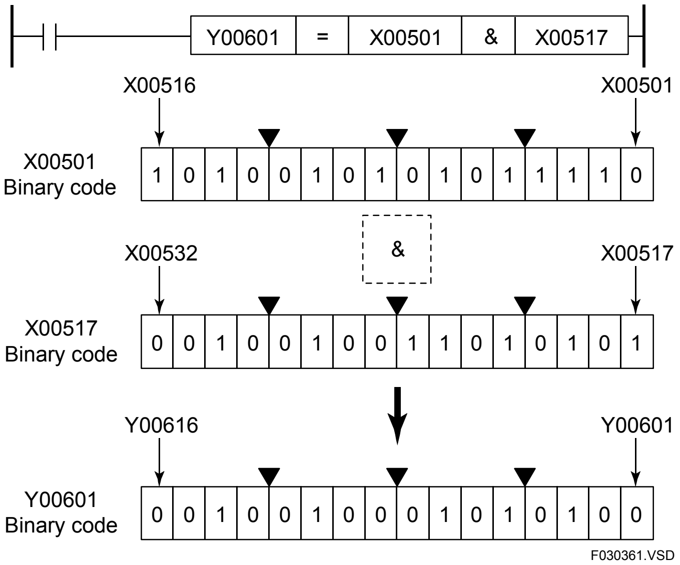
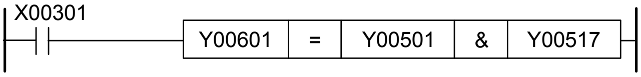


Figure 3.4.1 Example of a Logical AND Operation

■ Programming Example

The sample code shown below performs a logical AND on 16 bits of devices starting at Y00501 and 16 bits of devices starting at Y00517 and assigns the result to 16 bits of devices starting at Y00601 if X00301 is on.



Line No.	Instruction	Operands				
0001	LD	X00301				
0002	CAL	Y00601	=	Y00501	&	Y00517

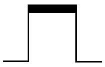
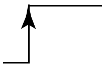
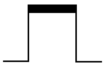
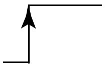
Note: The "=" operand need not be entered as it is automatically displayed when a CAL instruction is entered.

F030362.VSD

Figure 3.4.2 Example of a Logical AND Program

3.4.2 Logical OR (CAL), Logical OR Long-word (CAL L)

Table 3.4.4 Logical OR, Logical OR Long-word

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	20	Logical OR	CAL	┤ = ¦	✓	—		4	16 bit	—
	20P		↑CAL	↑ ┤ = ¦				5		
	20L	Logical OR Long-word	CAL L	L ┤ = ¦	✓	—		4	32 bit	—
	20LP		↑CAL L	↑L ┤ = ¦				5		

■ Parameter

Logical OR ┤ d = s1 ¦ s2

Logical OR Long-word

L
┤ d = s1 ¦ s2

F030401.VSD

d : Device number of the first device for storing the execution result
¦ : Logical OR operator
s1, s2 : Operand data or device numbers of the first devices storing the operand data

■ Available Devices

Table 3.4.5 Devices Available for the Logical OR and Logical OR Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
s1	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value (may not be used as a long-word parameter)
*3: Counter current value (may not be used as a long-word parameter)

■ Function

The Logical OR and Logical OR Long instructions perform a logical OR operation on 16- and 32-bit data, respectively and load the result into the specified devices. Use the Logical OR instruction to perform the logical OR on 16-bit data and the Logical OR Long-word instruction to perform the logical OR on 32-bit data. Neither Logical OR nor Logical OR Long-word instructions can perform a logical OR operation on a mixture of 16- and 32-bit data.

The numbers of bits in the execution results obtained through the Logical OR and Logical OR Long-word instructions are summarized in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.4.6 Numbers of Bits Resulting from of Logical OR Operations

Specification Item	Instruction	
	Logical OR (1-word instruction)	Logical OR long-word (2-word instruction)
Number of bits in execution result	16 bits	32 bits
Device where the execution result is placed	d	d+1, d

The operands on which a logical operation is to be performed may be either of binary, BCD, or a mixture of both types. In either case, the operands are handled as collections of bit data. Consequently, no BCD range check is made before and after the logical operation.

● Example of a logical OR operation

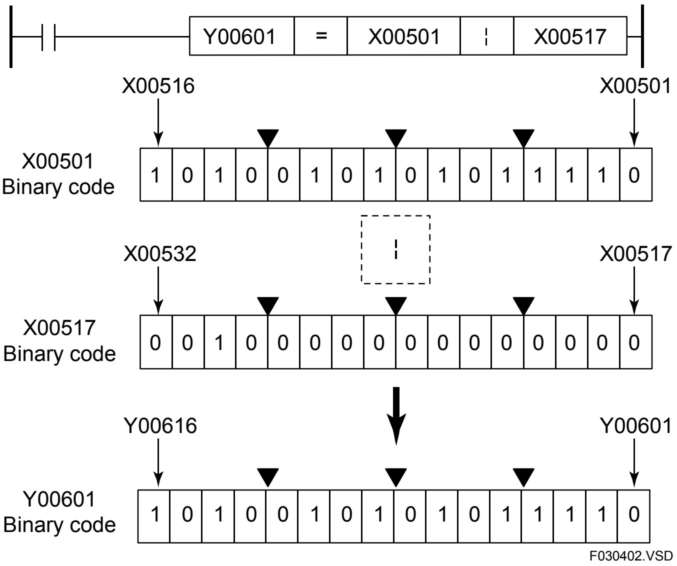
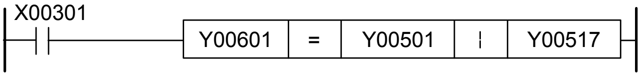


Figure 3.4.3 Example of a Logical OR Operation

■ Programming Example

The sample code shown below performs a logical OR on 16 bits of devices starting at Y00501 and 16 bits of devices starting at Y00517 and assigns the result to 16 bits of devices starting at Y00601 if X00301 is on.



Line No.	Instruction	Operands				
0001	LD	X00301				
0002	CAL	Y00601	=	Y00501	¦	Y00517

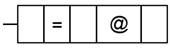
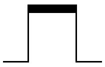
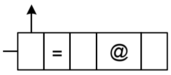
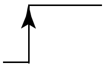
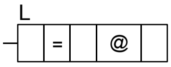
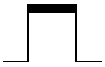
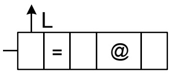
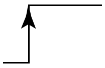
Note: The "=" operand need not be entered as it is automatically displayed when a CAL instruction is entered.

F030403.VSD

Figure 3.4.4 Example of a Logical OR Program

3.4.3 Logical XOR (CAL), Logical XOR Long-word (CAL L)

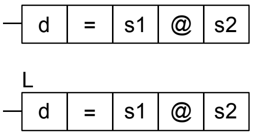
Table 3.4.7 Logical XOR, Logical XOR Long-word

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	20	Logical XOR	CAL		✓	—		4	16 bit	—
	20P		↑CAL					5		
	20L	Logical XOR Long-word	CAL L		✓	—		4	32 bit	—
	20LP		↑CAL L					5		

■ Parameter

Logical XOR

Logical XOR Long-word



F030404.VSD

d : Device number of the first device for storing the execution result
@ : Logical XOR operator
s1, s2 : Operand data or device number of the first devices storing the operand data

■ Available Devices

Table 3.4.8 Devices Available for the Logical XOR and Logical XOR Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1*	✓*1	✓*1	✓		Yes	Yes
s1	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value (may not be used as a long-word parameter)
*3: Counter current value (may not be used as a long-word parameter)

■ Function

The Logical XOR and Logical XOR Long-word instructions perform a logical XOR operation on 16- and 32-bit data, respectively, and load the result into the specified devices. Use the Logical XOR instruction to perform the logical XOR on 16-bit data and the Logical XOR Long-word instruction to perform the logical XOR on 32-bit data. Neither Logical XOR nor Logical XOR Long-word instructions can perform a logical XOR operation on a mixture of 16- and 32-bit data.

The numbers of bits in the execution results obtained through the Logical XOR and Logical XOR Long-word instructions are summarized in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.4.9 Numbers of Bits Resulting from of Logical XOR Operations

Specification Item	Instruction	
	Logical XOR (1-word instruction)	Logical XOR long-word (2-word instruction)
Number of bits in execution result	16 bits	32 bits
Device where the execution result is placed	d	d+1, d

The operands on which a logical operation is to be performed may be either of binary, BCD, or a mixture of both types. In either case, the operands are handled as collections of bit data. Consequently, no BCD range check is made before and after the logical operation.

● Example of a logical XOR operation

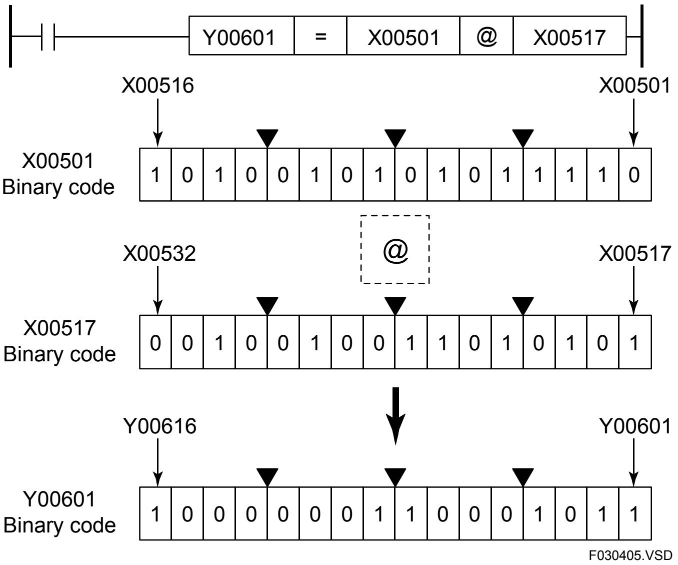
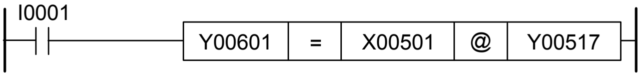


Figure 3.4.5 Example of a Logical XOR Operation

■ Programming Example

The sample code shown below performs a logical XOR on 16 bits of devices starting at Y00501 and 16 bits of devices starting at Y00517 and assigns the result to 16 bits of devices starting at Y00601 if I0001 is on.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	CAL	Y00601	=	X00501	@	X00517

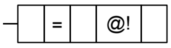
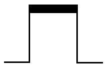
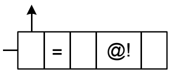
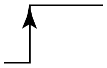
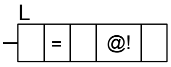
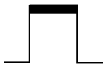
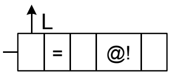
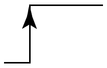
Note: The "=" operand need not be entered as it is automatically displayed when a CAL instruction is entered.

F030406.VSD

Figure 3.4.6 Example of a Logical XOR Program

3.4.4 Logical NXOR (CAL), Logical NXOR Long-word (CAL L)

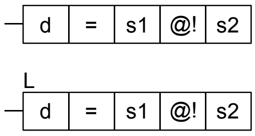
Table 3.4.10 Logical NXOR, Logical NXOR Long-word

Classifi-cation	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	20	Logical NXOR	CAL		✓	—		4	16 bit	—
	20P		↑CAL					5		
	20L	Logical NXOR Long-word	CAL L		✓	—		4	32 bit	—
	20LP		↑CAL L					5		

■ Parameter

Logical NXOR

Logical NXOR Long-word



F030407.VSD

d : Device number of the first device for storing the execution result
@ ! : Logical NXOR operator
s1, s2 : Operand data or device number of the first devices storing the operand data

■ Available Devices

Table 3.4.11 Devices Available for the Logical NXOR and Logical NXOR Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1*	✓*1	✓*1	✓		Yes	Yes
s1	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value (may not be used as a long-word parameter)
*3: Counter current value (may not be used as a long-word parameter)

■ Function

The Logical NXOR and Logical NXOR instructions perform a logical NXOR operation on 16- and 32-bit data, respectively, and load the result into the specified devices. Use the Logical NXOR instruction to perform the logical NXOR on 16-bit data and the Logical NXOR Long-word instruction to perform the logical NXOR on 32-bit data. Neither Logical NXOR nor Logical NXOR Long-word instructions can perform a logical NXOR operation on a mixture of 16- and 32-bit data.

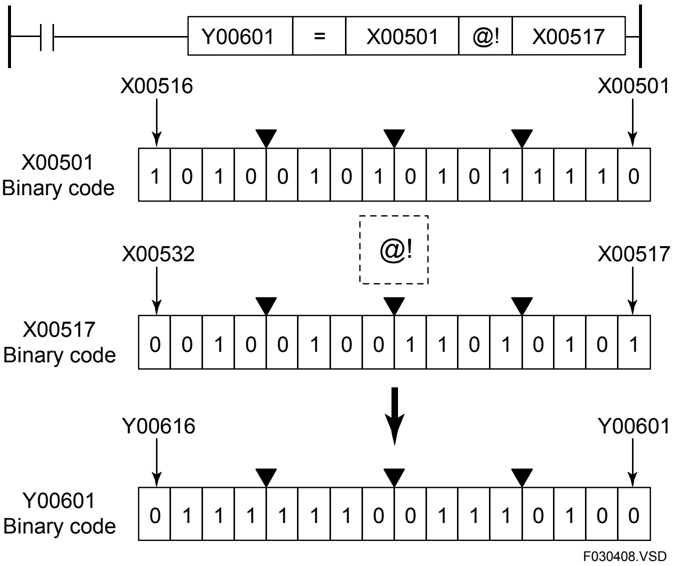
The numbers of bits in the execution results obtained through the Logical NXOR and Logical NXOR Long-word instructions are summarized in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.4.12 Numbers of Bits Resulting from of Logical NXOR Operations

Specification Item	Instruction	
	Logical NXOR (1-word instruction)	Logical NXOR long-word (2-word instruction)
Number of bits in execution result	16 bits	32 bits
Device where the execution result is placed	d	d+1, d

The operands on which a logical operation is to be performed may be either of binary, BCD, or a mixture of both types. In either case, the operands are handled as collections of bit data. Consequently, no BCD range check is made before and after the logical operation.

● Example of a logical NXOR operation



X00501 Binary code: 1 0 1 0 0 1 0 1 0 1 0 1 1 1 1 0

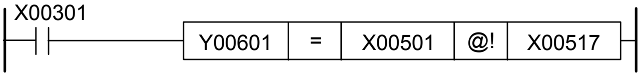
X00517 Binary code: 0 0 1 0 0 1 0 0 1 1 0 1 0 1 0 1

Y00601 Binary code: 0 1 1 1 1 1 1 0 0 1 1 1 0 1 0 0

Figure 3.4.7 Example of a Logical NXOR Operation

■ Programming Example

The sample code shown below performs a logical NXOR on 16 bits of devices starting at Y00501 and 16 bits of devices starting at Y00517 and assigns the result to 16 bits of devices starting at Y00601 if X00301 is on.



Line No.	Instruction	Operands				
0001	LD	X00301				
0002	CAL	Y00601	=	X00501	@!	X00517

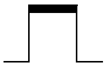
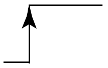
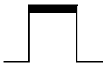
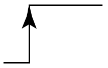
Note: The "=" operand need not be entered as it is automatically displayed when a CAL instruction is entered.

F030409.VSD

Figure 3.4.8 Example of a Logical NXOR Program

3.4.5 Two's Complement (NEG), Two's Complement Long-word (NEG L)

Table 3.4.13 Two's Complement, Two's Complement Long-word

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	21	Two's Complement	NEG	NEG	✓	—		2	16 bit	—
	21P		↑NEG	↑ NEG				3		
	21L	Two's Complement Long-word	NEG L	L NEG	✓	—		2	32 bit	—
	21LP		↑NEG L	↑L NEG				3		

■ Parameter

Two's Complement —NEG d

Two's Complement Long-word —L NEG d

F030410.VSD

d : Device number of the first device storing the operand and device number of the first device for storing the execution result

■ Available Devices

Table 3.4.14 Devices Available for the Two's Complement and Two's Complement Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1*	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value (may not be used as a long-word parameter)

*3: Counter current value (may not be used as a long-word parameter)

■ Function

The Two's Complement and Two's Complement Long-word instructions calculate the two's complement of 16- and 32-bit data, respectively, and load the result into the specified devices. The result is placed into the devices that carry the operand data.

Use the Two's Complement instruction to perform the two's complement operation on 16-bit data and the Two's Complement Long-word instruction to perform the two's complement operation on 32-bit data.

The numbers of bits in the execution results obtained through the Two's Complement and Two's Complement Long-word instructions are summarized in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.4.15 Numbers of Bits Resulting from of Two's Complement Operations

Specification Item	Instruction	
	Two's Complement (1-word instruction)	Two's Complement long-word (2-word instruction)
Number of bits in execution result	16 bits	32 bits
Device where the execution result is placed	d	d+1, d

● Example of a two's complement operation

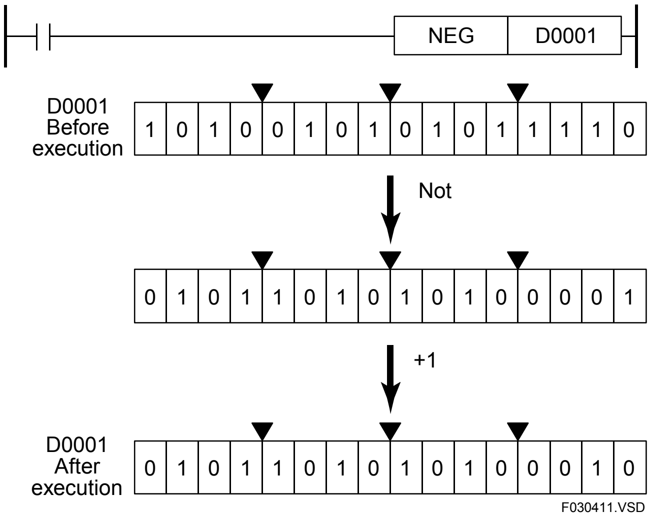
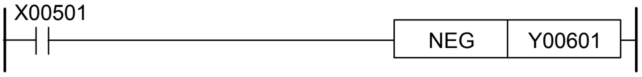


Figure 3.4.9 Example of a Two's Complement Operation

■ Programming Example

The sample code shown below converts the value carried by 16 bits of devices starting at Y00601 to its two's complement if X00501 is on.



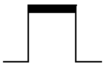
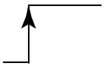
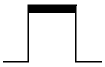
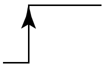
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	NEG	Y00601				

F030412.VSD

Figure 3.4.10 Example of a Two's Complement Program

3.4.6 Not (NOT), Not Long-word (NOT L)

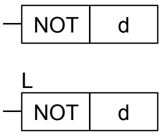
Table 3.4.16 Not, Not Long-word

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	22	Not	NOT	NOT	✓	—		2	16 bit	—
	22P		↑NOT	↑ NOT				3		
	22L	Not Long-word	NOT L	L NOT	✓	—		2	32 bit	—
	22LP		↑NOT L	↑L NOT				3		

■ Parameter

Not

Not Long-word



F030413.VSD

d : Device number of the first device storing the operand and device number of the first device for storing the execution result

■ Available Devices

Table 3.4.17 Devices Available for the Not and Not Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1*	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value (may not be used as a long-word parameter)

*3: Counter current value (may not be used as a long-word parameter)

■ Function

The Not and Not Long-word instructions calculate the Not of 16- and 32-bit data, respectively, and load the result into the specified devices. The result is placed into the devices that carry the operand data.

Use the Not instruction to perform the Not operation on 16-bit data and the Not Long-word instruction to perform the Not operation on 32-bit data.

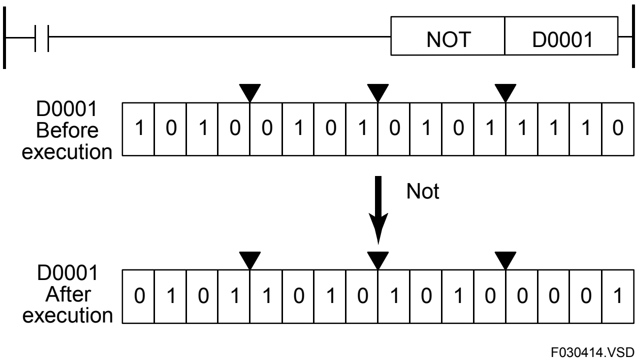
The numbers of bits in the execution results obtained through the Not and Not Long-word instructions are summarized in the following table. The execution result is stored in the location starting at the first device designated by the parameter d.

Table 3.4.18 Numbers of Bits Resulting from of Not Operations

Specification Item	Instruction	
	Not (1-word instruction)	Not Long (2-word instruction)
Number of bits in execution result	16 bits	32 bits
Device where the execution result is placed	d	d+1, d

The operands on which a logical operation is to be performed may be either of binary, BCD, or a mixture of both types. In either case, the operands are handled as collections of bit data. Consequently, no BCD range check is made before and after the logical operation.

● Example of a Not operation

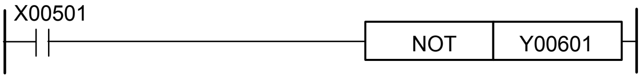


D0001 Before execution	1	0	1	0	0	1	0	1	0	1	0	1	1	1	1	0
D0001 After execution	0	1	0	1	1	0	1	0	1	0	1	0	0	0	0	1

Figure 3.4.11 Example of a Not Operation

■ Programming Example

The sample code shown below inverts the value carried by 16 bits of devices starting at Y00601 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	NOT	Y00601				

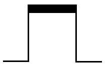
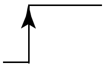
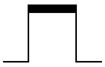
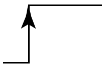
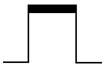
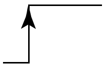
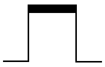
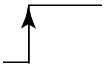
F030415.VSD

Figure 3.4.12 Example of a Not Program

3.5 Rotate Instructions

3.5.1 Rotate (RROT, LROT), Rotate Long-word (RROT L, LROT L)

Table 3.5.1 Rotate, Rotate Long-word

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	30	Right Rotate	RROT	RROT	✓	—		3	16 bit	✓
	30P		↑RROT	↑ RROT				4		
	30L	Right Rotate Long-word	RROT L	L RROT	✓	—		3	32 bit	✓
	30LP		↑RROT L	↑L RROT				4		
	31	Left Rotate	LROT	LROT	✓	—		3	16 bit	✓
	31P		↑LROT	↑ LROT				4		
	31L	Left Rotate Long-word	LROT L	L LROT	✓	—		3	32 bit	✓
	31LP		↑LROT L	↑L LROT				4		

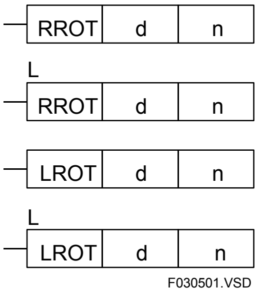
■ Parameter

Right Rotate

Right Rotate Long-word

Left Rotate

Left Rotate Long-word



d : Device number of the first device storing the operand to be rotated and for storing the rotation result

n : Device number of the first device storing the bit count by which rotation is to be performed[*1]
Right and Left Rotate : $1 \leq n \leq 16$
Right and Left Rotate Long : $1 \leq n \leq 32$

*1: n is handled as a word even in a 32-bit (long-word) instruction.

■ Available Devices

Table 3.5.2 Devices Available for the Rotate and Rotate Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
n	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value (may not be used as a long-word parameter)

*3: Counter current value (may not be used as a long-word parameter)

■ Function

The Rotate and Rotate Long-word instructions rotate 16- and 32-bit data n bits to the right or left, respectively.

Use the Right or Left Rotate instruction to rotate 16-bit data and the Right or Left Rotate Long-word instruction to rotate 32-bit data. The state of the carry flag is changed depending on the result of rotation.

The data d to be rotated may be either of binary or BCD type. In either case, the operand is handled as a collection of bit data. Consequently, no BCD range check is made before and after the rotation operation.

● Examples of rotate operations

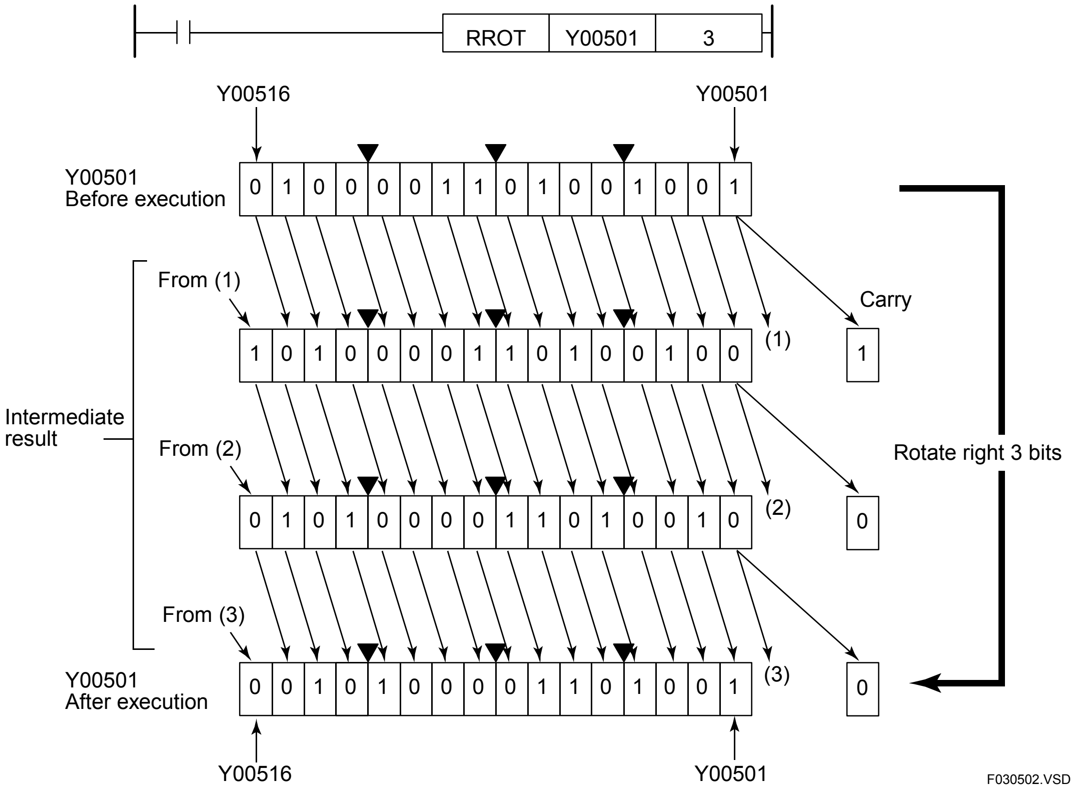


Figure 3.5.1 Example of a (Right) Rotate Operation

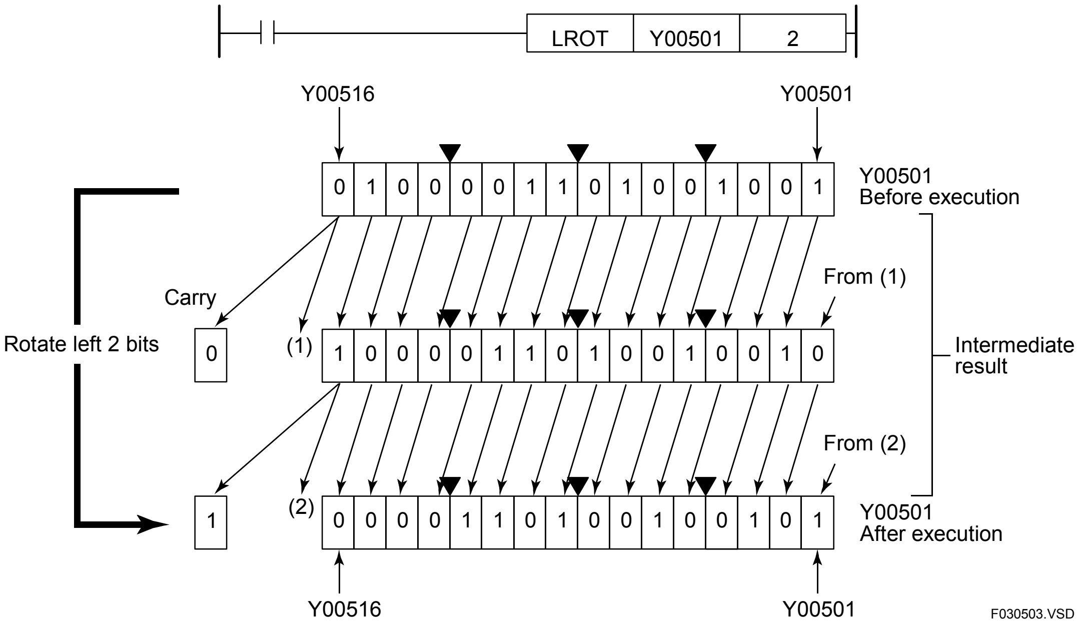
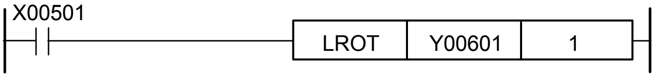


Figure 3.5.2 Example of a (Left) Rotate Operation

■ Programming Example

The sample code shown below rotates 16 bits starting at Y00601 1 bit to the left if X00501 is on.



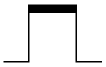
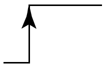
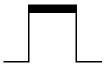
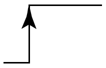
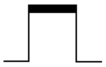
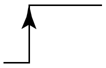
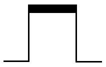
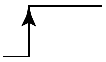
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	LROT	Y00601	1			

F030504.VSD

Figure 3.5.3 Example of a Rotate Program

3.5.2 Rotate with Carry (RROTC, LROTC), Rotate Long-word with Carry (RROTC L, LROTC L)

Table 3.5.3 Rotate with Carry, Rotate Long-word with Carry

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	130	Right Rotate With Carry	RROTC	RROTC	✓	—		3	16 bit	✓
	130P		↑RROTC	↑ RROTC				4		
	130L	Right Rotate Long-word With Carry	RROTC L	L RROTC	✓	—		3	32 bit	✓
	130LP		↑RROTC L	↑L RROTC				4		
	131	Left Rotate With Carry	LROTC	LROTC	✓	—		3	16 bit	✓
	131P		↑LROTC	↑ LROTC				4		
	131L	Left Rotate Long-word With Carry	LROTC L	L LROTC	✓	—		3	32 bit	✓
	131LP		↑LROTC L	↑L LROTC				4		

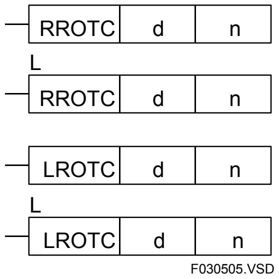
■ Parameter

Right Rotate With Carry

Right Rotate Long-word With Carry

Left Rotate With Carry

Left Rotate Long-word With Carry



d : Device number of the first device storing the operand to be rotated with Carry and the Device number of the first for storing the rotation result

n : Device number of the first device storing the bit count by which bits are to be rotated[*1]

Right and Left Carry	: 1 ≤ n ≤ 16
Right and Left Carry Long-word	: 1 ≤ n ≤ 32

*1: n is handled as a word even in a 32-bit (long-word) instruction.

■ Available Devices

Table 3.5.4 Devices Available for the Rotate with Carry and Rotate Long-word with Carry Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
n	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value (may not be used as a long-word parameter)

*3: Counter current value (may not be used as a long-word parameter)

■ Function

The Rotate with Carry and Rotate Long-word with Carry instructions rotate 16- and 32-bit data with carry n bits to the right or left, respectively.

Use the Right or Left Rotate with Carry instruction to Rotate with Carry 16-bit data and the Right or Left Rotate Long-word with Carry instruction to Rotate with Carry 32-bit data. The state of the carry flag is changed depending on the result of rotation.

The data d to be rotated may be either of binary or BCD type. In either case, the operand is handled as a collection of bit data. Consequently, no BCD range check is made before and after the rotation operation.

In a right rotate with a carry, the state of the carry flag established before the execution of the instruction is fed to the most significant bit position and the nth bit counted from the least significant bit is placed into the carry flag after the execution of the instruction. The operations are reversed with a left rotate with a carry.

The CSET (Carry Set) or CRST (Carry Reset) instruction may be used to change the state of the carry flag before executing the instruction (before rotation). Since the carry flag is allocated to a special relay M188, it may be set or reset with the SET (Set) or RST (Reset) instruction.

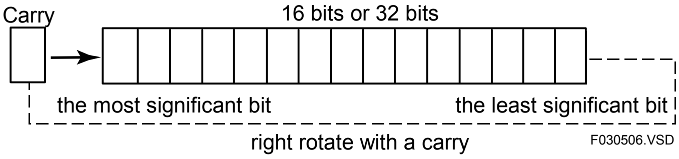


Figure 3.5.4 Example of a Rotation with Carry

● Examples of rotate operations

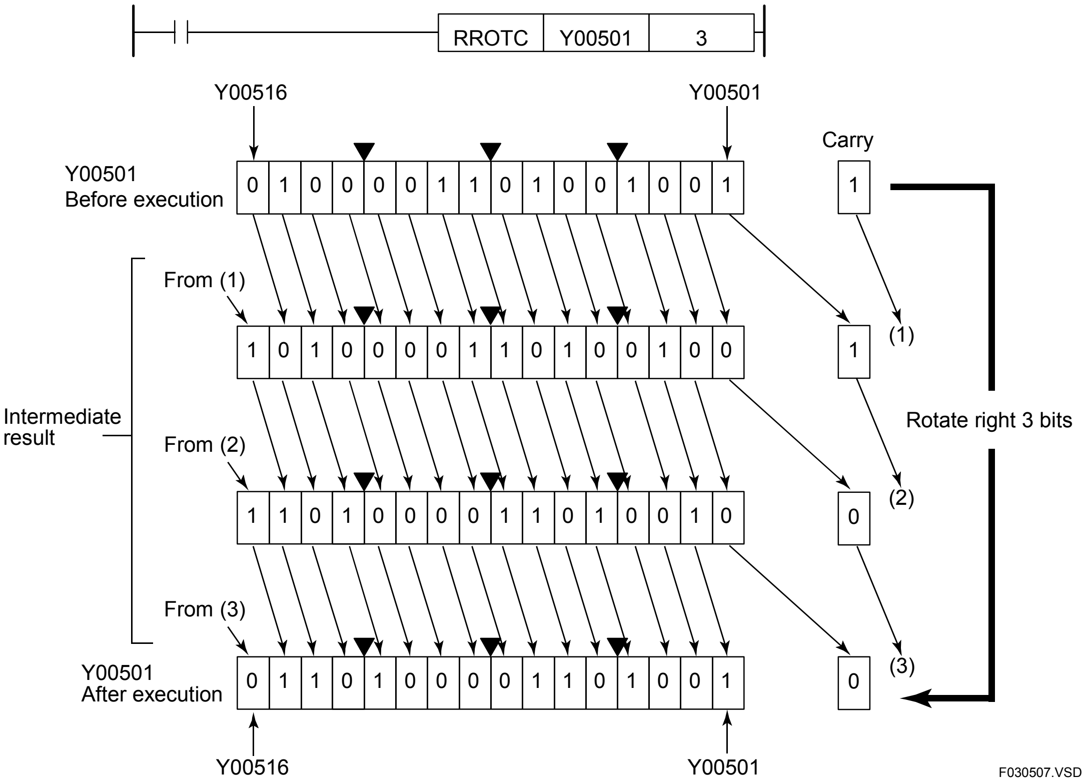


Figure 3.5.5 Example of a (Right) Rotate with Carry Operation

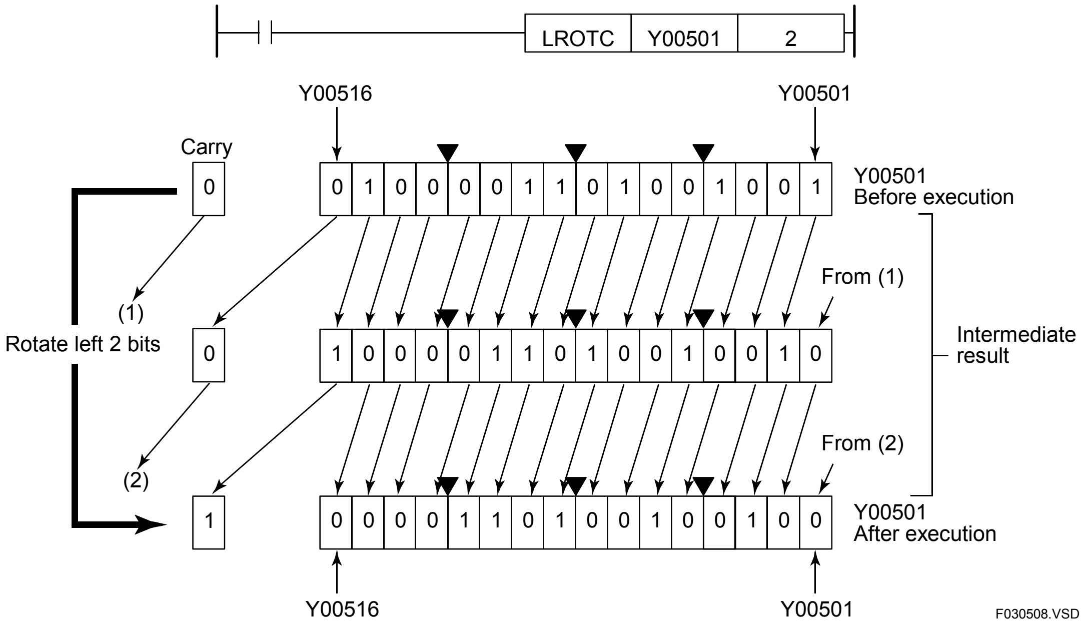
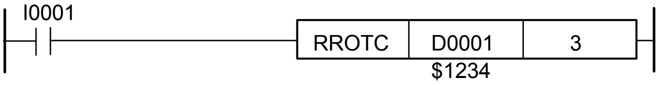


Figure 3.5.6 Example of a (Left) Rotate with Carry Operation

■ Programming Example

The sample code shown below rotates D0001 ($1234) 3 bits to the right with carry if I0001 is on.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	RROTC	D0001	3			

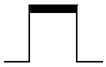
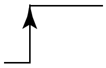
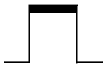
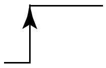
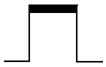
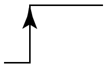
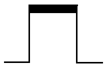
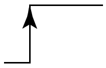
F030509.VSD

Figure 3.5.7 Example of a Rotate with Carry Program

3.6 Shift Instructions

3.6.1 Shift (RSFT, LSFT), Shift Long-word (RSFT L, LSFT L)

Table 3.6.1 Shift, Shift Long-word

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	32	Right Shift	RSFT	RSFT	✓	—		3	16 bit	✓
	32P		↑RSFT	↑ RSFT				4		
	32L	Right Shift Long-word	RSFT L	L RSFT	✓	—		3	32 bit	✓
	32LP		↑RSFT L	↑L RSFT				4		
	33	Left Shift	LSFT	LSFT	✓	—		3	16 bit	✓
	33P		↑LSFT	↑ LSFT				4		
	33L	Left Shift Long-word	LSFT L	L LSFT	✓	—		3	32 bit	✓
	33LP		↑LSFT L	↑L LSFT				4		

■ Parameter

Right Shift		RSFT	d	n
Right Shift Long-word	L	RSFT	d	n
Left Shift		LSFT	d	n
Left Shift Long-word	L	LSFT	d	n

F030601.VSD

d : Device number of the first device storing the operand to be shifted and the Device number of the first for storing the shifted result

n : Device number of the first device storing the bit count by which bits are to be shifted[*1]

Right and Left Shift	: $1 \leq n \leq 16$
Right and Left Shift Long	: $1 \leq n \leq 32$

*1 : n is handled as a word even in a 32-bit (long-word) instruction.

■ Available Devices

Table 3.6.2 Devices Available for the Shift and Shift Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
n	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value (may not be used as a long-word parameter)

*3: Counter current value (may not be used as a long-word parameter)

■ Function

The Shift and Shift Long-word instructions shift 16- and 32-bit data n bits to the right or left, respectively. The last bit that shifted out of the devices is loaded into the carry bit position.

Use the Right or Left Shift instruction to shift 16-bit data and the Right or Left Shift Long-word instruction to shift 32-bit data.

The data d to be shifted may be either of binary or BCD type. In either case, the operand is handled as a collection of bit data. Consequently, no BCD range check is made before and after the shift operation.

● Examples of shift operations

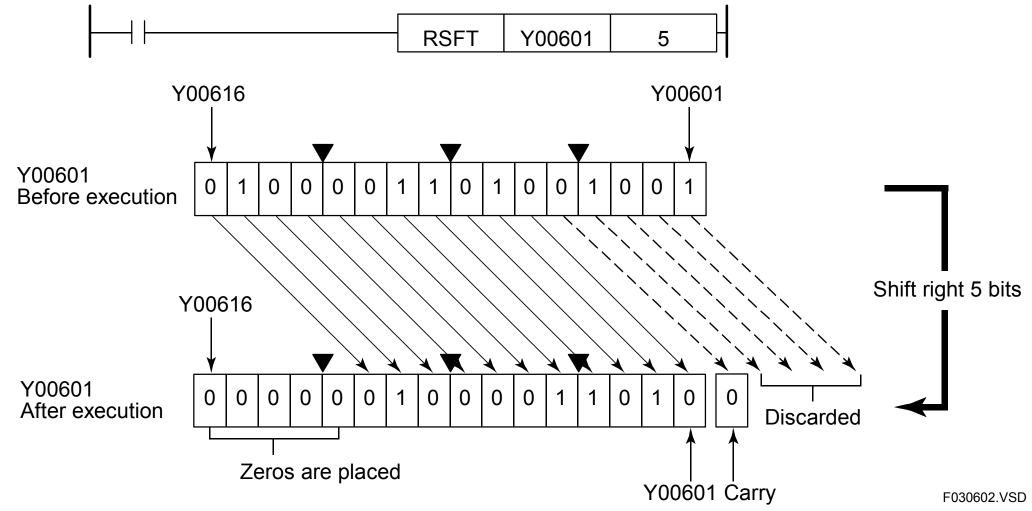


Figure 3.6.1 Example of a (Right) Shift Operation

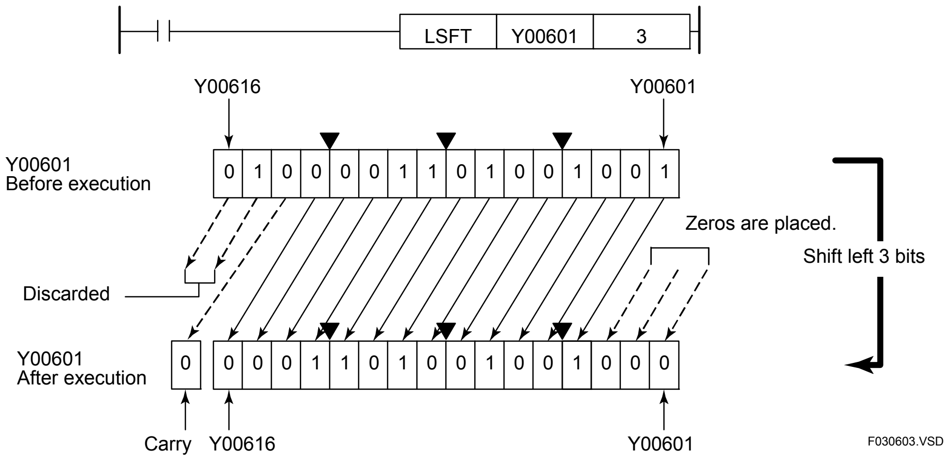
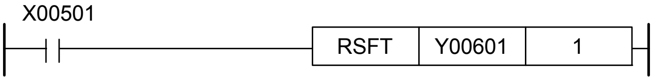


Figure 3.6.2 Example of a (Left) Shift Operation

■ Programming Example

The sample code shown below shifts 16 bits starting at Y00601 1 bit to the right if X00501 is on.



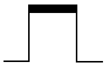
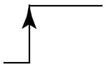
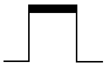
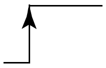
Line No.	Instruction	Operands			
0001	LD	X00501			
0002	RSFT	Y00601	1		

F030604.VSD

Figure 3.6.3 Example of a Shift Program

3.6.2 Shift m-bit Data by n Bits (RSFTN, LSFTN)

Table 3.6.3 Shift m-bit Data by n Bits

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	132	Right Shift m-bit Data by n bits	RSFTN	RSFTN	✓	—		4	—	✓
	132P		↑RSFTN	↑RSFTN				5		
	133	Left Shift m-bit Data by n bits	LSFTN	LSFTN	✓	—		4	—	✓
	133P		↑LSFTN	↑LSFTN				5		

■ Parameter

Right Shift m-bit Data by n Bits	RSFTN	d	n1	n2
Left Shift m-bit Data by n Bits	LSFTN	d	n1	n2

F030605.VSD

d : Device number of the first device storing the operand to be shifted
n1 : Number of bits to be shifted (m bits long)
n2 : Number of bits by which data is to be shifted
n1 and n2 are handled as a word.

■ Available Devices

Table 3.6.4 Devices Available for the Shift m-bit Data by n Bits Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
n1	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
n2	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Current timer value (may not be used as d if m is 17 or greater)

*3: Current counter value (may not be used as d if m is 17 or greater)

Function

A Shift m-bit Data n Bits instruction shifts m-bit data n bits to the right or left. The state of the carry flag is changed according to the result of shift. The carry flag is allocated to the special relay M188.

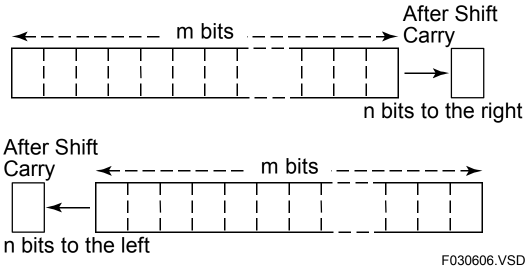


Figure 3.6.4 Shift m-bit Data by n Bits

Examples of a shift operation

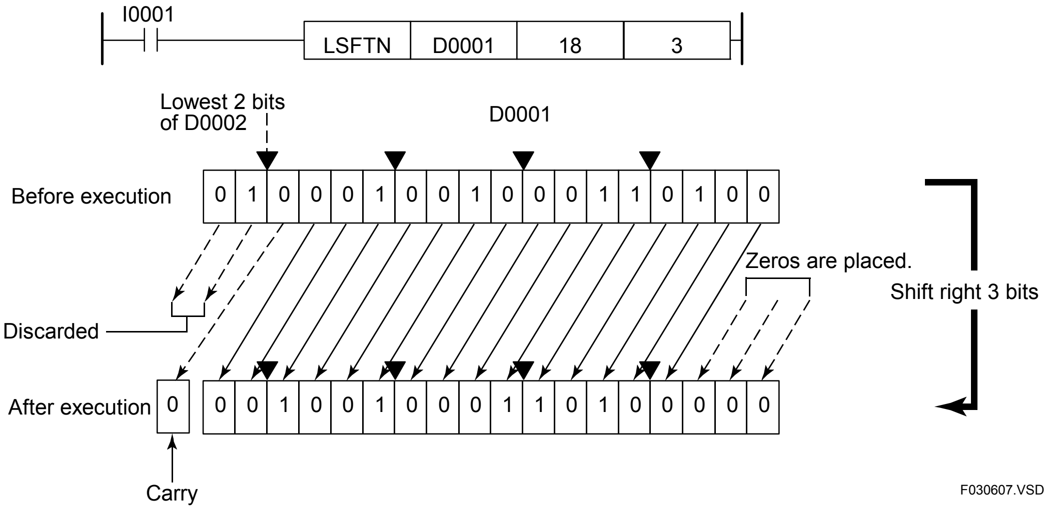
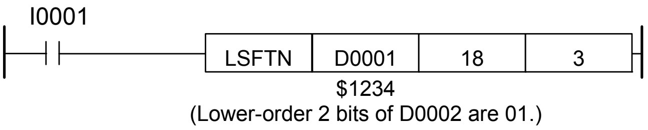


Figure 3.6.5 Example of a Shift m-bit Data n Bits Operation

If the device whose data is to be shifted is a register, the m bits starting at bit 0 are shifted. If the value of m is 17 or greater, shifting proceeds to the lowest bit side of the next device. The values of the bits that are not shifted (bits 2 to15 of D0002 in this example) remain unchanged.

Programming Example

The sample code shown below shifts the 16 bits of D0001 and the lowest 2 bits of D0002 (18 bits in total) to the left by 3 bits if I0001 is on.



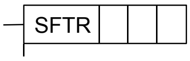
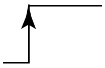
Line No.	Instruction	Operands				
0001	LD	I0001				
0002	LSFTN	D0001	18	3		

F030608.VSD

Figure 3.6.6 Example of a Shift m-bit Data n Bits Program

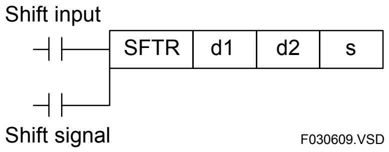
3.6.3 Shift Register (SFTR)

Table 3.6.5 Shift Register

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	34	Shift Register	SFTR		✓	—		4	n bit	✓

■ Parameter

Shift Register



d1 : Device number of the device identifying the beginning of the shift range

d2 : Device number of the device identifying the end of the shift range (the shift result is also placed in the same device.)

s : Direction of shift operation (1 = Right / 0 = Left)

■ Available Devices

Table 3.6.6 Devices Available for the Shift Register Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d1		✓	✓	✓*1	✓*1	✓*1			✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
d2		✓	✓	✓*1	✓*1	✓*1			✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
s	✓	✓	✓	✓*1	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Current timer value (may not be used for shifts of multiple words. Available only when d1 = d2.)

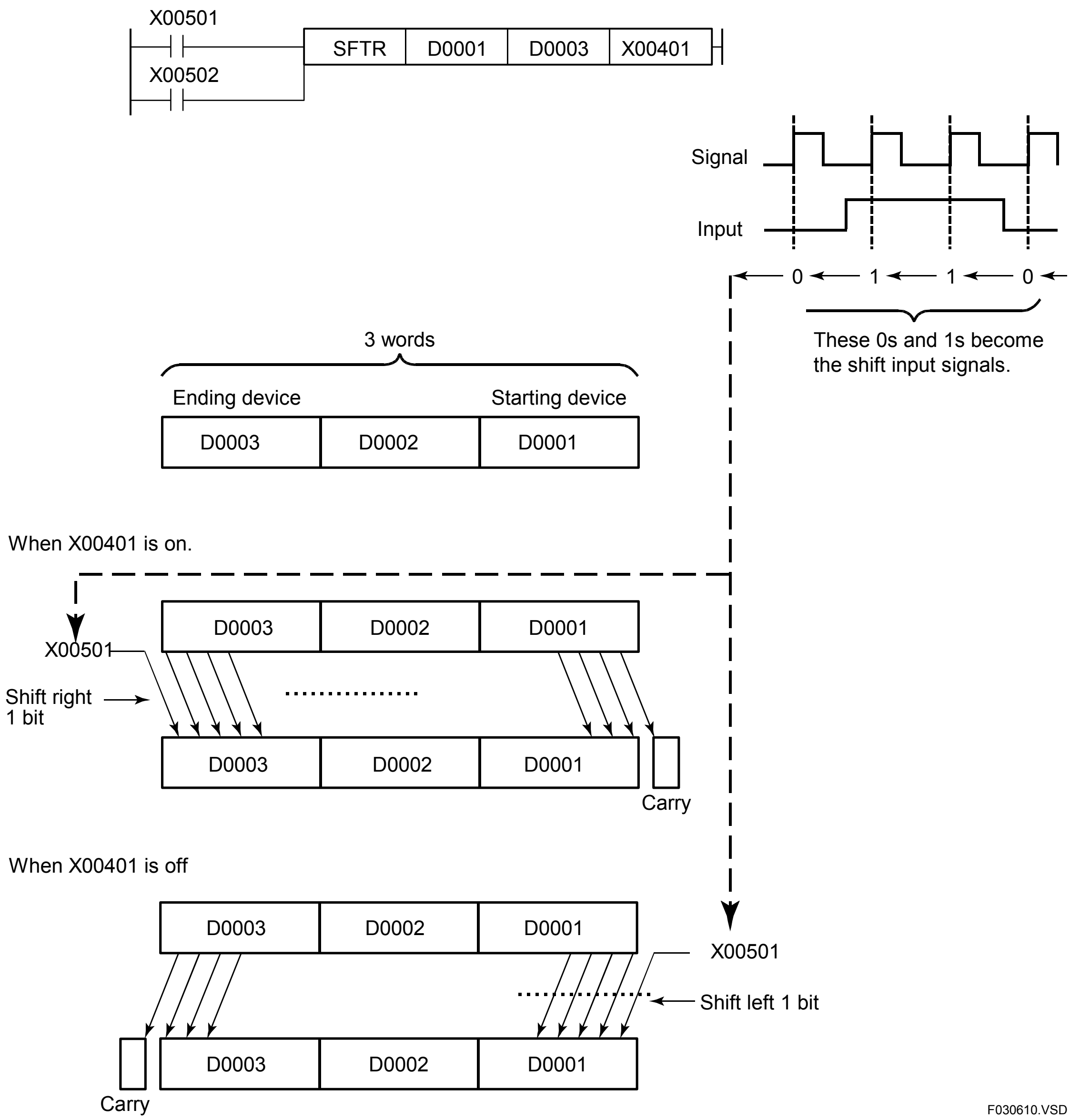
*3: Current counter value (may not be used for shifts of multiple words. Available only when d1 = d2.)

■ Function

The Shift Register instruction shifts n-bit data right or left 1 bit at a time. The shift occurs on the rising edge (OFF-to-ON transition) of the Shift signal. The direction of shift is specified by the device s. A 0 in the least significant bit of device s specifies left shift and a 1 specifies right shift.

In a right (left) shift, the left-most (right-most) bit is loaded with the value of "Shift Input." The bit that is shifted out of the register (the left-most (right-most) bit in a left (right) shift) is loaded into the carry flag. Since the instruction has no reset function, the register should be cleared to zero with the BSET instruction.

● Example of a shift register operation

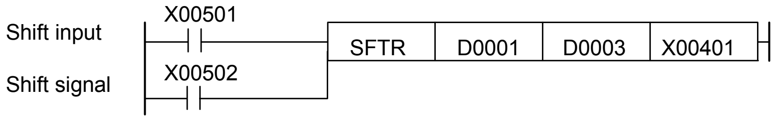


F030610.VSD

Figure 3.6.7 Example of a Shift Register Operation

■ Programming Example

The sample code shown below shifts 3 words (48 bits) of data in location from D001 to D003 1 bit to the right if X00401 is ON and to the left if X00401 is off when X00502 turns on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	LD	X00502				
0003	↑SFTR	D0001	D0003	X00401		

F030611.VSD

Figure 3.6.8 Example of a Shift Register Program

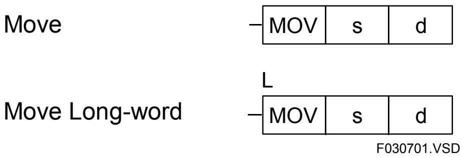
3.7 Data Transfer Instructions

3.7.1 Move (MOV), Move Long-word (MOV L)

Table 3.7.1 Move, Move Long-word

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required? Yes	Input Condition Required? No	Execution Condition	Step Count	Processing Unit	Carry
Application Instruction	40	Move	MOV	MOV	✓	—	⎍	3	16 bit	—
	40P		↑MOV	↑ MOV			↑	4		
	40L	Move Long-word	MOV L	L MOV	✓	—	⎍	3	32 bit	—
	40LP		↑MOV L	↑L MOV			↑	4		

■ Parameter



s : Device number of the first device storing the data to be moved (source)
d : Device number of the first device for storing the moved data (destination)

■ Available Devices

Table 3.7.2 Devices Available for the Move and Move Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value (may not be used as a long-word parameter)
*3: Counter current value (may not be used as a long-word parameter)

■ Function

The Move and Move Long-word instructions move 16- and 32-bit data, respectively.

Use the Move instruction to move 16-bit data and the Move Long-word instruction to move 32-bit data. The instructions support data move for the following combinations of source and destination devices:

- Binary code devices and binary code devices
- BCD code devices and BCD code devices
- Binary code devices and BCD code devices

Note that the value range of BCD code is narrower than that of binary code.

No data move operation is executed if the source devices (s) are defined in BCD code and its value exceeds the valid value range of the BCD code or if the destination devices (d) are defined in BCD code and the data to be moved is a negative number.

● Example of a move

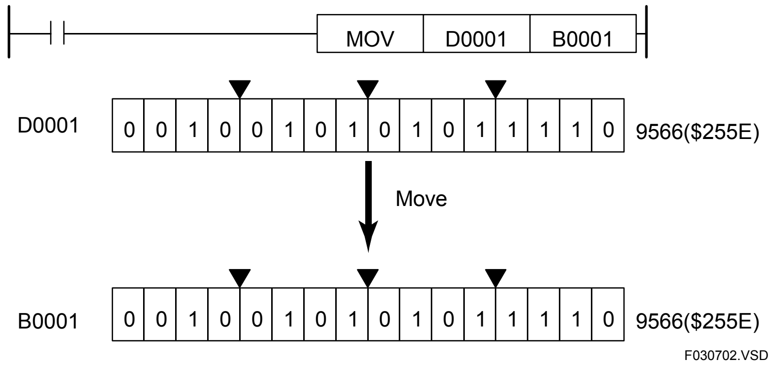


Figure 3.7.1 Example of a Move Operation (1)

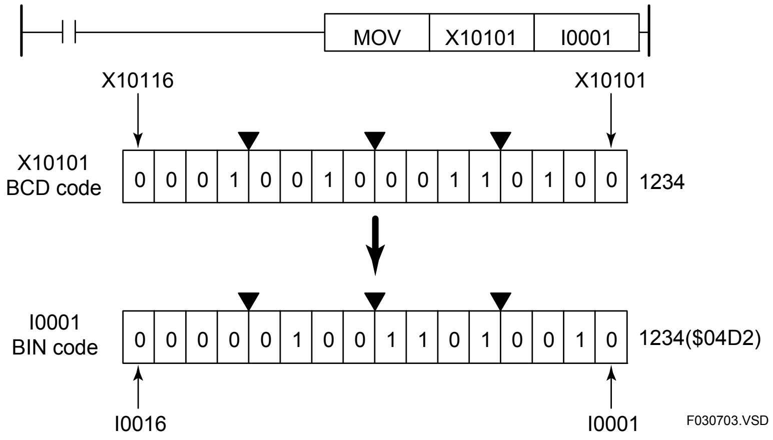
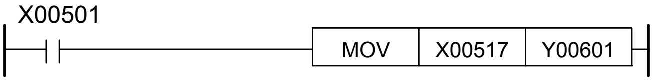


Figure 3.7.2 Example of a Move Operation (2)

■ Programming Example

The sample code shown below moves the 16 bits in location starting at X00517 to the 16-bit location starting at Y00601 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	MOV	X00517	Y00601			

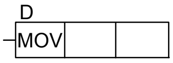
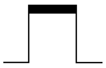
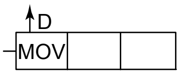
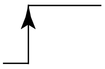
F030704.VSD

Figure 3.7.3 Example of a Move Program

3.7.2 Move Double Long-word (MOV D)

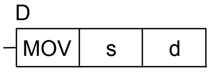
F3SP71
F3SP76

Table 3.7.3 Move Double Long-word

Classifi-cation	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	40D	Move Double Long-word	MOV D		✓	—		4	64 bit	—
	40DP		↑MOV D					5		

■ Parameter

Move Double Long-word



F372001.VSD

s : Device number of the first device storing the data to be moved (source)
d : Device number of the first device for storing the moved data (destination)

■ Available Devices

Table 3.7.4 Devices Available for the Move Double Long-word Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s									✓	✓*1	✓*1	✓		✓		✓	Yes	Yes
d									✓	✓*1	✓*1	✓*1		✓*1			Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Move Double Long-word instruction moves double long-words (64 bit data).
This instruction supports data move for binary code devices only.

● Example of a move

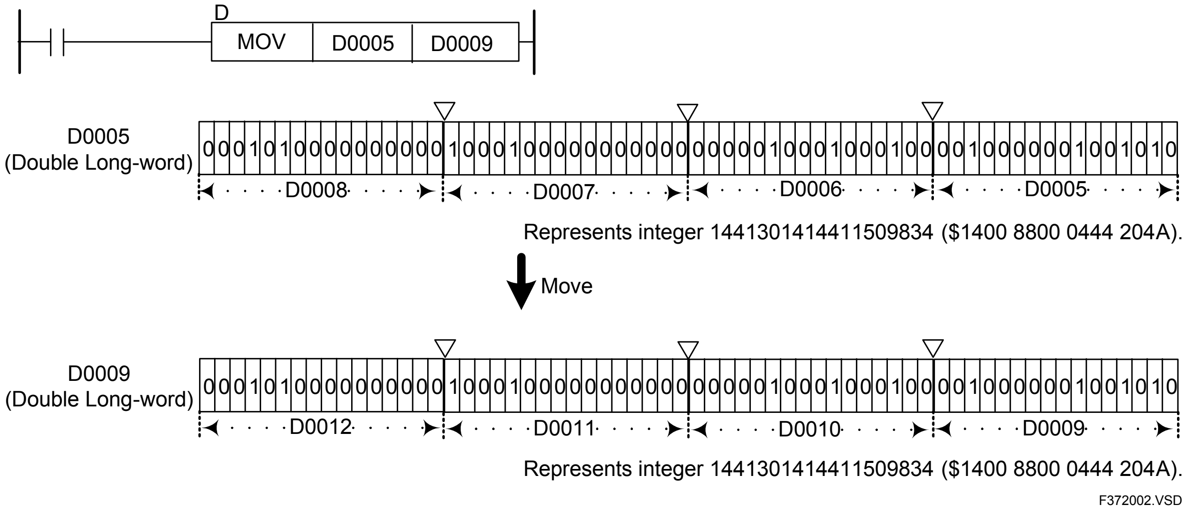
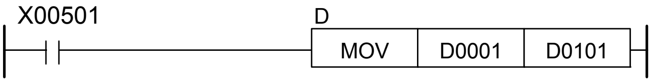


Figure 3.7.4 Example of a Move Double Long-word Operation

■ Programming Example

The sample code shown below moves 64 bit double long-word data in the location from D0001 to D0004 to the location from D0101 to D1014 if X00501 is on.



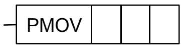
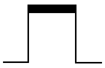
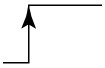
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	MOV D	D0001	D0101			

F372003.VSD

Figure 3.7.5 Example of a Move Double Long-word Program

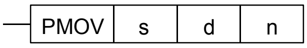
3.7.3 Partial Move (PMOV)

Table 3.7.5 Partial Move

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	41	Partial Move	PMOV	 PMOV	✓	—		4	16 bit	—
	41P		↑PMOV	PMOV				5		

■ Parameter

Partial Move



F030705.VSD

s : Device number of the first device storing the data to be moved (source)
d : Device number of the first device for storing the moved data (destination)
n : Number of bits to be transferred (1 to 16)
4, 8, 12, or 16 if s is coded in BCD.

■ Available Devices

Table 3.7.6 Devices Available for the Partial Move Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
n	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value
*3: Counter current value

■ Function

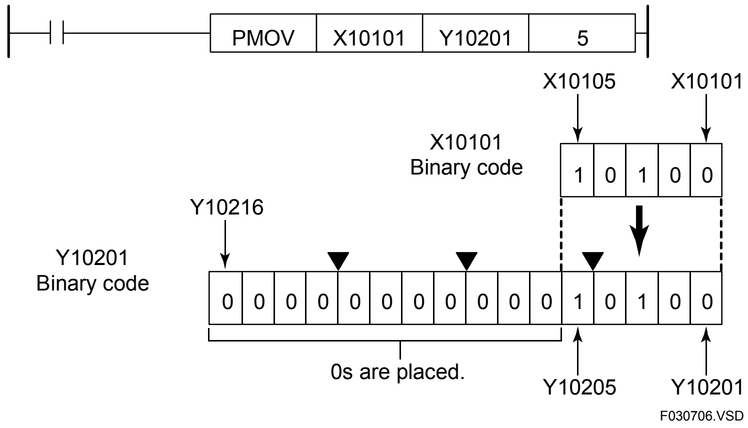
The Partial Move instruction moves 16 bits of data. It moves n bits of data starting at the device designated by s to the 16-bit location starting at the device designated by d.

The instruction supports data move for the following combinations of source and destination devices:

- Binary code devices and binary code devices
- BCD code devices and BCD code devices
- Binary code devices and BCD code devices

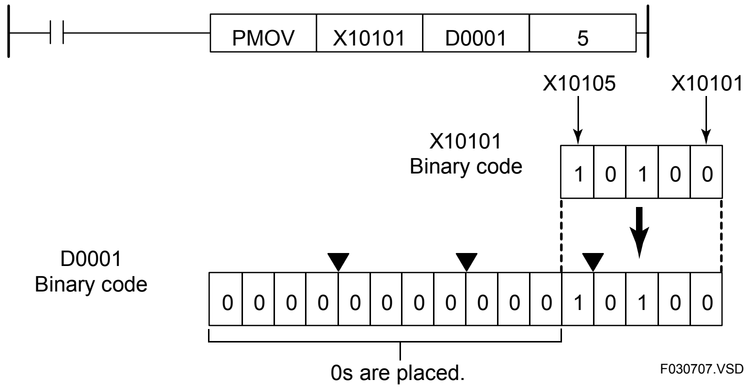
The number (n) of bits that can be moved is either 4, 8, 12, or 16 if the source consists of BCD coded devices.

No data move operation is executed if the source devices (s) are defined in BCD code and its value exceeds the valid value range of the BCD code or if the destination devices (d) are defined in BCD code and the data to be moved is a negative number.



F030706.VSD

Figure 3.7.6 Example of a Relay-to-relay Move

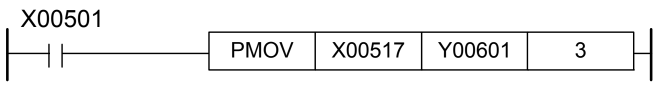


F030707.VSD

Figure 3.7.7 Example of a Relay-to-register Move

■ Programming Example

The sample code shown below moves 3 bits in location starting at X00517 to the 3-bit location starting at Y00601 if X00501 is on (Y00604 to Y00616 are padded with zeros).



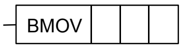
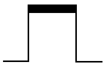
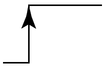
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	PMOV	X00517	Y00601	3		

F030708.VSD

Figure 3.7.8 Example of a Move Program

3.7.4 Block Move (BMOV)

Table 3.7.7 Block Move

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	42	Block Move	BMOV	 BMOV	✓	—		4	n words	—
	42P		↑BMOV	BMOV				5		

■ Parameter

Block Move

BMOV	s	d	n

F030709.VSD

s : Device number of the first device storing the data to be moved (source)
d : Device number of the first device for storing the moved data (destination)
n : Number of words to be transferred (1 to 2048)

■ Available Devices

Table 3.7.8 Devices Available for the Block Move Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
n	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Current timer value (may not be used as s or d if n is 1)

*3: Current counter value (may not be used as s or d if n is 1)

■ Function

The Block Move instruction moves n words (points) of data starting at the device designated by s to the location starting at the device designated by d. The instruction is equivalent to the BSET instruction if s is a constant.

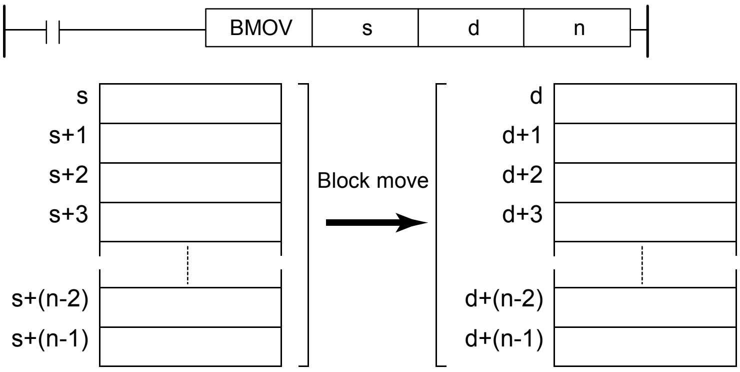


Figure 3.7.9 Block Move

- **Example**

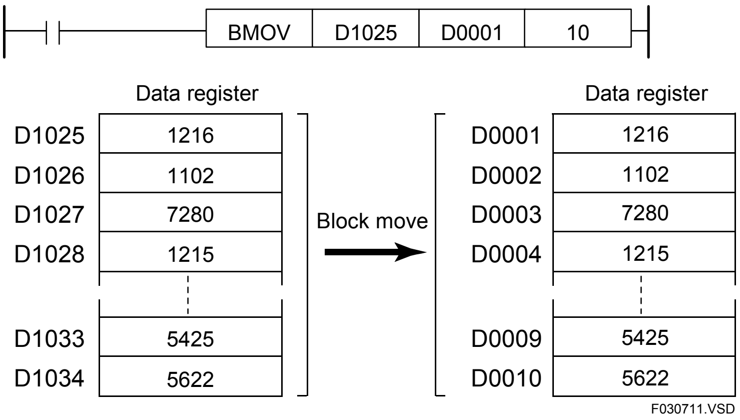


Figure 3.7.10 Example of a Block Move between Registers

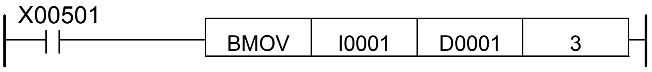
CAUTION

If the starting device number (transfer destination) of the destination is out of the device range, an error is generated. However, if the starting device number of the destination + the number of words to be transferred is out of the device range, no error is signaled. This may result in modification of unintended devices and result in unexpected operations.

Errors can be detected only when F3SP71 or F3SP76 CPU module is used. For details, see Section 1.10.3, "Device Boundary Check."

■ Programming Example

The sample code shown below moves 3 words (16 bits x 3 = 48 bits) in location starting at I0001 to the location starting at D0001 if X00501 is on.



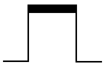
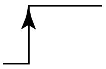
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	BMOV	I0001	D0001	3		

F030712.VSD

Figure 3.7.11 Example of a Block Move Program

3.7.5 Block Set (BSET)

Table 3.7.9 Block Set

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	43	Block Set	BSET	BSET	✓	—		4	n words	—
	43P		↑BSET	BSET				5		

■ Parameter

Block Set — BSET | s | d | n

F030713.VSD

s : Device number of the first device storing the data to be moved (source)
d : Device number of the first device for storing the move data (destination)
n : Number of words to be transferred (1 to 2048)

■ Available Devices

Table 3.7.10 Devices Available for the Block Set Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
n	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Current timer value (may not be used as s or d if n is 1)

*3: Current counter value (may not be used as s or d if n is 1)

CAUTION

If the starting device number (transfer destination) of the destination is out of the device range, an error is generated. However, if the starting device number of the destination + the number of words to be transferred is out of the device range, no error is signaled. This may result in modification of unintended devices and result in unexpected operations.

Errors can be detected only when F3SP71 or F3SP76 CPU module is used. For details, see Section 1.10.3, "Device Boundary Check."

■ Function

The Block Set instruction transfers 1 word (16 bits) of data starting at the device designated by s to n words of area starting at the device designated by d. Data registers and holding type internal relays can be initialized at a time by setting them with 0s using the Block Set instruction.

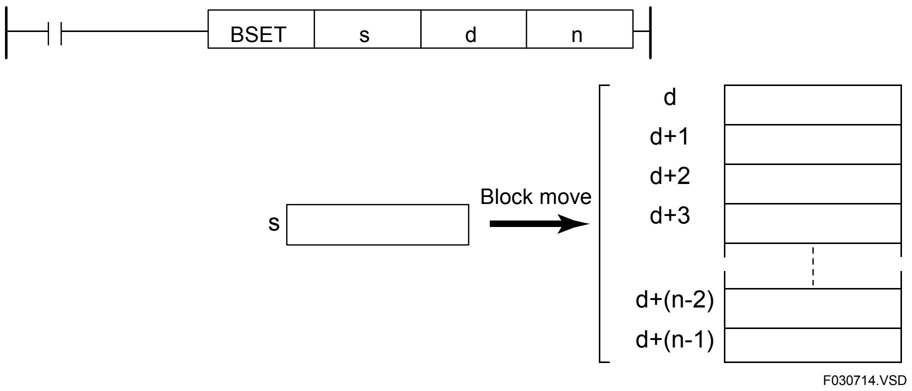


Figure 3.7.12 Example of a Block Set

● Example: Initialing data registers

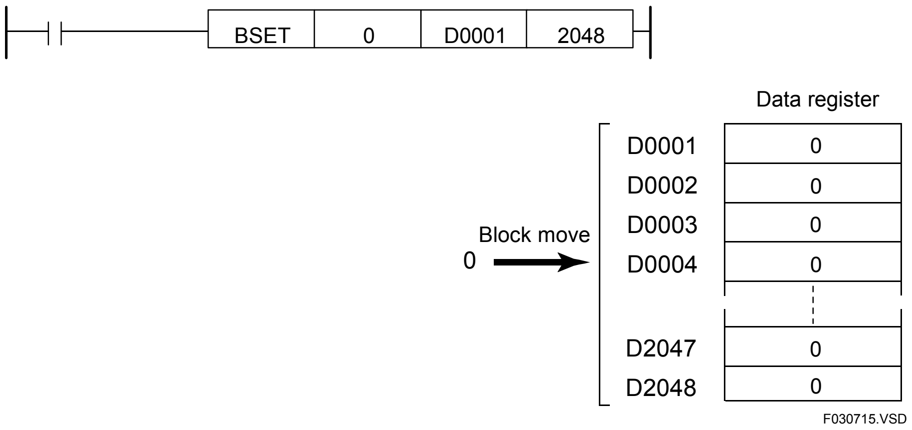
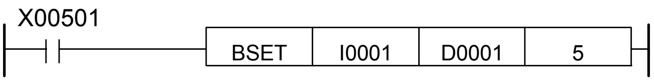


Figure 3.7.13 Example of Initializing Data Registers with Block Set

■ Programming Example

The sample code shown below transfers 1 word (16 bits) of data in location starting at I0001 to 5 words of area starting at D0001 if X00501 is on.



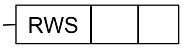
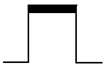
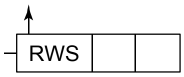
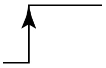
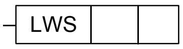
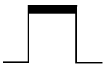
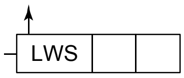
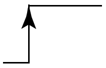
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	BSET	I0001	D0001	5		

F030716.VSD

Figure 3.7.14 Example of a Block Set Program

3.7.6 Word Shift (RWS, LWS)

Table 3.7.11 Word Shift

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	44	Right Word Shift	RWS		✓	—		3	n words	—
	44P		↑RWS					4		
	45	Left Word Shift	LWS		✓	—		3	n words	—
	45P		↑LWS					4		

■ Parameter

Right Word Shift	RWS	d	n
Left Word Shift	LWS	d	n

F030717.VSD

d : Device number of the first device storing the words to shift
n : Number of words to shift

■ Available Devices

Table 3.7.12 Devices Available for the Word Shift Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
n	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value

*3: Counter current value

CAUTION

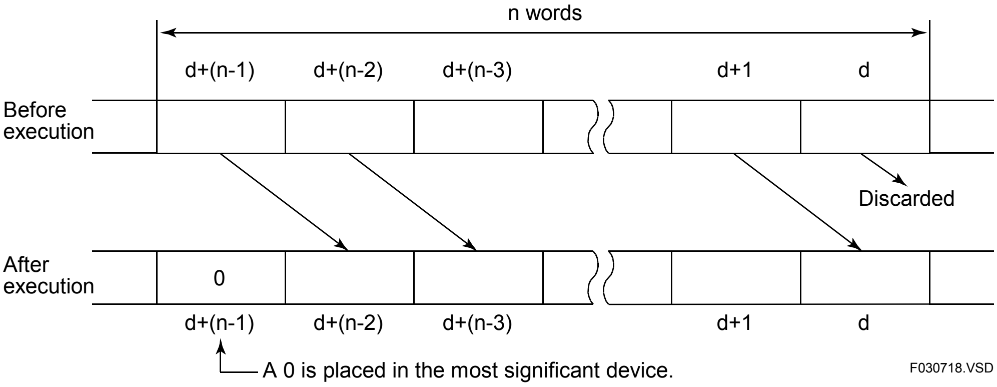
If the starting device number of the data to be shifted is out of the device range, an error is generated. However, if the starting device number of the data to be shifted + the number of words to be shifted is out of the device range, no error is signaled. This may result in modification of unintended devices and result in unexpected operations.

Errors can be detected only when F3SP71 or F3SP76 CPU module is used. For details, see Section 1.10.3, "Device Boundary Check."

■ Function

The Right and Left Word Shift instructions shift n words data starting at the device designated by d to the right and left, respectively, 1 word at a time.

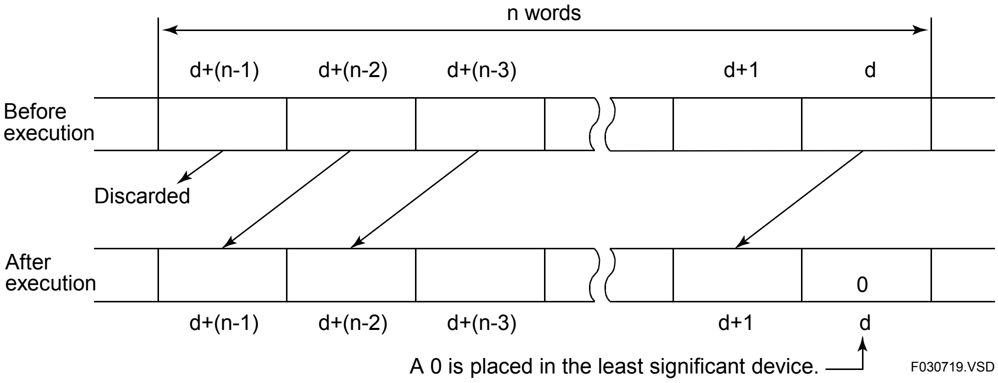
(1) Right Word Shift



A 0 is placed in the most significant device.

Figure 3.7.15 Example of a Right Word Shift Operation

(2) Left Word Shift

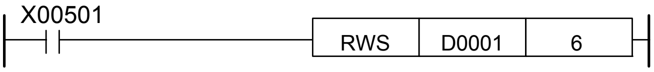


A 0 is placed in the least significant device.

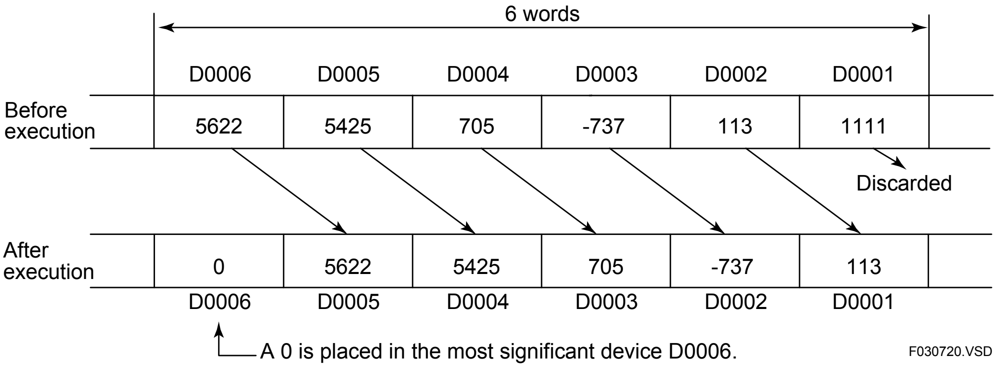
Figure 3.7.16 Example of a Left Word Shift Operation

■ Programming Example

The sample code shown below shifts 6 words of data starting at D001 to the right, 1 word at a time, if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	RWS	D0001	6			

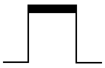
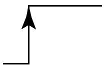
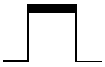
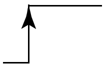


A 0 is placed in the most significant device D0006.

Figure 3.7.17 Example of a Right Shift Program

3.7.7 Indexed Move (IXMOV), Indexed Move Long-word (IXMOV L)

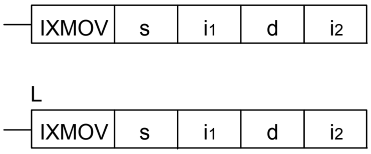
Table 3.7.13 Indexed Move, Indexed Move Long-word

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	46	Index Move	IXMOV	IXMOV	✓	—		5	16 bits	—
	46P		↑IXMOV	↑ IXMOV				6		
	46L	Index Move Long-word	IXMOV L	L IXMOV	✓	—		5	32 bits	—
	46LP		↑IXMOV L	↑L IXMOV				6		

■ Parameter

Indexed Move

Indexed Move Long-word



F030721.VSD

s : Device number serving as the starting address of the source
i_1 : Number* of the device storing the index to the beginning of the source*1
d : Device number serving as the starting address of the destination
i_2 : Number* of the device storing the index to the beginning of the source*1
*1: i_1 and i_2 are handled as 16 bits (1 word) even in a 32-bit (long-word) instruction.

■ Available Devices

Table 3.7.14 Devices Available for the Indexed Move and Indexed Move Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	No	Yes
i_1	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	No	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		No	Yes
i_2	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	No	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value (may not be used as a long-word parameter)
*3: Counter current value (may not be used as a long-word parameter)

■ Function

The Indexed Move and Indexed Move Long-word instructions perform an indexed move operation on 16- and 32-bit data, respectively.

Use the Indexed Move instruction to perform an indexed move operation on 16-bit data and the Indexed Move Long-word instruction to perform an indexed move on operation 32-bit data.

These instructions move 16- and 32-bit data, respectively, starting at the device whose location is specified by the device designated by s (source starting address) plus the value of index i1 to the device whose location is specified by the device designated by d (destination starting address) plus the value of index i2. If s is a literal, the literal proper is transferred regardless of the value of index i1.

● Example

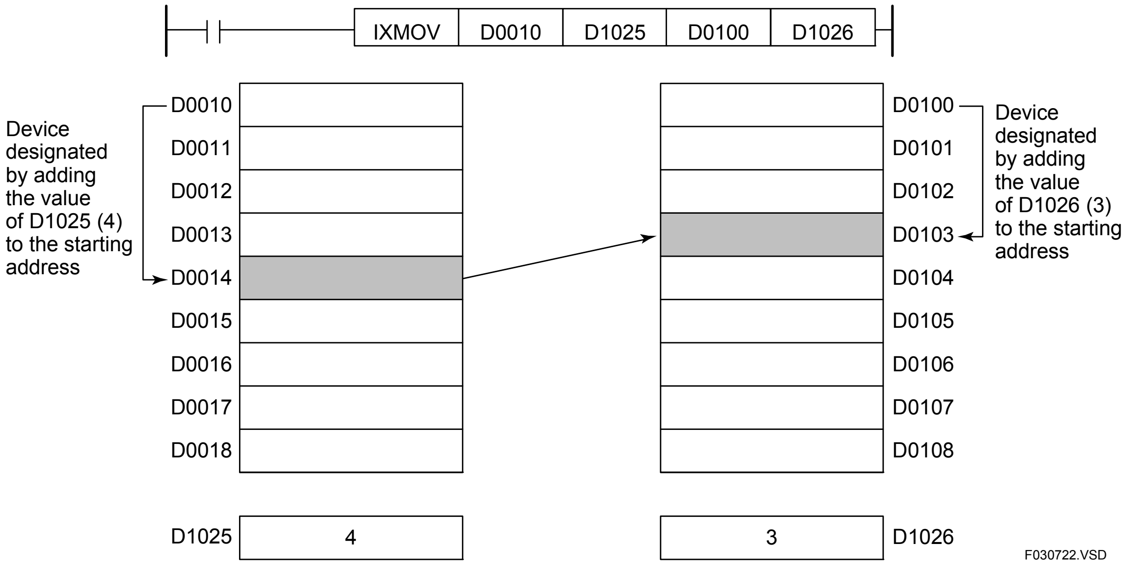

IXMOV
D0010
D1025
D0100
D1026
Device designated by adding the value of D1025 (4) to the starting address
D0010
D0011
D0012
D0013
D0014
D0015
D0016
D0017
D0018
D1025
4
3
D1026
D0100
D0101
D0102
D0103
D0104
D0105
D0106
D0107
D0108
Device designated by adding the value of D1026 (3) to the starting address
F030722.VSD

Figure 3.7.18 Example of a Register-to-register Indexed Move Operation

If the source devices are internal relays (I), shared relays (E), or special relays (M), or if the destination devices are internal relays (I) or shared relays (E), the index values (values of i1 and i2) are added on a relay basis.

● Example

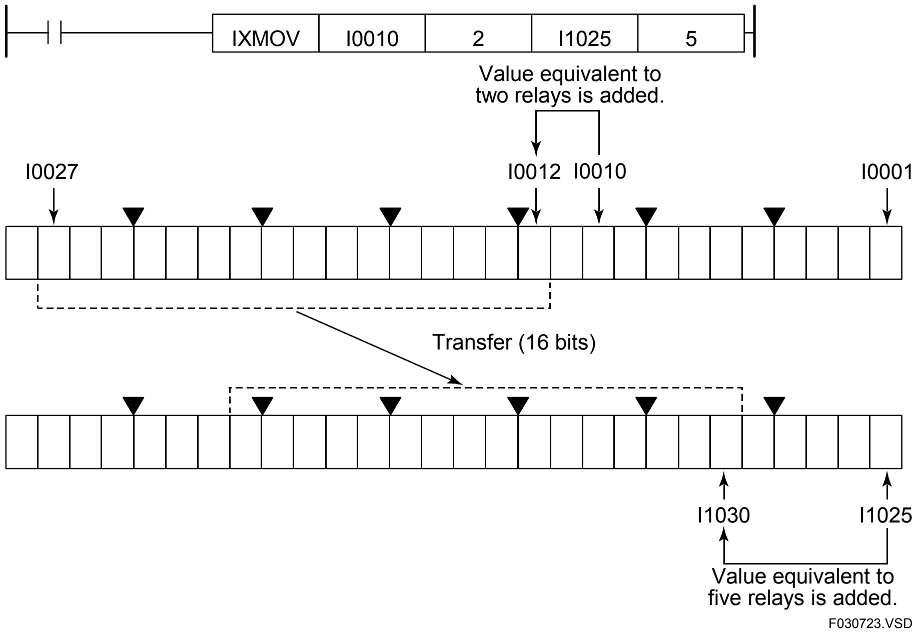
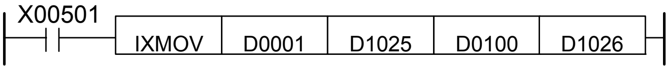


Figure 3.7.19 Example of an Indexed Move Operation

■ Programming Example

The sample code shown below moves the contents of the data register whose address is determined by the contents of D0001 (s) plus D1025 (i1) to the data register whose address is determined by the contents of D0100 (d) plus D1026 (i2) if X00501 is on.



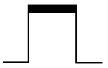
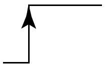
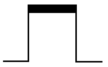
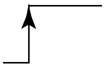
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	IXMOV	D0001	D1025	D0100	D1026	

F030724.VSD

Figure 3.7.20 Example of an Indexed Move Program

3.7.8 Exchange (XCHG), Exchange Long-word (XCHG L)

Table 3.7.15 Exchange, Exchange Long-word

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	47	Exchange Data	XCHG	XCHG	✓	—		3	16 bits	—
	47P		↑XCHG	↑ XCHG				4		
	47L	Exchange Data Long-word	XCHG L	L XCHG	✓	—		3	32 bits	—
	47LP		↑XCHG L	↑L XCHG				4		

■ Parameter

Exchange Data — XCHG | d1 | d2

Exchange Data Long-word — L XCHG | d1 | d2

F030725.VSD

d1, d2 : Device number of the first device storing the data to be exchanged

■ Available Devices

Table 3.7.16 Devices Available for the Exchange and Exchange Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d1		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
d2		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value (may not be used as a long-word parameter)

*3: Counter current value (may not be used as a long-word parameter)

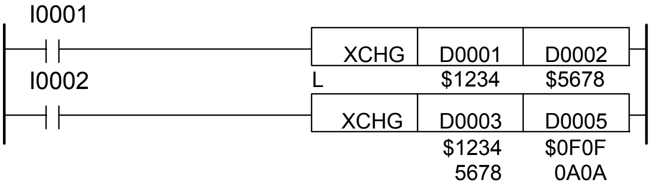
■ Function

The Exchange and Exchange Long-word instructions exchange the contents of the device designated by d1 with the contents of the device designated by d2.

Use the Exchange instruction to exchange 16-bit data and the Exchange Long-word instruction to exchange 32-bit data.

■ Programming Example

The sample code shown below exchange the contents of D0001 ($1234) with the contents of D0002 ($5678) if I0001 is on, and the contents of D0003 ($12345678) with the contents of D0005 ($0F0F0A0A) if I00012 is on.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	XCHG	D0001	D0002			
0003	LD	I0002				
0004	XCHG L	D0003	D0005			

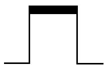
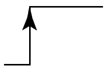
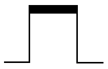
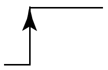
	Exchange Word Data	Exchange Long Word Data
Before execution	D0001 $1234 ↔ D0002 $5678	D0004 $1234, D0003 $5678 ↔ D0006 $0F0F, D0005 $0A0A
After execution	D0001 $5678 ↔ D0002 $1234	D0004 $0F0F, D0003 $0A0A ↔ D0006 $1234, D0005 $5678

F030726.VSD

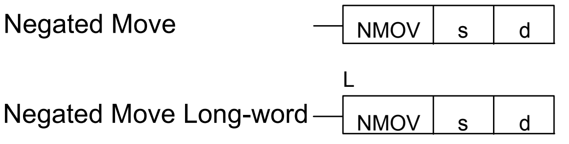
Figure 3.7.21 Example of a Data Exchange Program and a Long-word Data Exchange Program

3.7.9 Negated Move (NMOV), Negated Move Long-word (NMOV L)

Table 3.7.17 Negated Move, Negated Move Long-word

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	140	Negated Move	NMOV	NMOV	✓	—		3	16 bits	—
	140P		↑NMOV	↑ NMOV				4		
	140L	Negated Move Long-word	NMOV L	L NMOV	✓	—		3	32 bits	—
	140LP		↑NMOV L	↑L NMOV				4		

■ Parameter



F030727.VSD

s : Device number of the first source device
d : Device number of the first destination device

■ Available Devices

Table 3.7.18 Devices Available for the Negated Move and Negated Move Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value (may not be used as a long-word parameter)

*3: Counter current value (may not be used as a long-word parameter)

■ Function

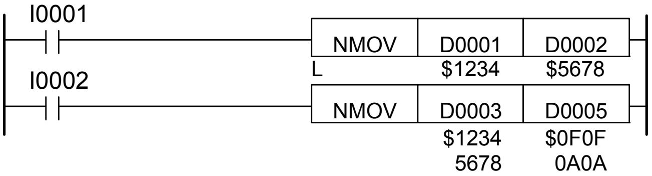
The Negated Move and Negated Move Long-word instructions negate 16- and 32-bit data, respectively, and move the result to the specified devices.

Use the Negated Move instruction to move 16-bit data and the Negated Move Long-word instruction to move 32-bit data.

The operand data to be negated may be either of binary or BCD type. In either case, the operand is handled as a collection of bit data. Consequently, no BCD range check is made before and after the negation.

■ Programming Example

The sample code shown below takes NOT of the contents of D0001 ($1234) and moves the result to D0002 if I0001 is on. If I0002 is on, the program takes NOT of the contents of D0003 ($12345678) and moves the result to D0005.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	NMOV	D0001	D0002			
0003	LD	I0002				
0004	NMOV L	D0003	D0005			

	Negated Move		Negated Move Long-word			
Before execution	D0001 $1234	D0002 $5678	D0004 $1234	D0003 $5678	D0006 $0F0F	D0005 $0A0A
After execution	D0001 $1234	*1 D0002 $EDCB	D0004 $1234	D0003 $5678	*1 D0006 $EDCB	D0005 $A987

F030728.VSD

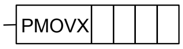
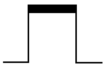
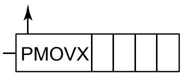
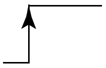
*1: Negation of $1234/$12345678 results in $EDCB/$EDCBA987

1	2	3	4	5	6	7	8	Hexadecimal
0001	0010	0011	0100	0101	0110	0111	1000	Binary
1110	1101	1100	1011	1010	1001	1000	0111	Binary
E	D	C	B	A	9	8	7	Hexadecimal

Figure 3.7.22 Example of a Negated Move Program

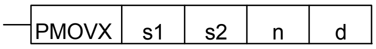
3.7.10 Extended Partial Move (PMOVX)

Table 3.7.19 Partial Move Extended

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required? Yes	No	Execution Condition	Step Count	Pro-cessing Unit	Carry
Appli-cation Instruc-tion	141	Extended Partial Move	PMOVX		✓	—		5	16 bits or less	—
	141P		↑PMOVX					6		

■ Parameter

Partial Move Extended



F030729.VSD

s1 : Device number of the first source device
s2 : Starting bit position (0 to 15) of bits to be moved
n : Number of bits to be moved (1 to 16)
d : Device number of the first destination device

■ Available Devices

Table 3.7.20 Devices Available for the Digit Move Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s1	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
n	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

*2: Timer current value

*3: Counter current value

■ Function

The Extended Partial Move instruction moves n bits of 16-bit data designated by s1 starting at bit s2 to the device designated by d. The most significant bit of s2 is 15 ($000F) and the least significant bit is 0 ($0000).

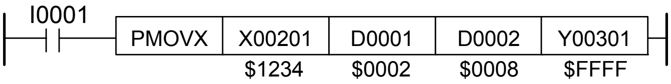
The bit value that is established before the execution of the instruction is retained for the bits except the lowest-order n bits of the destination. While 0s are set in the remaining bits with PMOV, the old bit value is retained with PMOVX.

The operand data to be moved may be either of binary or BCD type. In either case, the operand is handled as a collection of bit data. Consequently, no BCD range check is made before and after the negation.

CAUTION

If s2 does not fall within the value range of 0 to 15 or s2+n within the value range of 1 to 16, an error is signaled and the instruction is not executed.

■ Programming Example



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	PMOVX	X00201	D0001	D0002	Y00301	

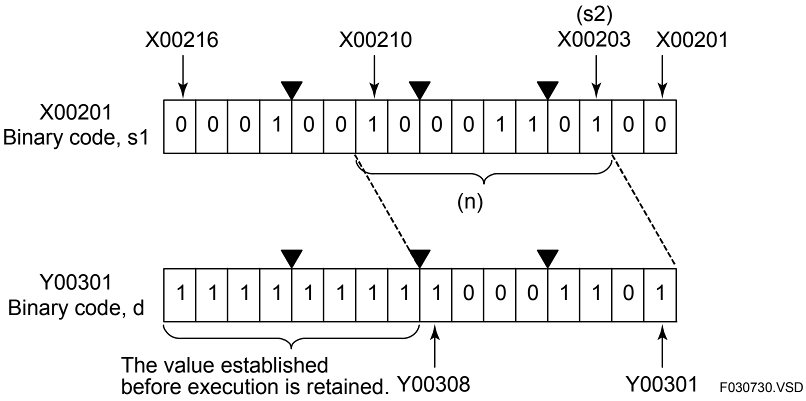


Figure 3.7.23 Example of an Extended Partial Move Program

● High-speed Extended Partial Move

When the source consists of bit devices (input/output relays or internal relays), the result of move remains unchanged even if the source device number is set to the starting device number of the source plus the starting bit position and the starting bit position is set to 0 (see Figure (2) below). The move operation, however, will be faster if the starting device number of the source is set to a word boundary (1, 17, 33, ..., (16n + 1)) (see Figure below).

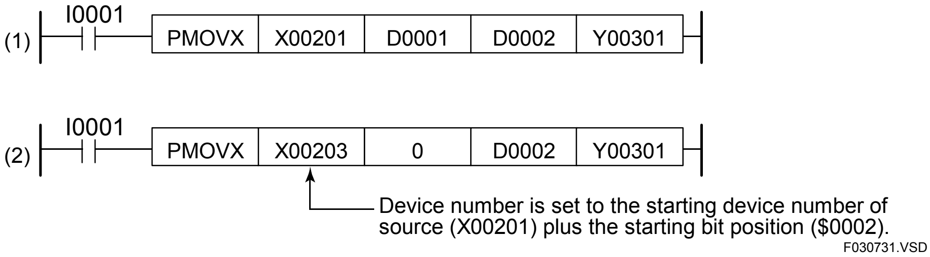
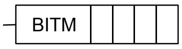
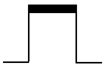
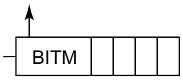
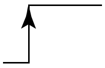

(1)
I0001
PMOVX
X00201
D0001
D0002
Y00301
(2)
I0001
PMOVX
X00203
0
D0002
Y00301
Device number is set to the starting device number of source (X00201) plus the starting bit position ($0002).
F030731.VSD

Figure 3.7.24 Example of an Extended Partial Move Program

Although these steps (1) and (2) produce the same result, step (1) executes faster because the starting device (X00201) of the source is set to a word boundary.

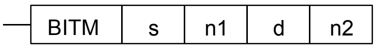
3.7.11 Bit Move (BITM)

Table 3.7.21 Bit Move

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	142	Bit Move	BITM		✓	—		5	16 bits	—
	142P		↑BITM					6		

■ Parameter

Bit Move



F030732.VSD

s : Device number of the first source device
n1 : Starting bit position (0 to 15) in the source
d : Device number of the first destination device
n2 : Starting bit position (0 to 15) in the destination

■ Available Devices

Table 3.7.22 Devices Available for the Bit Move Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
n1	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
n2	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value
*3: Counter current value

■ Function

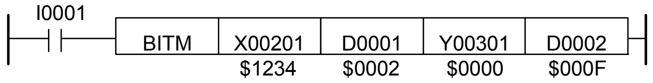
The Bit Move instruction moves the bit designated by n1 of the 16-bit data designated by device s to the bit position designated by n2 of the device designated by d. The most significant bit of n1 and n2 is 15 ($000F) and the least significant bit is 0 ($0000). Except for bit n2, all other bits of the destination remain unchanged.

CAUTION

An error is signaled and the execution of the Bit Move instruction is suppressed if n1 or n2 does not fall within the value range of 0 to 15.

■ Programming Example

The sample code shown below moves the bit D0001 ($0002) of device X00201 ($1234) to bit D0002 ($000F) of device Y00301 ($0000) if I0001 is on.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	BITM	X00201	D0001	Y00301	D0002	

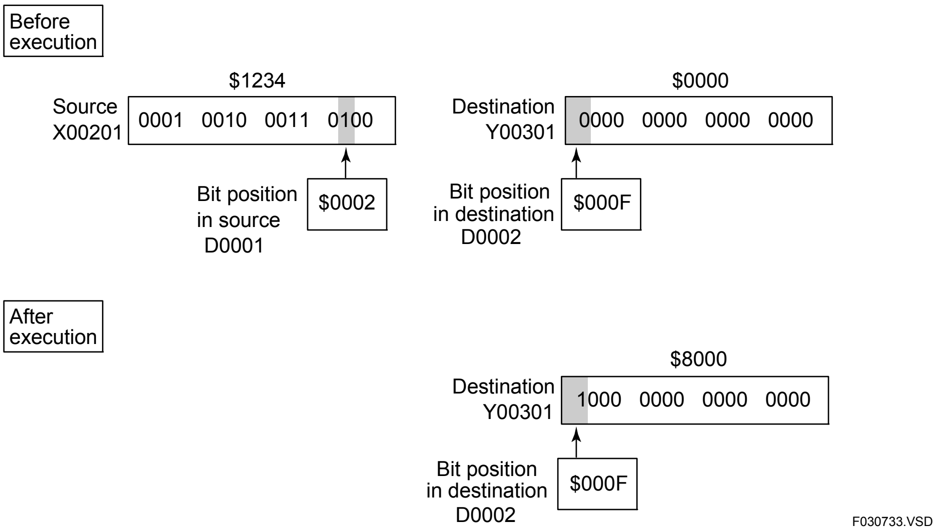
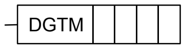
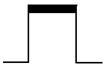
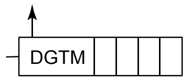
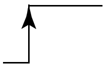


Figure 3.7.25 Example of a Bit Move Program

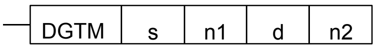
3.7.12 Digit Move (DGTM)

Table 3.7.23 Digit Move

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	143	Digit Move	DGTM	 DGTM	✓	—		5	16 bits	—
	143P		↑DGTM	 DGTM				6		

■ Parameter

Digit Move



F030734.VSD

s : Device number of the first source device
n1 : Starting digit position (0 to 3) in the source
d : Device number of the first destination device
n2 : Starting digit position (0 to 3) in the destination

■ Available Devices

Table 3.7.24 Devices Available for the Digit Move Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
n1	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
n2	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value
*3: Counter current value

■ Function

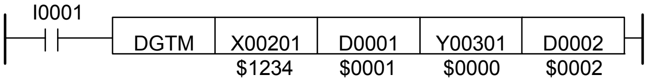
The Digit Move instruction moves the digit designated by n1 of the 16-digit data designated by device s to the digit position designated by n2 of the device designated by d. The least significant digit (bits 0 to 3) of n1 and n2 is 0 ($0000) and the most significant digit (bits 12 to 15) is 3 ($0003). Except for bit n2, all other bits of the destination remain unchanged.

CAUTION

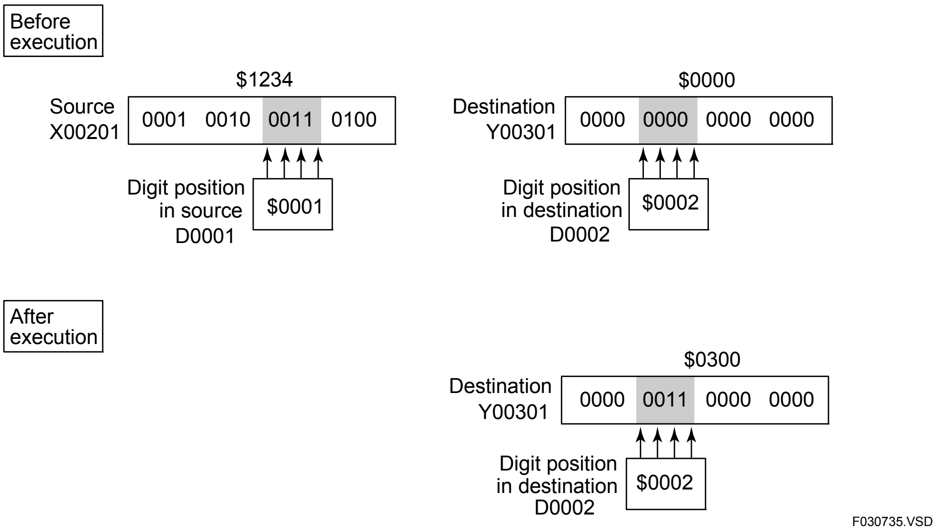
An error is signaled and the execution of the Digit Move instruction is suppressed if n1 or n2 does not fall within the value range of 0 to 3.

■ Programming Example

The sample code shown below moves the digit D001 ($0001) of device X00201 ($1234) to digit D0002 ($0002) of device Y00301 ($0000) if I0001 is on.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	DGTM	X00201	D0001	Y00301	D0002	



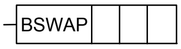
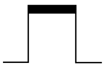
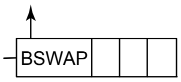
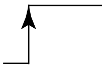
F030735.VSD

Figure 3.7.26 Example of a Digit Move Program

3.7.13 Block Swap Move (BSWAP)

F3SP71-4S
F3SP76-7S

Table 3.7.25 Block Swap Move

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	48	Block Swap Move	BSWAP		✓	—		4	16 bits	—
	48P		↑BSWAP					5		

■ Parameter

Block Swap Move

T030735.VSD

s : Device number of the first device storing the data to be moved (source)
d : Device number of the first device for storing the moved data (destination)
n : Number of words to be transferred (1 to 2048)

■ Available Devices

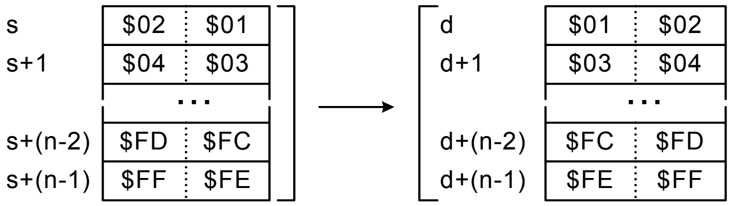
Table 3.7.26 Devices Available for the Block Swap Move Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s									✓	✓	✓*1	✓		✓			Yes	Yes
d									✓	✓	✓*1	✓*1		✓*1			Yes	Yes
n									✓	✓	✓*1	✓		✓		✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Block Swap Move instruction swaps highest-order 8 bits and lowest-order 8 bits of n words of data starting at the device designated by s, for each word (16 bits) at a time, and writes the result to n words of area starting at the device designated by d.



s	$02	$01
s+1	$04	$03
	...	
s+(n-2)	$FD	$FC
s+(n-1)	$FF	$FE

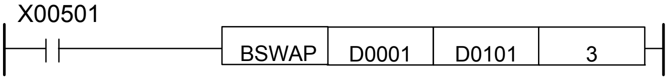
d	$01	$02
d+1	$03	$04
	...	
d+(n-2)	$FC	$FD
d+(n-1)	$FE	$FF

F030736.VSD

Figure 3.7.27 Block Swap Move

■ Program Example

The sample code shown below moves 3 words of data (16 bits x 3 = 48 bits) in the location starting at D0001 to the location starting at D0101 using the Block Swap Move instruction if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	BSWAP	D0001	D0101	3		

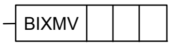
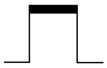
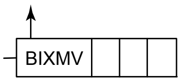
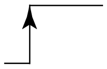
F030737.VSD

Figure 3.7.28 Example of a Block Swap Move Program

3.7.14 Byte Index Move (BIXMV)

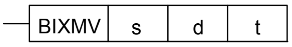
F3SP71-4S
F3SP76-7S

Table 3.7.27 Byte Index Move

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	49	Byte Index Move	BIXMV		✓	–		4	8 bits	–
	49P		↑BIXMV					5		

■ Parameter

Byte Index Move



F030738.VSD

Table 3.7.28 Parameter

Parameter		Description
s		Device number of the first device storing the data to be moved (source)
d		Device number of the first device for storing the moved data (destination)
t	t+0	No. of bytes of data to be moved (1 to 4096)
	t+1	Offset from the beginning of the source device (No. of bytes) (0 to 32767)
	t+2	Offset from the beginning of the device for storing the moved data (No. of bytes) (0 to 32767)

■ Available Devices

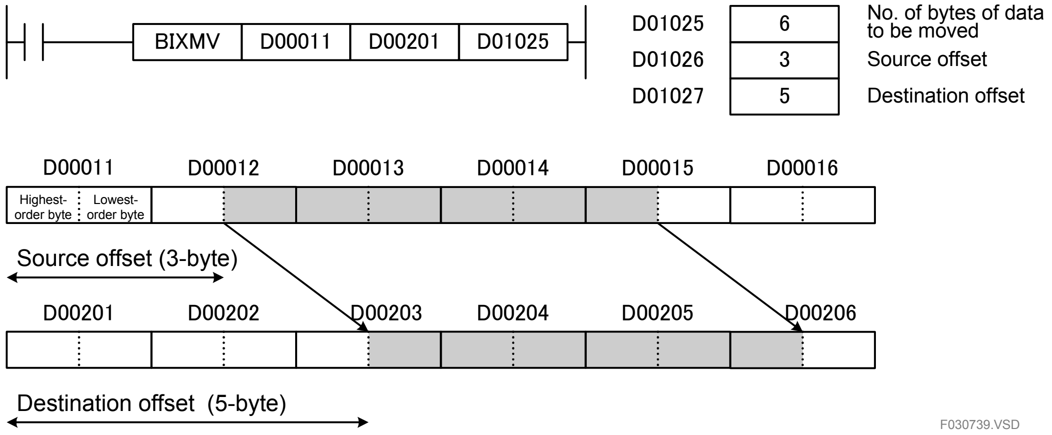
Table 3.7.29 Devices Available for the Byte Index Move Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s									✓	✓	✓*1	✓		✓			Yes	Yes
d									✓	✓	✓*1	✓*1		✓*1			Yes	Yes
t									✓	✓	✓*1	✓		✓			Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Byte Index Move instruction extracts number of bytes of data to be moved designated by (t+0) from the offset position (number of bytes) designated by (t+1) starting at the device designated by s and writes the extracted data to the offset position (number of bytes) designated by (t+2) bytes starting at the device designated by d.



F030739.VSD

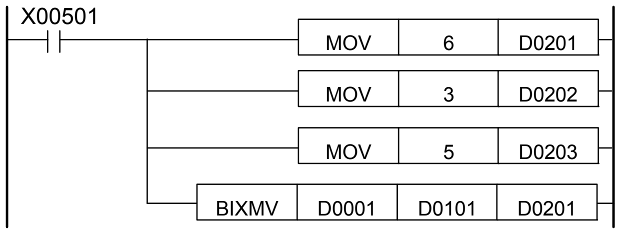
Figure 3.7.29 Byte Index Move

CAUTION

Note that it takes longer to move data if the source offset is an odd number of bytes and the destination offset is an even number of bytes, or if the source offset is an even number of bytes and the destination offset is an odd number of bytes.

■ Programming Example

The sample code shown below moves 6 bytes of data starting at the offset position at 3 bytes from D0001 to the offset position at 5 bytes from D0101 using the Byte Index Move instruction if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	PUSH					
0003	MOV	6	D0201			
0004	STCRD					
0005	MOV	3	D0202			
0006	STCRD					
0007	MOV	5	D0203			
0008	POP					
0009	BIXMV	D0001	D0101	D0201		

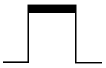
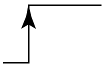
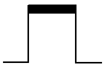
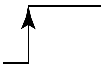
F030740.VSD

Figure 3.7.30 Example of a Byte Index Move Program

3.8 Data Processing Instructions

3.8.1 FIFO Instructions (FIFRD, FIFWR)

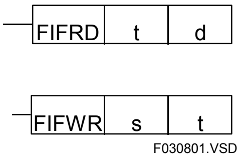
Table 3.8.1 FIFO Instructions

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	50	FIFO Read	FIFRD	FIFRD	✓	—		3	16 bit	—
	50P		↑FIFRD	↑ FIFRD				4		
	51	FIFO Write	FIFWR	FIFWR	✓	—		3	16 bit	—
	51P		↑FIFWR	↑ FIFWR				4		

■ Parameter

FIFO Read

FIFO Write



t : Device number identifying the beginning of the FIFO table
d : Device number of the first device storing the data read from the FIFO table
s : Device number of the first device storing the data to be loaded into the FIFO table

■ Available Devices

Table 3.8.2 Devices Available for the FIFO Instructions

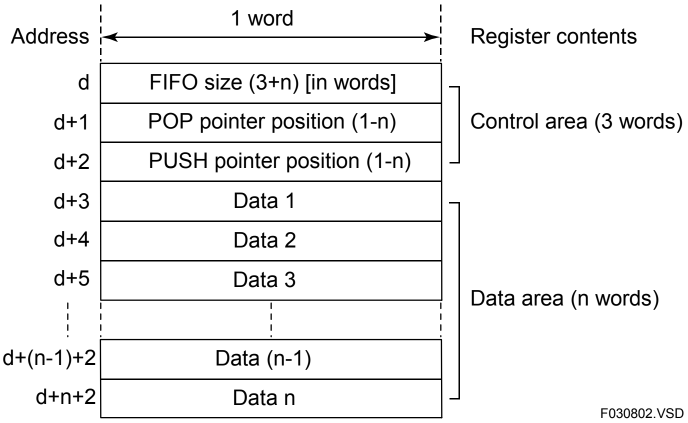
Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
t		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
s	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Current timer value
*3: Current counter value

■ Function

(1) FIFO Read

An FIFO Read instruction reads data from an FIFO (First In/First Out) table. It reads 1 word of data from the FIFO table whose beginning is designated by the parameter t at the entry designated by the POP pointer and places the word on the device designated by d. The instruction then advances the POP pointer by 1 address.



Address	1 word	Register contents
d	FIFO size (3+n) [in words]	Control area (3 words)
d+1	POP pointer position (1-n)	
d+2	PUSH pointer position (1-n)	
d+3	Data 1	Data area (n words)
d+4	Data 2	
d+5	Data 3	
⋮	⋮	
d+(n-1)+2	Data (n-1)	
d+n+2	Data n	

F030802.VSD

Note: An FIFO table requires a space of FIFO table size (n) + 3 words.

Figure 3.8.1 FIFO Table Structure

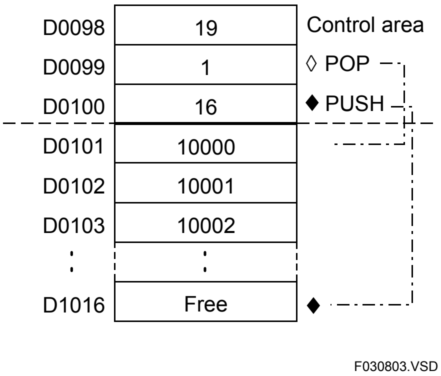
(2) FIFO Write

An FIFO Write instruction writes 1 word of data designated by the device s into the FIFO table whose beginning is designated by the parameter t at the entry designated by the PUSH pointer. The instruction then advances the PUSH pointer by 1 address.

The maximum number of data words that can be written to an FIFO data area is (n–1) words. The data area cannot contain n words of data. Consequently, data must be read with FIFO Read instructions before the data area becomes full.

CAUTION

The M025 special relay is set to ON to signal a processing error if an attempt is made to write more than (n–1) words into an FIFO table.



D0098	19	Control area
D0099	1	◊ POP
D0100	16	♦ PUSH
D0101	10000	
D0102	10001	
D0103	10002	
⋮	⋮	
D1016	Free	♦

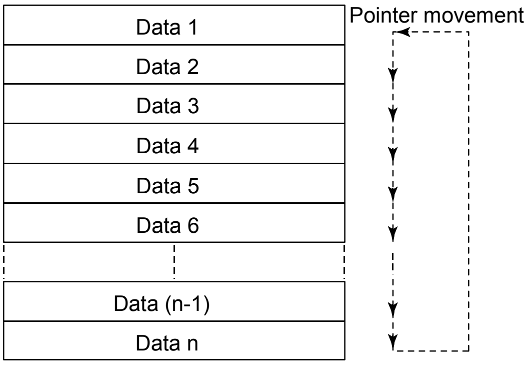
F030803.VSD

Note: The FIFO table in this state can no longer accommodate a word.

Figure 3.8.2 FIFO Table in the Full State

● Pointer Position

The data area for an FIFO table is made up of a rotary buffer in which every word read or write advances its pointer by one. When the pointer reaches the last entry (entry n) of the data area, it wraps around to the beginning of the data area (entry 1) on the next read or write.



F030804.VSD

Figure 3.8.3 FIFO Table Pointer

Before using an FIFO table, the programmer must initialize its FIFO size and POP pointers by writing initial values directly into their devices with Move instructions.

CAUTION

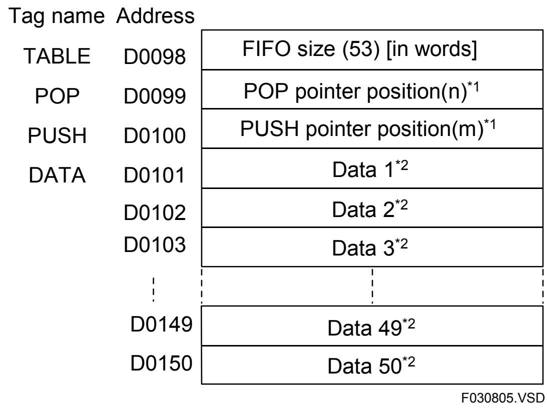
An instruction error is signaled and the special relay M201 is set to ON if an FIFO table pointer contains an invalid value (Note) or if a pointer value that goes beyond the data area of the FIFO table is specified. In such a case, the error number identifying the FIFO error, the block number identifying the block in error, and the step number are loaded into special registers Z22, Z23, and Z24.

Note: A FIFO pointer having an invalid pointer value refers to one of the following conditions:
- FIFO read: POP pointer position = PUSH pointer position
- FIFO write: PUSH pointer position = POP pointer position – 1
- The POP or PUSH pointer points to a location beyond the FIFO table area.

Except during initialization, the programmer need not be aware of the FIFO size, POP pointer, and PUSH pointer.

■ Programming Example

The sample code shown below is an inventory control program for an automatic warehouse. The program pushes any received items into an FIFO buffer (FIFO table) and pops issuing items out of the buffer (FIFO table) on a first-in/first-out basis.

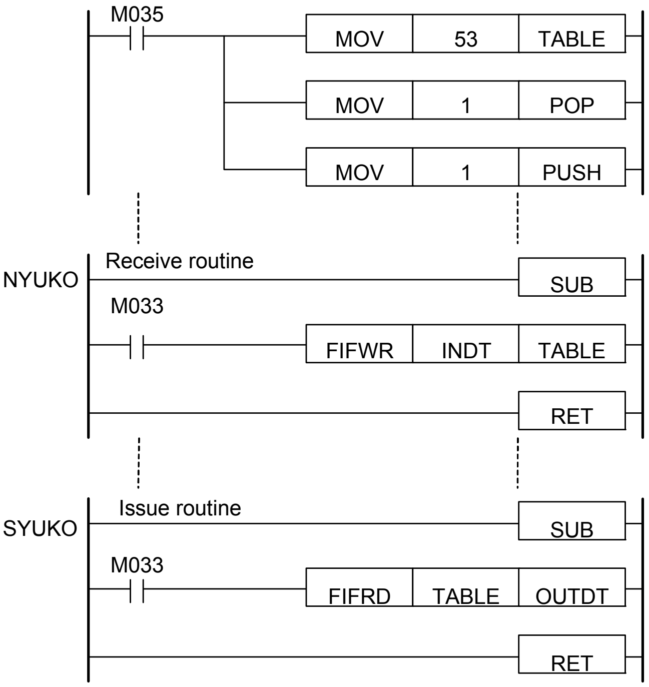


Tag name	Address	
TABLE	D0098	FIFO size (53) [in words]
POP	D0099	POP pointer position(n)*1
PUSH	D0100	PUSH pointer position(m)*1
DATA	D0101	Data 1*2
	D0102	Data 2*2
	D0103	Data 3*2
	⋮	⋮
	D0149	Data 49*2
	D0150	Data 50*2

F030805.VSD

*1 : Both POP and PUSH pointers must be initialized to 1.
*2 : The table entries must be initialized to 0.

Figure 3.8.4 Pointers in an FIFO Table



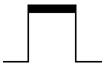
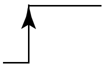
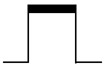
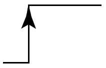
Line No.	Instruction	Operands				
0001	LD	M035				
0002	PUSH					
0003	MOV	53	TABLE			
0004	STCRD					
0005	MOV	1	POP			
0006	POP					
0007	MOV	1	PUSH			
			⋮			
NYUKO	SUB					
0051	LD	M033				
0052	FIFWR	INDT	TABLE			
0053	RET					
SYUKO	SUB					
0061	LD	M033				
0062	FIFRD	TABLE	OUTDT			
0063	RET					

F030806.VSD

Figure 3.8.5 Example of an FIFO Program

3.8.2 Binary Conversion (BIN), Long-word Binary Conversion (BIN L)

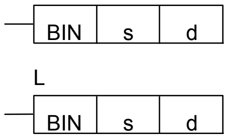
Table 3.8.3 Binary Conversion, Long-word Binary Conversion

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	52	Binary Conversion	BIN	BIN	✓	—		3	16 bits	—
	52P		↑BIN	↑ BIN				4		
	52L	Long-word Binary Conversion	BIN L	L BIN	✓	—		3	32 bits	—
	52LP		↑BIN L	↑L BIN				4		

■ Parameter

Binary Conversion

Long-word Binary Conversion



F030807.VSD

s : Data to be converted, or device number of the first device storing data to be converted to binary data
d : Device number of the first device storing the converted data

■ Available Devices

Table 3.8.4 Devices Available for Binary Conversion and Long-word Binary Conversion

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*3	✓*3	✓*3	✓*1	✓*2	✓	✓*3	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes

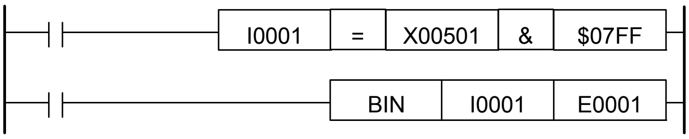
*1: Current timer value (may not be used in the Long-word Binary Conversion instruction.)
*2: Current counter value (may not be used in the Long-word Binary Conversion instruction.)
*3: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Binary Conversion and Binary Conversion Long instructions convert 16- and 32-bit BCD code to binary code, respectively. These instructions are used in combination with logical instructions in situations in which data to be stored in input relays is coded in BCD and 16- or 32-bit data contains information other than BCD code.

Use the Binary Conversion instruction to convert 16-bit data and the Binary Conversion Long instruction to convert 32-bit data.

● Example



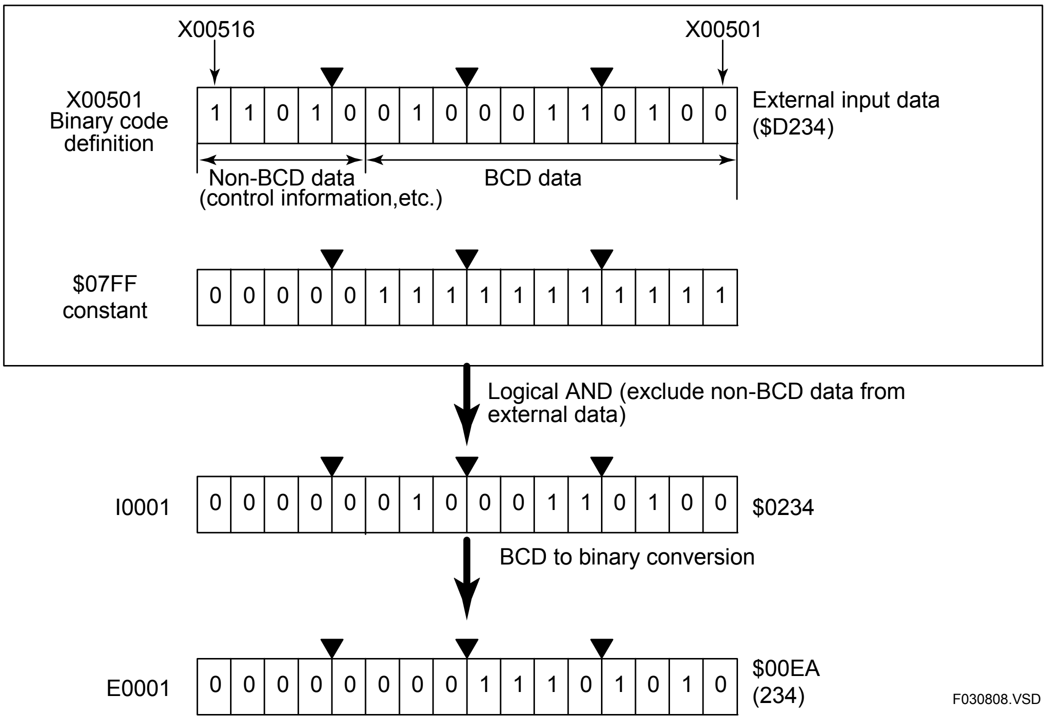


Figure 3.8.6 Example of Binary Conversion (Used with a Logical AND Instruction)

Explanation of the above Figure

When external data that is input to X00501 to X00516 consists only of BCD codes, it is automatically moved or subjected to logical operation as BCD code simply by defining X00501 to X00516 as BCD code.

In the example shown in the Figure, however, X00512 to X00516 of X00501 to X00516 carry not BCD code but control information. Defining X00501 to X00516 as BCD code will not result valid data (an instruction error will be signaled to indicate an out-of-BCD-range condition because X00513 to X00516 carry a value of $D in the above example).

To avoid this error in the example in the Figure, X00501 to X00516 are defined in binary code and is removed from the BCD code part through a logical AND. The result of logical AND is then submitted to binary conversion processing.

CAUTION

- The programmer need not perform a binary conversion when assigning data to devices whose data type is defined in binary code. Even if the source data is defined in BCD code, it is automatically converted to binary code when it is assigned to the binary devices.
- Care must be taken if the devices for storing the result of binary conversion are defined in BCD code. A reference to the value of such devices with a MOVE or logical instruction will yield a value that is different from the original value.

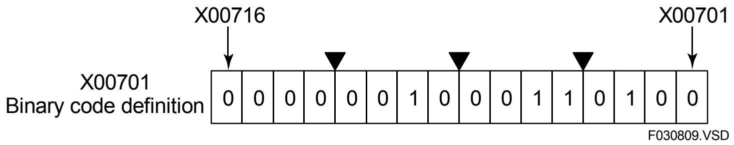
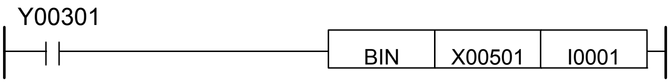


Figure 3.8.7 Notes on Binary Conversion

- Assume that X00716 to X00701 contain the result of binary conversion. They should carry 564 ($2^9 + 2^5 + 2^4 + 2^2$) in binary. If X00701 (to X00716) are used as are in a move or logical instruction, however, the bit stream will be read as BCD code, yielding 234, since the devices are defined in BCD code.

■ Programming Example

The sample code shown below converts the 16-bit value starting at X00501 to binary code and stores the result in I0001 if Y00301 is on.



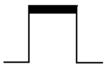
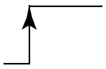
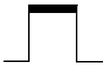
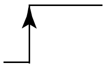
Line No.	Instruction	Operands				
0001	LD	Y00301				
0002	BIN	X00501	I0001			

F030810.VSD

Figure 3.8.8 Example of a Binary Conversion Program

3.8.3 BCD Conversion (BCD), Long-word BCD Conversion (BCD L)

Table 3.8.5 BCD Conversion, Long-word BCD Conversion

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	53	BCD Conversion	BCD	BCD	✓	—		3	16 bits	—
	53P		↑BCD	↑ BCD				4		
	53L	Long-word BCD Conversion	BCD L	L BCD	✓	—		3	32 bits	—
	53LP		↑BCD L	↑L BCD				4		

■ Parameter

BCD Conversion — BCD | s | d

Long-word BCD Conversion — L BCD | s | d

F030811.VSD

s : Data to be converted, or device number of the first device storing data to be converted to BCD data
d : Device number of the first device storing the converted data

■ Available Devices

Table 3.8.6 Devices Available for the BCD Conversion and BCD Conversion Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*3	✓*3	✓*3	✓*1	✓*2	✓	✓*3	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes

*1: Current timer value (may not be used in the Long-word BCD Conversion instruction.)
*2: Current counter value (may not be used in the Long-word BCD Conversion instruction.)
*3: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The BCD Conversion and Long-word BCD Conversion instructions convert 16- and 32-bit binary code to BCD code, respectively.

These instructions are used in combination with logical instructions in situations in which data to be stored in output relays is coded in BCD and 16- or 32-bit data contains information other than BCD code.

Use the BCD Conversion instruction to convert 16-bit data and the Long-word BCD Conversion instruction to convert 32-bit data.

If the result of BCD conversion exceeds the valid value range of BCD code, the specified devices are loaded with the lowest 16 bits for 16-bit data and with the lowest 32 bits for 32-bit data.

● Example

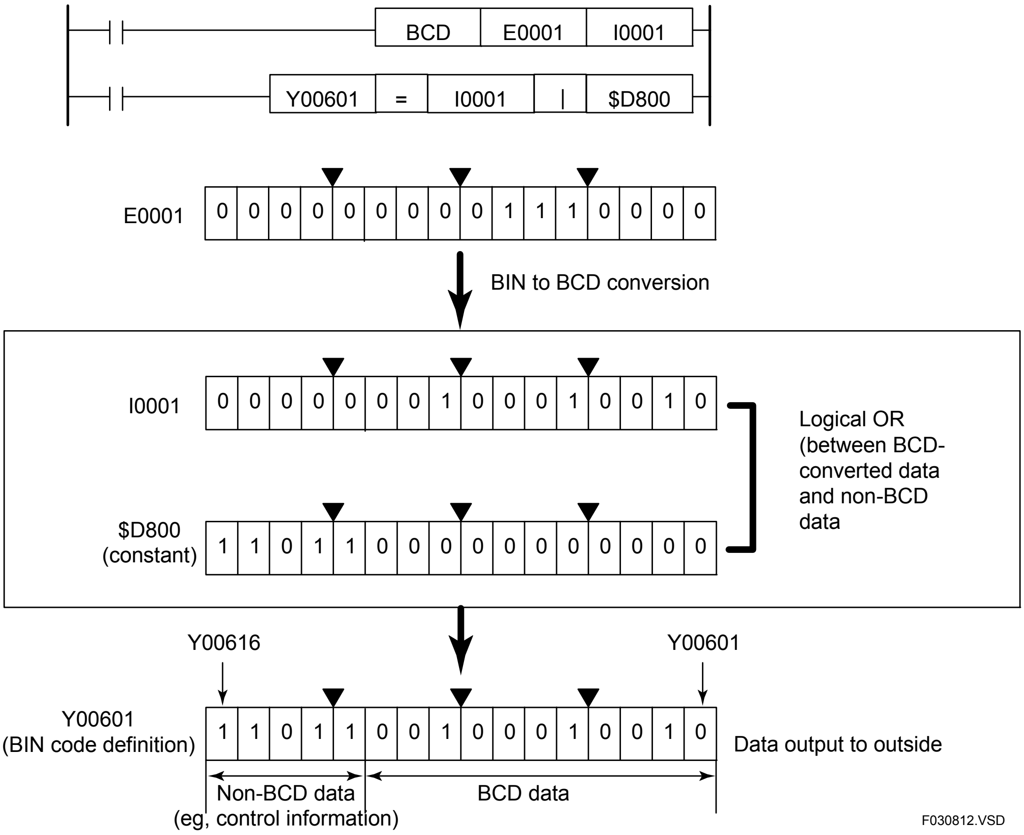


Figure 3.8.9 Example of BCD Conversion (Used with a Logical OR Instruction)

Explanation of the Above Figure

When external data that is output from Y00601 to Y00616 consists only of BCD codes, it is automatically converted into a bit stream of BCD code when it is assigned from other devices through a move or logical operation simply by defining Y00601 to Y00616 as BCD code.

In the example shown in the figure, however, Y00612 to Y00616 of Y00601 to Y00616 carry not BCD code but control information. Defining Y00601 to Y00616 as BCD code will result in an error or invalid data (an instruction error will be signaled to indicate an out-of-BCD-range condition because Y00613 to Y00616 carry a value of $D in the above example).

To avoid this error in the example in the figure, Y00601 to Y00616 are defined in binary code and subject to BCD conversion. The result of conversion is merged with the control information through a logical OR. The result of logical OR is then loaded into Y00601 to Y00616.

CAUTION

- The programmer need not perform a BCD Conversion when assigning data to devices whose data type is defined in BCD. Even if the source data is defined in binary code, it is automatically converted to BCD code when it is assigned to the BCD devices.
- Care must be taken if the devices for storing the result of BCD conversion are defined in binary code. A reference to the value of such devices with no conversion will yield a value that is different from the original value.

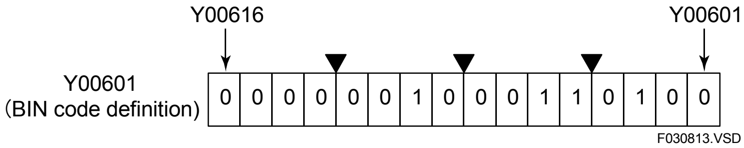
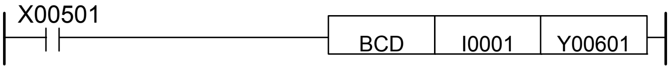


Figure 3.8.10   Notes on BCD Conversion

- Assume that Y00616 to Y00601 contain the result of BCD conversion. They should carry 234 in BCD. If Y00601 (to Y00616) are used as are in a Move or logical instruction, however, the bit stream will be read as binary code, yielding 564, since the devices are defined in binary code.

■ Programming Example

The sample code shown below converts the 16-bit value starting at I0001 to BCD code and stores the result in I00601 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	BCD	I0001	Y00601			

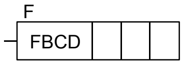
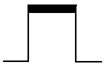
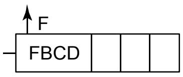
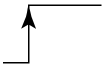
F030814.VSD

Figure 3.8.11   Example of a BCD Conversion Program

3.8.4 Float to BCD (FBCD)

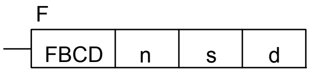
F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.8.7 Float to BCD

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	916	Float to BCD	FBCD		✓	—		5	32 bits	—
	916P		↑FBCD					6		

■ Parameter

Float to BCD



F030815.VSD

n : BCD format (always set to 0; integer/fraction separated format) (integer)
s : Device number (integer) of the first device storing data to be subject to floating-point-to-BCD conversion
d : Number (integer) of the first device storing the converted data
s must be represented in the IEEE single-precision floating-point format (32 bits).

■ Available Devices

Table 3.8.8 Devices Available for the Float to BCD Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
n	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓	✓*3	✓	✓	✓	✓	✓	Yes	Yes
s	✓	✓	✓	✓	✓	✓			✓	✓	✓*3	✓	✓	✓	✓		Yes	Yes
d		✓	✓	✓*3	✓*3	✓*3			✓	✓	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes

*1: Current timer value
*2: Current counter value
*3: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Float to BCD instruction converts single-precision floating-point data s to BCD format and places the result into the devices designated by d. The single-precision floating-point data must be represented in the IEEE format.

● BCD format

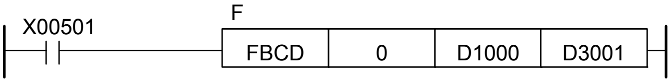
When n = 0 (fixed) (integer part/fraction part separated format)

In sAAAA.BBBB:

s : Sign (d) .... 0 represents + and 1 represents –.
AAAA : Integer part (d+1) .... 4 BCD digits
BBBB : Fraction part (d+2) .... 4 BCD digits

■ Programming Example

The sample code shown below converts the real number (IEEE single-precision floating point) in location from D1000 to D1001 to integer/fraction separate type BCD code and stores the result in the location from D3001 to D3003 if X00501 is set on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FBCD	0	D1000	D3001		

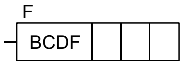
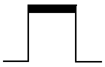
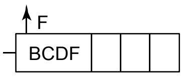
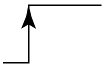
F030816.VSD

Figure 3.8.12 Example of a Floating-point to BCD Conversion Program

3.8.5 BCD to Float (BCDF)

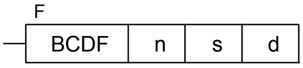
F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.8.9 BCD to Float

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	917	BCD to Float	BCDF		✓	—		5	32 bits	—
	917P		↑BCDF					6		

■ Parameter

BCD to Float



F030817.VSD

n : BCD format (always set to 0; integer/fraction separated format) (integer)
s : Device number (integer) of the first device storing data to be subject to BCD-to-floating-point conversion
d : Device number of the first device storing the converted data

d must be represented in the IEEE single-precision floating-point format (32 bits).

■ Available Devices

Table 3.8.10 Devices Available for the BCD to Float Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
n	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓	✓*3	✓	✓	✓	✓	✓	Yes	Yes
s	✓	✓	✓	✓	✓	✓			✓	✓	✓*3	✓	✓	✓	✓		Yes	Yes
d		✓	✓	✓*3	✓*3	✓*3			✓	✓	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes

*1: Current timer value
*2: Current counter value
*3: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The BCD to Float instruction converts the data s in the BCD format designated by n to single-precision floating-point data and places the result into the devices designated by d. The single-precision floating-point data is represented in the IEEE format.

● BCD format

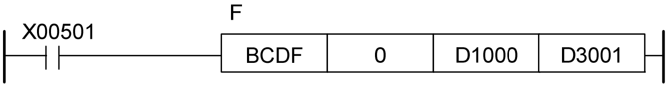
When n = 0 (fixed) (integer part/fraction part separated format)

In sAAAA.BBBB:

s	:	Sign (s)	0 represents + and 1 represents –.
AAAA	:	Integer part (s+1)	4 BCD digits
BBBB	:	Fraction part (s+2)	4 BCD digits

■ Programming Example

The sample code shown below converts the integer/fraction separate type BCD code (D1000 to D1002) to a single-precision floating-point number and stores the result in the location from D3001 to D3002 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	BCDF	0	D1000	D3001		

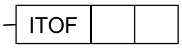
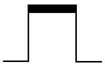
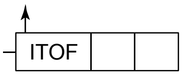
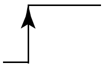
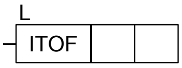
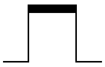
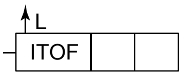
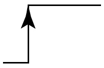
F030818.VSD

Figure 3.8.13 Example of a BCD to Floating-point Conversion Program

3.8.6 Integer to Float (ITOF), Long-word Integer to Float (ITOF L)

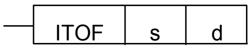
F3SP25
F3SP35 | F3SP22
F3SP28
F3SP38 | F3SP53
F3SP58
F3SP59 | F3SP66
F3SP67 | F3SP71
F3SP76

Table 3.8.11 Integer to Float, Long-word Integer to Float

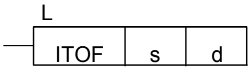
Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	901	Integer to Float	ITOF		✓	—		4	16 bits	—
	901P		↑ITOF					5		
	901L	Long-word Integer to Float	ITOF L		✓	—		4	32 bits	—
	901LP		↑ITOF L					5		

■ Parameter

Integer to Float



Long-word Integer to Float



F030819.VSD

s : Integer data to be converted or device number of the first device storing data to be converted (source)
d : Device number of the first device storing the converted data (destination)
The destination is 2 words long (32 bits) regardless of whether the source is word or long word data.

■ Available Devices

Table 3.8.12 Devices Available for the Integer to Float and Long-word Integer to Float Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓	✓*3	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*3	✓*3	✓*3			✓	✓	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes

*1: Current timer value (may not be used in the Long-word Integer to Float instruction.)
*2: Current counter value (may not be used in the Long-word Integer to Float instruction.)
*3: See Section 1.17, "Devices Available As Instruction Parameters."

Function

The Integer to Float and Long-word Integer to Float instructions convert 16- and 32-bit integer data to 32-bit single-precision floating-point data, respectively. Before performing any floating-point operation, convert any integer data to floating-point data with these instructions.

The single-precision floating-point data is represented in the IEEE format.

Example of a conversion

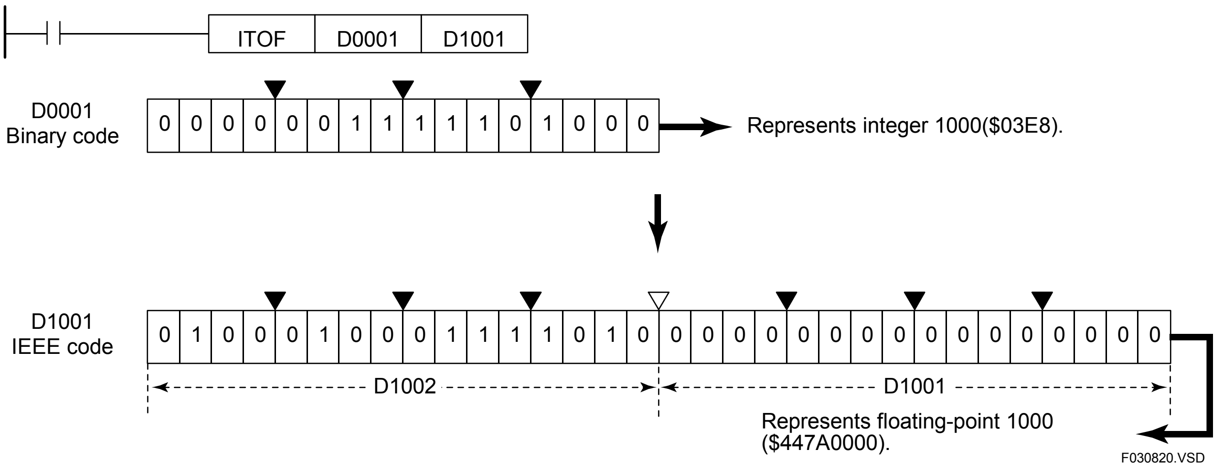
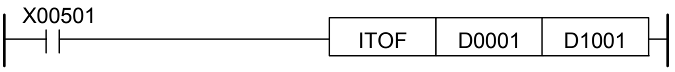


Figure 3.8.14 Example of an Integer-to-floating-point Conversion

Programming Example

The sample code shown below converts 1 word of integer data at D0001 to an IEEE single-precision floating-point number and stores the result in the location from D1001 to D1002 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	ITOF	D0001	D1001			

F030821.VSD

Figure 3.8.15 Example of an Integer to Floating-point Conversion Program

CAUTION

Data may be rounded during the execution of the Long-word Integer to Float instruction.

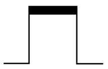
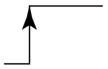
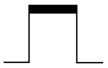
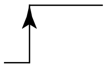
SEE ALSO

For details, see Section 1.6, "Floating-point Processing."

3.8.7 Long-word Integer to Double-precision Float (ITOE L), Double Long-word Integer to Double-precision Float (ITOE D)

F3SP71
F3SP76

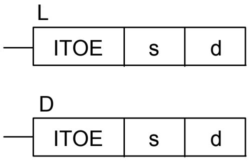
Table 3.8.13 Long-word Integer to Double-precision Float, Double Long-word Integer to Double-precision Float

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	920L	Long-word integer to Double-precision Float	ITOE L	L ITOE	✓	—		4	32 bits	—
	920LP		↑ITOE L	↑L ITOE				5		
	921D	Double Long-word Integer to Double-precision Float	ITOE D	D ITOE	✓	—		4	64 bits	—
	921DP		↑ITOE D	↑D ITOE				5		

■ Parameter

Long-word Integer to Double-precision Float

Double Long-word Integer to Double-precision Float



F387001.VSD

s : Long-word integer data or double long-word integer data to be converted, or device number of the first device storing data to be converted (source)
d : Device number of the first device storing the converted double-precision floating-point data (destination)
The destination is 4 words long (64 bits) regardless of whether the source is long-word or double long-word data.

■ Available Devices

Table 3.8.14 Devices Available for the Long-word Integer to Double-precision Float and Double Long-word Integer to Double-precision Float Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s									✓	✓*1	✓*1	✓		✓		✓	Yes	Yes
d									✓	✓*1	✓*1	✓*1		✓*1			Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

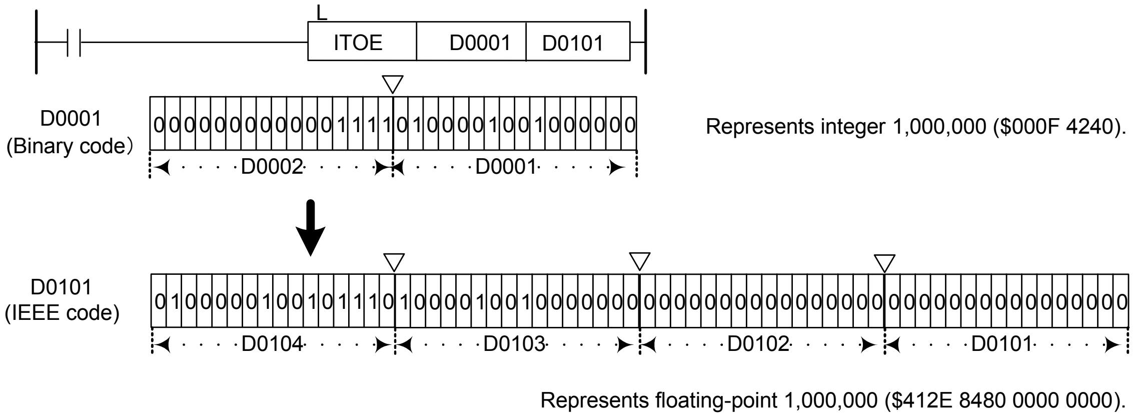
Function

The Long-word Integer to Double-precision Float and Double Long-word Integer to Double-precision Float instructions convert 32- and 64-bit integer data to 64-bit double-precision floating-point data, respectively.

Before performing any double-precision floating-point operation, convert any integer data to double-precision floating-point data with these instructions.

The double-precision floating-point data is represented in the IEEE format.

● Example of a conversion

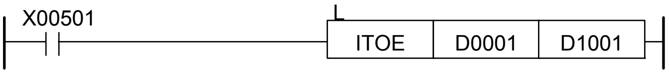


F387002.VSD

Figure 3.8.16 Example of a Long-word Integer to Double-precision Floating-point Conversion

Programming Example

The sample code shown below converts 2 words of integer data from D0001 to D0002 to an IEEE double-precision floating-point data and stores the result in the location from D1001 to D1004 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	ITOE L	D0001	D1001			

F387003.VSD

Figure 3.8.17 Example of a Long-word Integer to Double-precision Floating-point Conversion Program

CAUTION

Data may be rounded during the execution of a Double Long-word Integer to Double-precision Float instruction.

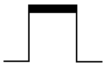
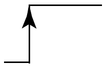
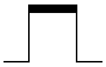
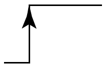
SEE ALSO

For details, see Section 1.7, "Floating-point Processing."

3.8.8 Float to Integer (FTOI), Float to Long-word Integer (FTOI L)

F3SP25
F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

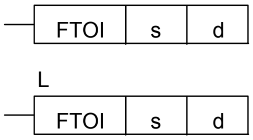
Table 3.8.15 Float to Integer, Float to Long-word Integer

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	902	Float to Integer	FTOI	FTOI	✓	—		4	16 bits	—
	902P		↑FTOI	↑ FTOI				5		
	902L	Float to Long-word Integer	FTOI L	L FTOI	✓	—		4	32 bits	—
	902LP		↑FTOI L	↑L FTOI				5		

■ Parameter

Float to Integer

Float to Long-word Integer



F030822.VSD

s : Device number of the first device storing data to be converted (source)
d : Device number of the first device storing the converted integer data (destination)

■ Available Devices

Table 3.8.16 Devices Available for the Float to Integer and Float to Long-word Integer Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓		Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Current timer value (may not be used in the Float to Long-word Integer instruction.)
*3: Current counter value (may not be used in the Float to Long-word Integer instruction.)

■ Function

The Float to Integer and Float to Long-word Integer instructions convert single-precision floating-point data (32 bits) to 16- and 32-bit integer data, respectively. Before using the result of any floating-point operation in an application instruction as an integer, convert the floating-point data to integer data with these instructions.

The single-precision floating-point data must be represented in the IEEE format.

● Example of a conversion

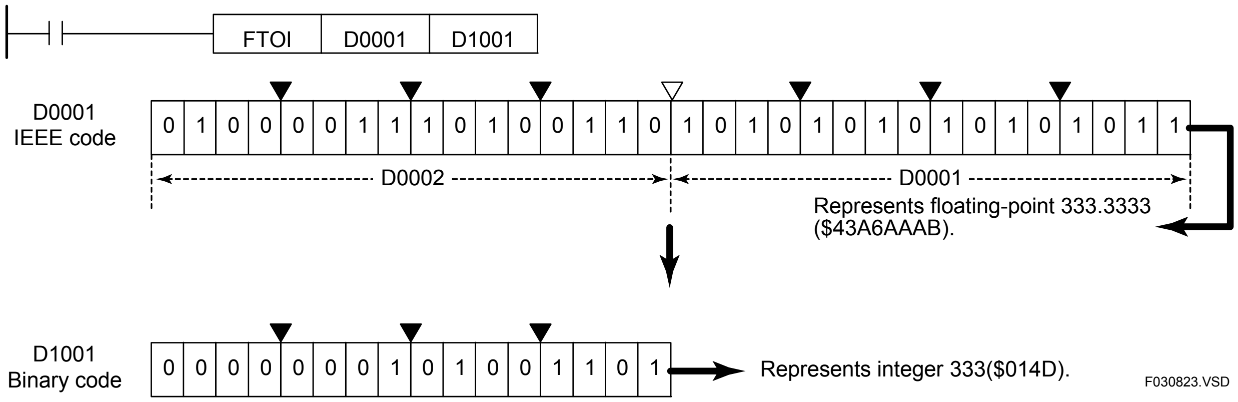
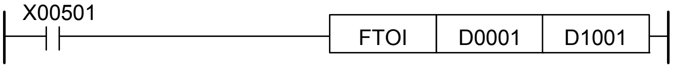


Figure 3.8.18 Example of a Floating-point to Integer Conversion

■ Programming Example

The sample code shown below converts the IEEE floating-point data in location from D0001 to D0002 to an integer and stores the result in the location D1001 if X00501 is on.



Line No.	Instruction	Operands					
0001	LD	X00501					
0002	FTOI L	D0001	D1001				

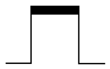
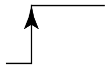
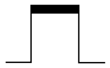
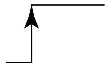
F030824.VSD

Figure 3.8.19 Example of a Floating-point to Integer Conversion Program

3.8.9 Double-precision Float to Long-word Integer (ETOI L), Double-precision Float to Double Long-word Integer (ETOI D)

F3SP71
F3SP76

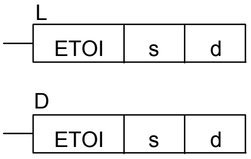
Table 3.8.17 Double-precision Float to Long-word Integer, Double-precision Float to Double Long-word Integer

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	922L	Double-precision Float to Long-word Integer	ETOI L	L ETOI	✓	—		4	32 bits	—
	922LP		↑ETOI L	↑L ETOI				5		
	923D	Double-precision Float to Double Long-word Integer	ETOI D	D ETOI	✓	—		4	64 bits	—
	923DP		↑ETOI D	↑D ETOI				5		

■ Parameter

Double-precision Float to Long-word Integer

Double-precision Float to Double Long-word Integer



F389001.VSD

s : Double-precision floating-point data to be converted
or device number of the first device storing data to be converted (source)

d : Device number of the first device storing the converted integer data (destination)

■ Available Devices

Table 3.8.18 Devices Available for the Double-precision Float to Long-word Integer and Double-precision Float to Double Long-word Integer Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s									✓	✓*1	✓*1	✓		✓		✓	Yes	Yes
d									✓	✓*1	✓*1	✓*1		✓*1			Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Double-precision Float to Long-word Integer and Double-precision Float to Double Long-word Integer instructions convert IEEE double-precision floating-point data (64 bits) to 32- and 64-bit integer data, respectively. Before using the result of any double-precision floating-point operation in an application instruction as an integer, convert the double-precision floating-point data to integer data with these instructions.

The double-precision floating-point data must be represented in the IEEE format.

● Example of a conversion

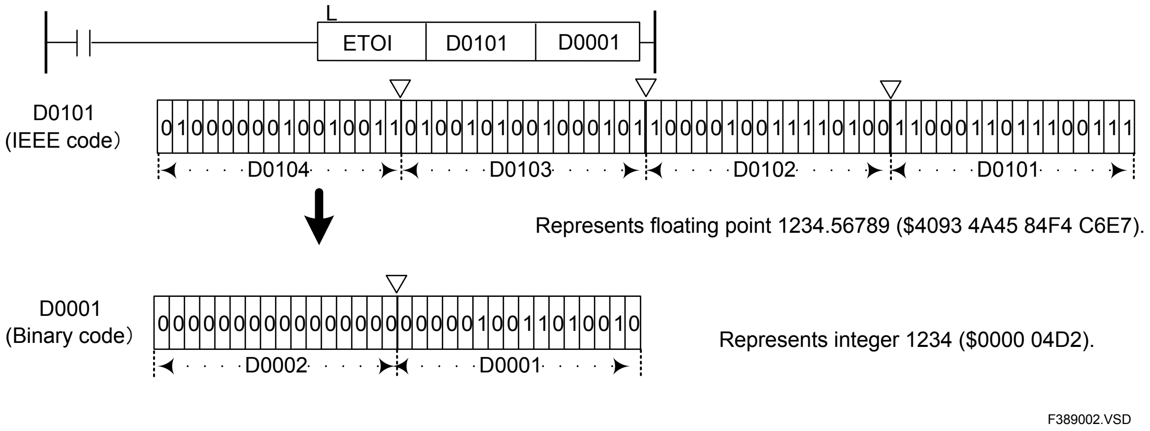
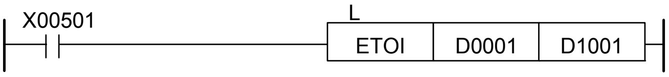


Figure 3.8.20 Example of a Double-precision Floating-point to Long-word Integer Conversion

■ Programming Example

The sample code shown below converts the IEEE double-precision floating-point data in location from D0001 to D0004 to a long-word integer and stores the result in the location from D1001 to D1002 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	ETOI L	D0001	D1001			

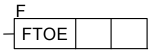
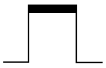
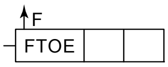
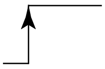
F389003.VSD

Figure 3.8.21 Example of a Double-precision Floating-point to Long-word Integer Conversion Program

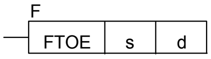
3.8.10 Float to Double-precision Float (FTOE)

F3SP71
F3SP76

Table 3.8.19 Float to Double-precision Float

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	925F	Single-precision Float to Double-precision Float	FTOE		✓	—		4	32 bits	—
	925FP		↑FTOE					5		

■ Parameter

Single-precision Float to Double-precision Float 

F3810001.VSD

s : Single-precision floating-point data to be converted
or device number of the first device storing data to be converted (source). It is handled as 2 words data.

d : Device number of the first device storing the converted double-precision floating-point data (destination).
It is handled as 4 words data.

■ Available Devices

Table 3.8.20 Devices Available for the Single-precision Float to Double-precision Float Instruction

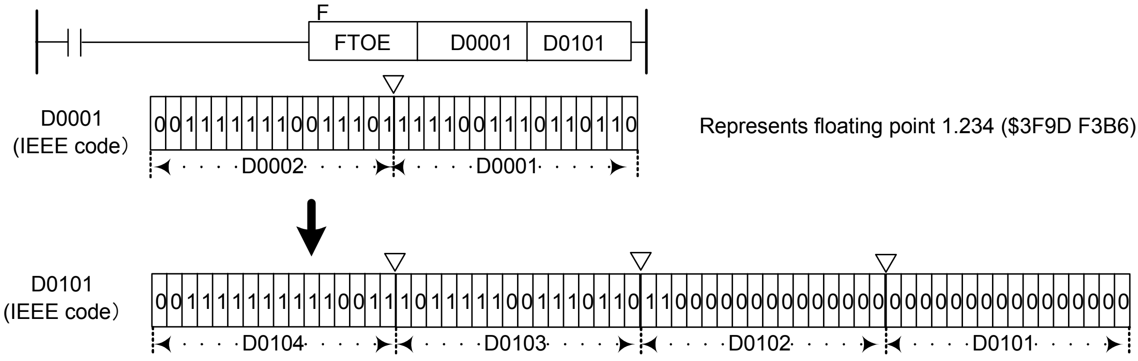
Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s									✓	✓*1	✓*1	✓		✓		✓	Yes	Yes
d									✓	✓*1	✓*1	✓*1		✓*1			Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Float to Double-precision Float instruction converts IEEE single-precision floating point data (32-bit data) to IEEE double-precision floating-point data (64-bit data).

● Example of a conversion



Represents floating point 1.234 ($3F9D F3B6).

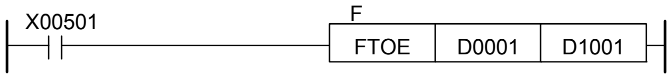
Represents double-precision floating-point 1.234 ($3FF3 BE76 C000 0000).

F381002.VSD

Figure 3.8.22 Example of a Floating-point to Double-precision Floating-point Conversion

■ Programming Example

The sample code shown below converts single-precision floating-point data at location from D0001 to D0002 to IEEE double-precision floating-point data and stores the result in the location from D1001 to D1004 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FTOE	D0001	D1001			

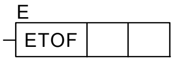
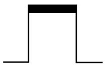
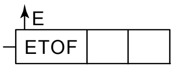
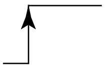
F3810003.VSD

Figure 3.8.23 Example of a Floating-point to Double-precision Floating-point Conversion Program

3.8.11 Double-precision Float to Float (ETOF)

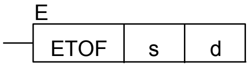
F3SP71
F3SP76

Table 3.8.21 Double-precision Float to Float

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Application Instruction	926E	Double-precision Float to Single-precision Float	ETOF	 E ETOF	✓	—		4	64 bits	—
	926EP		↑ETOF	 ↑E ETOF				5		

■ Parameter

Double-precision Float to Single-precision Float



F3811001.VSD

s : Double-precision floating-point data to be converted
or device number of the first device storing data to be converted (source). It is handled as 4 words data.

d : Device number of the first device storing the converted single-precision floating-point data (destination). It is handled as 2 words data.

■ Available Devices

Table 3.8.22 Devices Available for the Single-precision Float to Double-precision Float Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s									✓	✓*1	✓*1	✓		✓		✓	Yes	Yes
d									✓	✓*1	✓*1	✓*1		✓*1			Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Double-precision Float to Float instruction converts IEEE double-precision floating point data (64-bit data) to IEEE single-precision floating-point data (32-bit data).

● Example of a conversion

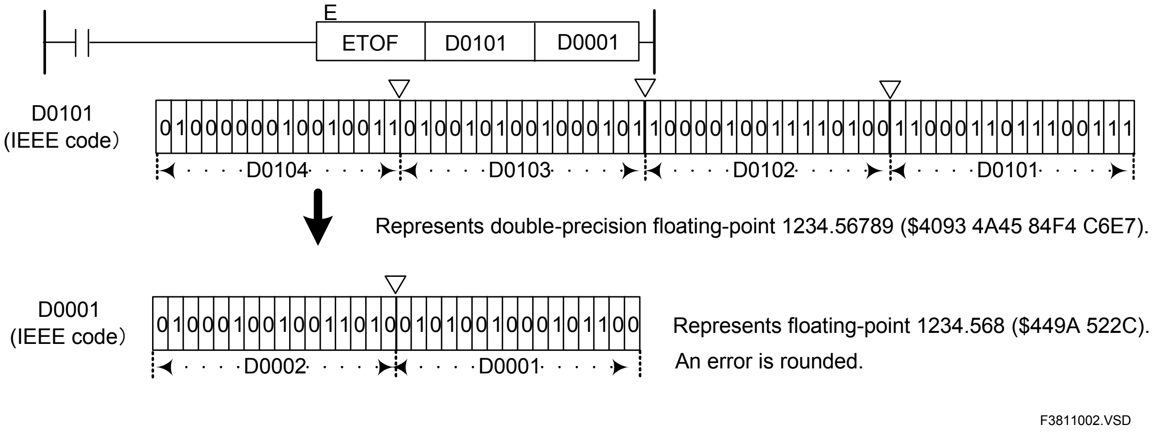
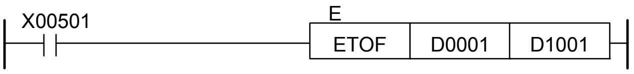


Figure 3.8.24 Example of a Double-precision Floating-point to Floating-point Conversion

■ Programming Example

The sample code shown below converts double-precision floating-point data at location from D0001 to D0004 to IEEE single-precision floating-point data and stores the result in the location from D1001 to D1002 if X00501 is on.



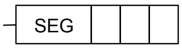
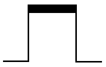
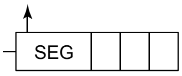
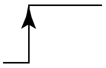
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	ETOF	D0001	D1001			

F3811003.VSD

Figure 3.8.25 Example of a Double-precision Floating-point to Floating-point Conversion Program

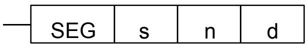
3.8.12 7-segment Decoder (SEG)

Table 3.8.23 7-segment Decoder

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	54	7-segment Decoder	SEG		✓	—		4	16 bits	—
	54P		↑SEG					5		

■ Parameter

7-segment Decoder



F030825.VSD

s : Device number of the first device storing the data to be converted
n : Digit position of data to be converted (0 to 3)
d : Device number of the first device storing the converted data

■ Available Devices

Table 3.8.24 Devices Available for the 7-segment Decoder Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
n	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*3	✓*3	✓*3	✓*1	✓*2	✓	✓*3	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes

*1: Timer current value
*2: Counter current value
*3: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

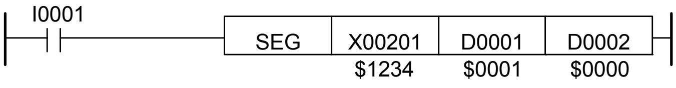
The 7-segment Decoder instruction decodes the value ($0 to F) in the nth digit (source position) of 16-bit data designated by device s to 7-segment LED display data and places the result in the devices designated by d.

The highest-order 8 bits (bits 8 to 15) of the decoded data are padded with 0s ($00). The least significant digit (bits 0 to 3) of the digit position n is 0 ($0000) and the most significant digit (bits 12 to 15) is 3 ($0003).

If n does not fall within a value range of 0 to 3, an error is signaled and the execution of the 7-segment Decoder instruction is suppressed.

■ Programming Example

The sample code shown below decodes the digit designated by D0001 ($0001) of data at X00201 ($1234) to 7-segment data and places the result in D0002 if I0001 is on.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	SEG	X00201	D0001	D0002		

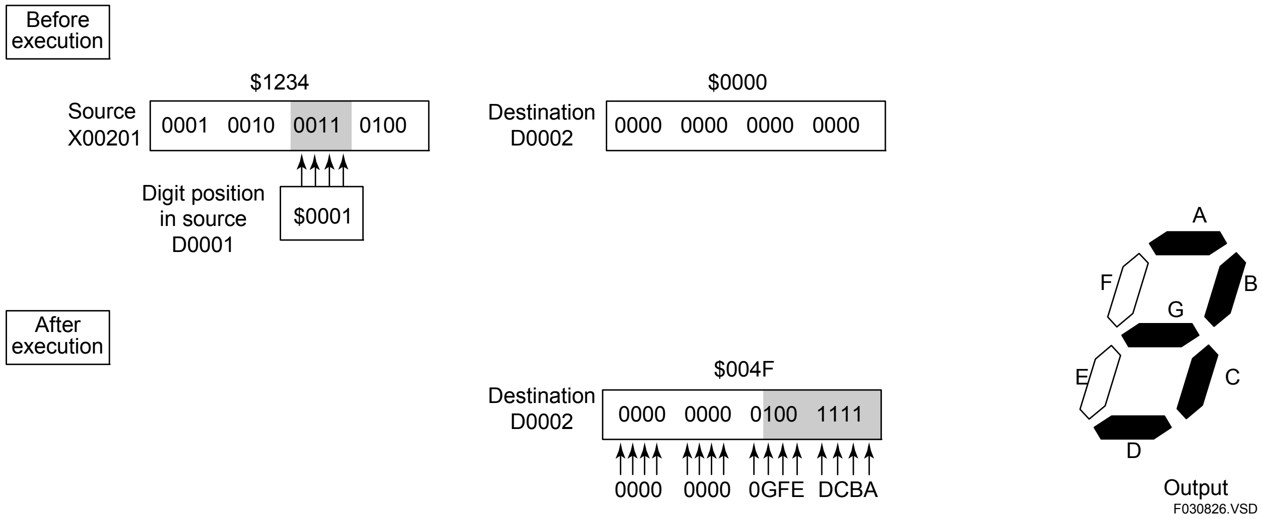
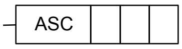
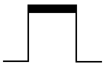
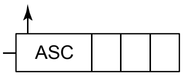
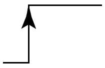


Figure 3.8.26 Example of a 7-segment Decoder Program

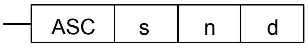
3.8.13 Convert ASCII (ASC)

Table 3.8.25 Convert ASCII

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	55	Convert ASCII	ASC		✓	—		4	16 bits	—
	55P		↑ASC					5		

■ Parameter

Convert ASCII



F030827.VSD

s : Device number of the first device storing the data to be converted
n : Digit position of data to be converted (0 to 3)
d : Device number of the first device storing the converted data

■ Available Devices

Table 3.8.26 Devices Available for the Convert ASCII Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
n	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*3	✓*3	✓*3	✓*1	✓*2	✓	✓*3	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes

*1: Timer current value
*2: Counter current value
*3: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

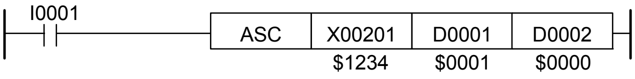
The Convert ASCII instruction converts the value ($0 to F) in the nth digit (source position) of 16-bit data designated by device s to an ASCII code ('0' = $30 to 'F' = $46) and places the result in the devices designated by d.

The highest-order 8 bits (bits 8 to 15) of the converted data are padded with 0s ($00). The least significant digit (bits 0 to 3) of the digit position n is 0 ($0000) and the most significant digit (bits 12 to 15) is 3 ($0003).

If n does not fall within a value range of 0 to 3, an error is signaled and the execution of the Convert ASCII instruction is suppressed.

■ Programming Example

The sample code shown below converts the digit designated by D0001 ($0001) of data at X00201 ($1234) to an ASCII code places the result in D0002 if I0001 is on.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	ASC	X00201	D0001	D0002		

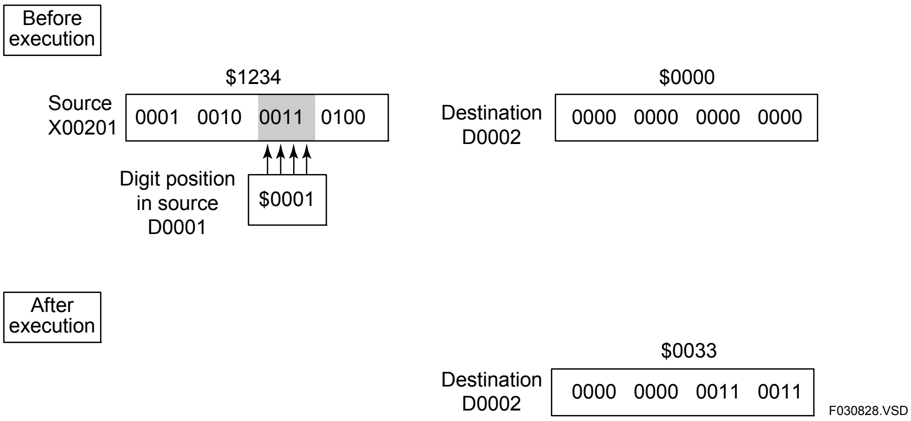
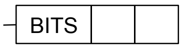
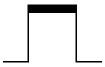
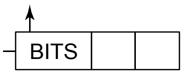
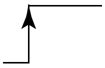
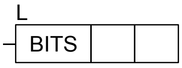
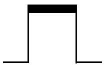
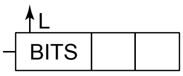
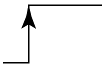
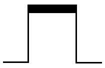
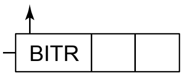
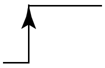
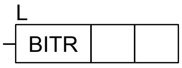
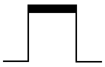
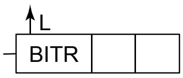
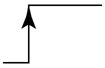


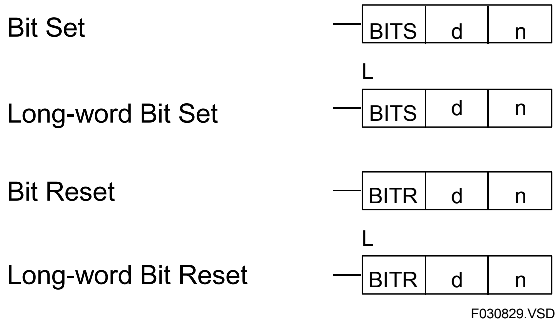
Figure 3.8.27 Example of a Convert ASCII Program

3.8.14 Bit Set (BITS), Long-word Bit Set (BITS L), Bit Reset (BITR), Long-word Bit Reset (BITR L)

Table 3.8.27 Bit Set, Bit Reset

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	56	Bit Set	BITS		✓	—		3	16 bits	—
	56P		↑BITS					4		
	56L	Long-word Bit Set	BITS L		✓	—		3	32 bits	—
	56LP		↑BITS L					4		
	57	Bit Reset	BITR		✓	—		3	16 bits	—
	57P		↑BITR					4		
	57L	Long-word Bit Reset	BITR L		✓	—		3	32 bits	—
	57LP		↑BITR L					4		

■ Parameter



d : Device number of the first device storing the data to be bit-set or bit-reset

n : Bit position of the data to be bit-set or bit-reset (0 to 15, 0 to 31)[*1]

*1 : n is handled as a word even in a 32-bit (long-word) instruction.

■ Available Devices

Table 3.8.28 Devices Available for the Bit Set or Bet Reset Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
n	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value (may not be used as a long-word parameter)
*3: Counter current value (may not be used as a long-word parameter)

■ Function

(1) Bit Set

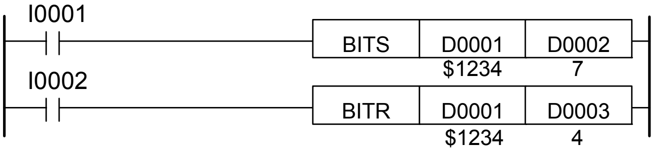
The Bit Set and Long-word Bit Set instructions set bit n of 16- and 32-bit data (d) to ON, respectively. Use the Bit Set instruction to set 16-bit data and the Long-word Bit Set instruction to set 32-bit data.

(2) Bit Reset

The Bit Reset and Long-word Bit Reset instructions reset bit n of 16- and 32-bit data (d) to OFF, respectively. Use the Bit Reset instruction to reset 16-bit data and the Long-word Bit Reset instruction to reset 32-bit data.

Programming Example

The sample code shown below sets bit 7 of D0001 ($1234) to ON if I0001 is on and bit 4 to OFF if I0002 is on.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	BITS	D0001	D0002			
0003	LD	I0002				
0004	BITR	D0001	D0003			

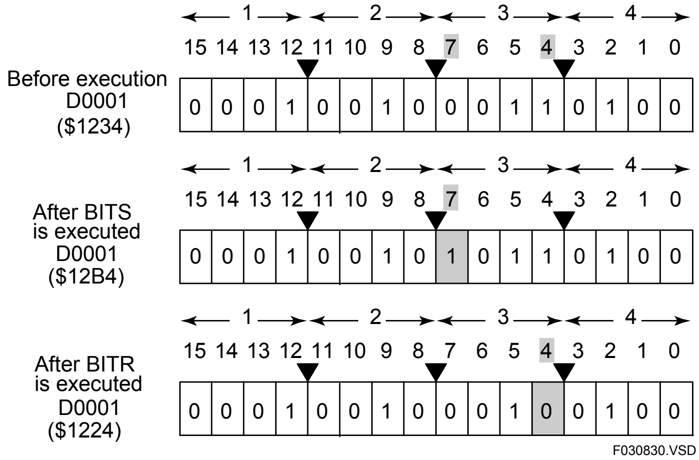
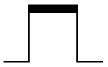
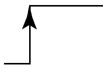
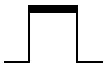
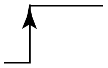


Figure 3.8.28 Example of a Bit Set/Reset Program

3.8.15 Carry Set (CSET), Carry Reset (CRST)

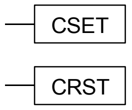
Table 3.8.29 Carry Set, Carry Reset

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	151	Carry Set	CSET	CSET	✓	—		1	—	—
	151P		↑CSET	↑ CSET				2		
	152	Carry Reset	CRST	CRST	✓	—		1	—	—
	152P		↑CRST	↑ CRST				2		

■ Parameter

Carry Set

Carry Reset



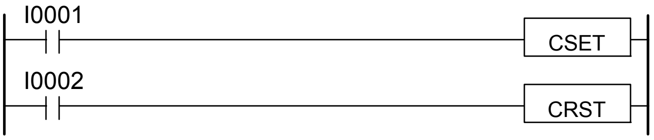
F030831.VSD

■ Function

The Carry Set and Carry Reset instructions set and reset the carry flag (special relay M188), respectively.

■ Programming Example

The sample code shown below sets the carry flag if I0001 is on and resets if I0002 is on.



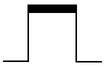
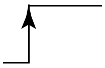
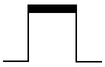
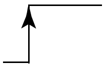
Line No.	Instruction	Operands					
0001	LD	I0001					
0002	CSET						
0003	LD	I0002					
0004	CRST						

F030832.VSD

Figure 3.8.29 Example of a Carry Set/Reset Program

3.8.16 Distribute Data (DIST), Distribute Long-word Data (DIST L)

Table 3.8.30 Distribute Data, Distribute Long-word Data

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	153	Distribute Data	DIST	DIST	✓	—		3	16 bits	—
	153P		↑DIST	↑ DIST				4		
	153L	Distribute Long-word Data	DIST L	L DIST	✓	—		3	32 bits	—
	153LP		↑DIST L	↑L DIST				4		

■ Parameter

Distribute Data —DIST s d

Distribute Long-word Data —L DIST s d

F030833.VSD

s : Device number of the first device storing the source data to be distributed
d : Device number of the first device storing the distributed data

■ Available Devices

Table 3.8.31 Devices Available for the Distribute Data and Distribute Long-word Data Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*3	✓*3	✓*3	✓*1	✓*2	✓	✓*3	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes

*1: Timer current value (may not be used as a long-word parameter)
*2: Counter current value (may not be used as a long-word parameter)
*3: See Section 1.17, "Devices Available As Instruction Parameters."

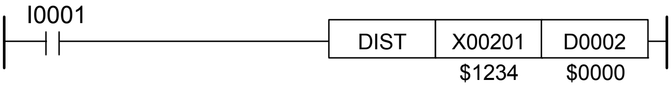
■ Function

The Distribute Data and Distribute Long-word Data instructions extract digits from 16- and 32-bit data, respectively, 1 digit (4 bits) at a time starting at the least significant bit side and distribute them into 4 words and 8 long words areas, respectively, designated by device d. The highest order 12 bits or 28 bits of the destination are padded with 0s.

The flow of data in these instructions is opposite to that in the UNIT instructions.

■ Programming Example

The sample code shown below distributes the digits from X00201 ($1234) into the location from D0002 to D0005, 1 digit (4 bits) at a time, if I0001 is on.



Line No.	Instruction	Operands			
0001	LD	I0001			
0002	DIST	X00201	D0002		

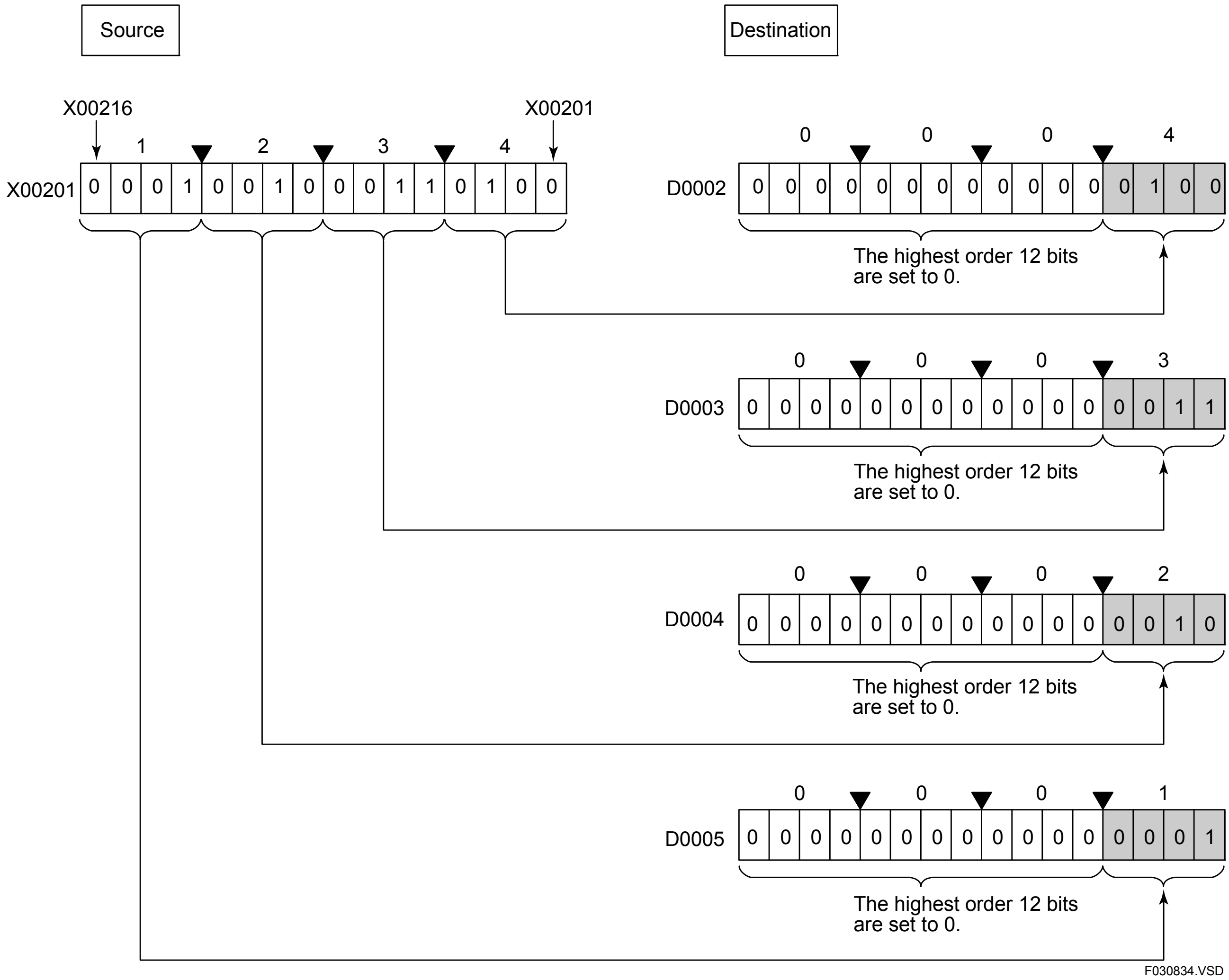
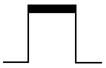
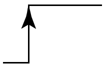
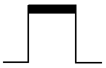
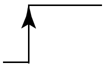

Source
Destination
X00216
X00201
1 2 3 4
X00201 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0
D0002 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 4
The highest order 12 bits are set to 0.
D0003 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 3
The highest order 12 bits are set to 0.
D0004 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 2
The highest order 12 bits are set to 0.
D0005 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 1
The highest order 12 bits are set to 0.
F030834.VSD

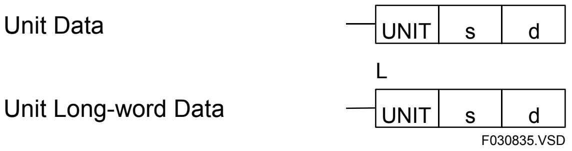
Figure 3.8.30 Example of a Distribute Program

3.8.17 Unit Data (UNIT), Unit Long-word Data (UNIT L)

Table 3.8.32 Unit Data, Unit Long-word Data

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	154	Unit Data	UNIT	UNIT	✓	—		3	16 bits	—
	154P		↑UNIT	↑ UNIT				4		
	154L	Unit Long-word Data	UNIT L	L UNIT	✓	—		3	32 bits	—
	154LP		↑UNIT L	↑L UNIT				4		

■ Parameter



s : Device number of the first source device storing the source data to be extracted
d : Device number of the first destination device for storing the extracted data

■ Available Devices

Table 3.8.33 Devices Available for the Unit Data and Unit Long-word Data Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*3	✓*3	✓*3	✓*1	✓*2	✓	✓*3	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes

*1: Timer current value (may not be used as a long-word parameter)
*2: Counter current value (may not be used as a long-word parameter)
*3: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

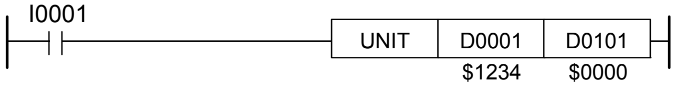
The Unit Data and Unit Long-word Data instructions extract digits from 4 words and 8 long words data, respectively, 1 digit (4 bits) at a time starting at the lowest-order bit side and place them into 16- and 32-bit areas, respectively, designated by device d.

The flow of data in these instructions is opposite to that in the Distribute (DIST) instructions.

If s is a constant, the lowest-order 4 bits are loaded into each digit position of d.

■ Programming Example

The sample code shown below extracts and digits from D0001 to D0004 and places 4 digits of 4-bit data in D0101 if I0001 is on.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	UNIT	D0001	D0101			

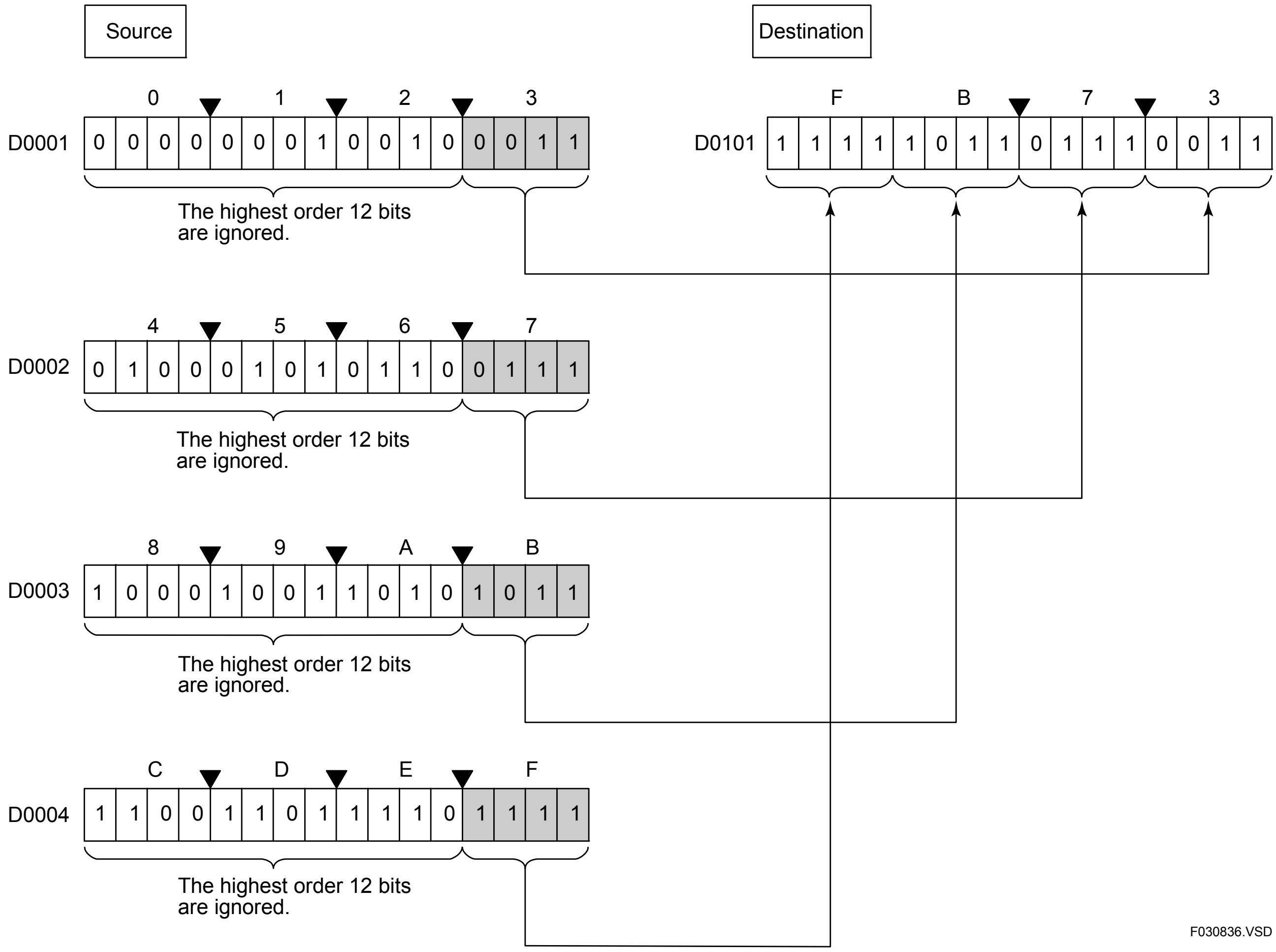
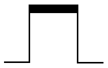
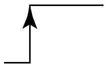
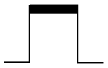
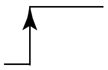

Source
Destination
0 1 2 3
D0001 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1
The highest order 12 bits are ignored.
4 5 6 7
D0002 0 1 0 0 0 1 0 1 0 1 1 0 0 1 1 1
The highest order 12 bits are ignored.
8 9 A B
D0003 1 0 0 0 1 0 0 1 1 0 1 0 1 0 1 1
The highest order 12 bits are ignored.
C D E F
D0004 1 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1
The highest order 12 bits are ignored.
F B 7 3
D0101 1 1 1 1 1 0 1 1 0 1 1 1 0 0 1 1
F030836.VSD

Figure 3.8.31 Example of a Unit Program

3.8.18 Decode (DECO), Encode (ENCO)

Table 3.8.34 Decode, Encode

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	155	Decode	DECO	DECO	✓	—		5	—	—
	155P		↑DECO	↑ DECO				6		
	156L	Encode	ENCO	ENCO	✓	—		5	—	—
	156LP		↑ENCO	↑ ENCO				6		

■ Parameter

Decode	DECO	s	n1	d	n2
Encode	ENCO	s	n1	d	n2

F030837.VSD

s : Device number of the first source device
n1 : Number of bits of source data (1 to 8 for decoding and 1 to 256 for encoding)*1
d : Device number of the first destination device
n2 : Number of bits of destination data (1 to 256 for decoding and 1 to 8 for encoding)*1
A value greater than the above listed ranges may be specified if the indirect mode (via a register or relay) is used.

*1: n1 and n2 are handled as a word.

■ Available Devices

Table 3.8.35 Devices Available for the Decode and Encode Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
n1	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*3	✓*3	✓*3	✓*1	✓*2	✓	✓*3	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes
n2	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes

*1: Timer current value (may not be used as s or d for data 17 bits or longer)
*2: Counter current value (may not be used as s or d for data 17 bits or longer)
*3: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Decode and Encode instructions decode n1 bits of data in the devices designated by s and load the results into the lowest-order n2 bits of the devices designated by d. The bits other than the lowest order n2 bits of d retain the old values.

The most significant bit of value 1 is encoded in an encode operation if there is more than one "1" bit in the (n1-bit) source data (priority encoder).

CAUTION

If there is no "1" bit, an error is signaled and the special relay M201 is set to ON.

● Number of bits required for decoding or encoding

The numbers of destination bits required to decode n1 bits are given below. If the bit count of destination (n2) is greater than these values, the extra bit positions are padded with 0s (example 1).

If the number of decoded bits is greater than the number of destination bits (n2), the extra highest-order bits are discarded (example 2).

The extra bits in the decoded data retain the old bit value (example 2).

● Number of bits required for 8 to 256 bits decoding

Table 3.8.36 Number of bits required for 8 to 256 bits decoding

Number of Source Bits	Number of Destination Bits Required
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256

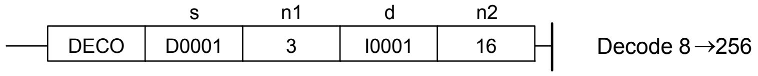
● Number of bits required for 256 to 8 bits encoding

Table 3.8.37 Number of bits required for 256 to 8 bits encoding

Number of Source Bits	Number of Destination Bits Required
1 to 2	1
3 to 4	2
5 to 8	3
9 to 16	4
17 to 32	5
33 to 64	6
65 to 128	7
129 to 256	8

Example 1:

When the bit count of destination (n2) is greater than the number of required destination bits



Number of Source Bits (n1)	Number of Destination Bits Required
3	8

F030839.VSD

Figure 3.8.32 When Bit Count of Destination > Number of Required Destination Bits (1/2)

Since decoding a 3-bit binary value (000 to 111) yields a (0 to 7) decimal number, 8 bits of destination bits are required. If the bit count of destination (n2) is greater than this required destination bit count, the higher bit positions are padded with 0s.

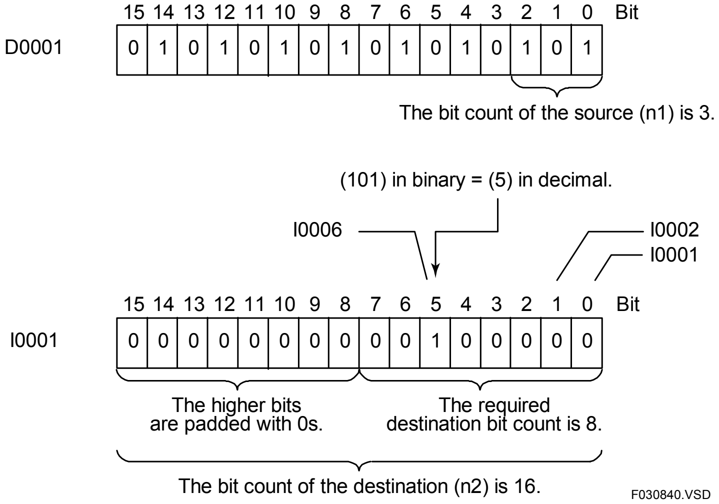

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 Bit
D0001
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
The bit count of the source (n1) is 3.
(101) in binary = (5) in decimal.
I0006
I0002
I0001
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 Bit
I0001
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
The higher bits are padded with 0s.
The required destination bit count is 8.
The bit count of the destination (n2) is 16.
F030840.VSD

Figure 3.8.33 When Bit Count of Destination > Number of Required Destination Bits (2/2)

Example 2:

When the number of required destination bits is greater than the bit count of destination (n2)

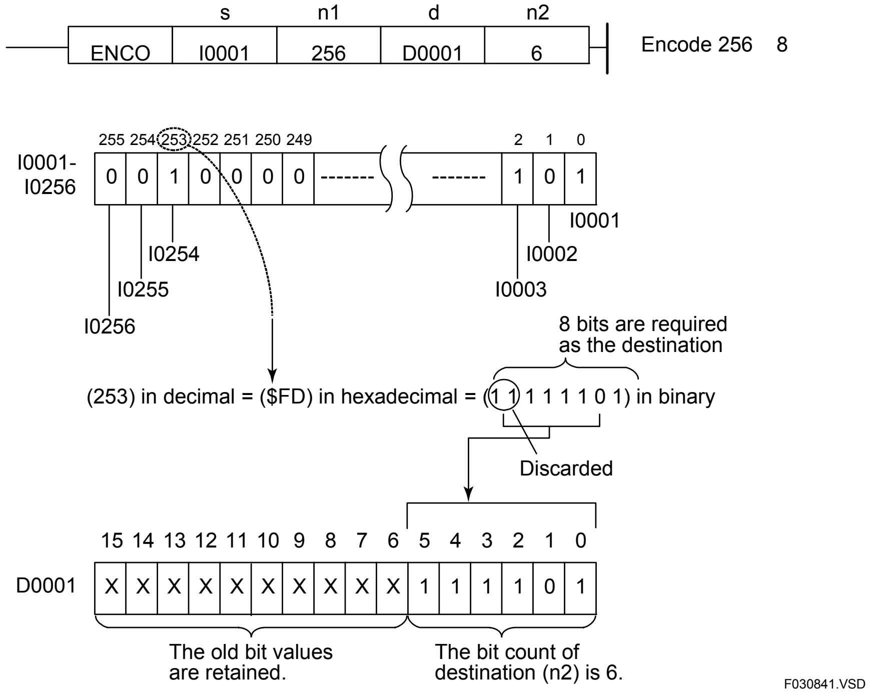
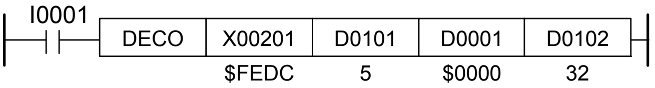

s
n1
d
n2
ENCO
I0001
256
D0001
6
Encode 256 8
255 254 253 252 251 250 249
2 1 0
I0001-
I0256
0 0 1 0 0 0 0 ------- ------- 1 0 1
I0001
I0002
I0003
I0254
I0255
I0256
8 bits are required as the destination
(253) in decimal = ($FD) in hexadecimal = (1 1 1 1 1 1 0 1) in binary
Discarded
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
D0001
X X X X X X X X X X 1 1 1 1 0 1
The old bit values are retained.
The bit count of destination (n2) is 6.
F030841.VSD

Figure 3.8.34 When Number of Required Destination Bits > Bit Count of Destination

■ Programming Example

● Decoding

The sample code shown below decodes the lowest-order 5 bits (X00201 to X00205) of X00201 ($FEDC) into 32 bits in D0001 starting at the least significant bit if I0001 is on.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	DECO	X00201	D0101	D0001	D0102	

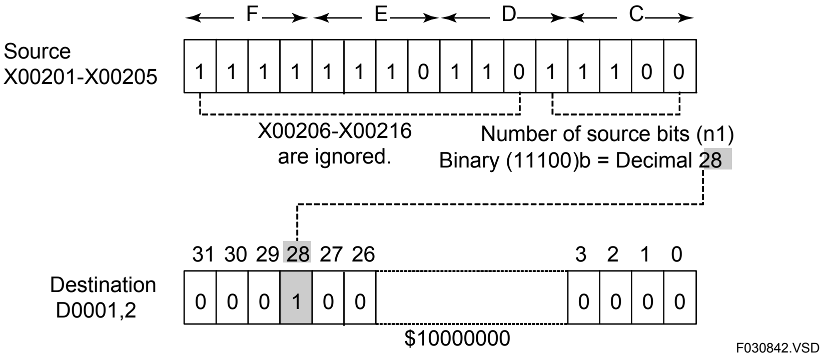
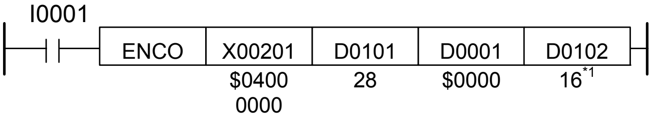


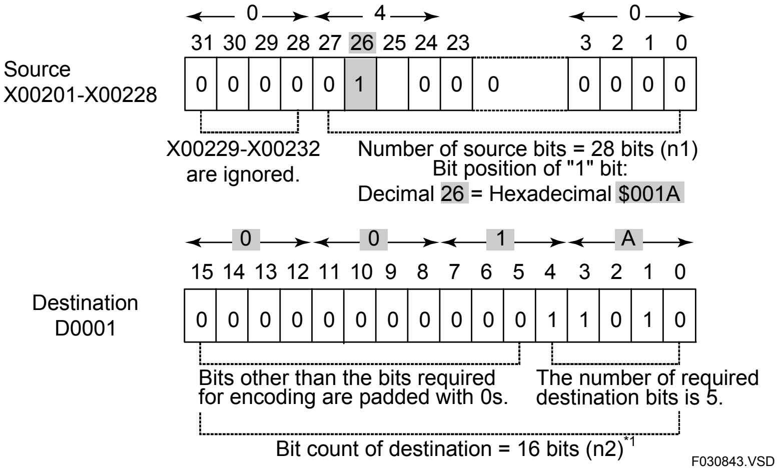
Figure 3.8.35 Example of a Decode Program

● Encoding

The sample code shown below encodes the lowest-order 28 bits (X00201 to X00228) of X00201 ($04000000) into 16 bits in D0001 starting at the least significant bit if I0001 is on.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	ENCO	X00201	D0101	D0001	D0102	

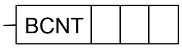
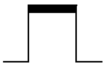
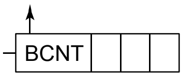
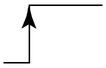
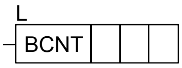
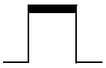
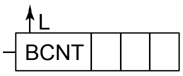
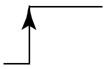


*1: Specify indirectly when setting the bit count of destination to a value greater than 9.

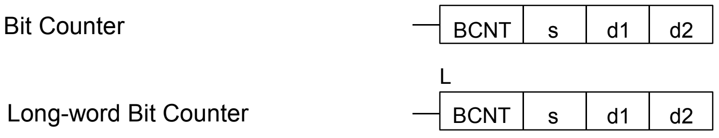
Figure 3.8.36 Example of an Encode Program

3.8.19 Bit Counter (BCNT), Long-word Bit Counter (BCNT L)

Table 3.8.38 Bit Counter, Long-word Bit Counter

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	157	Bit Counter	BCNT		✓	—		4	16 bits	—
	157P		↑BCNT					5		
	157L	Long-word Bit Counter	BCNT L		✓	—		4	32 bits	—
	157LP		↑BCNT L					5		

■ Parameter



F030844.VSD

s : Device number of the first device storing the data whose bits are to be counted
d1 : Device number of the first device for storing the number of "1" bits[*1]
d2 : Device number of the first device for storing the bit position of the least significant "1" bit[*1]

*1: d1 and d2 are handled as a word even in a 32-bit (long-word) instruction.

■ Available Devices

Table 3.8.39 Devices Available for the Bit Counter and Long-word Bit Counter Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
d1		✓	✓	✓*3	✓*3	✓*3	✓*1	✓*2	✓	✓*3	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes
d2		✓	✓	✓*3	✓*3	✓*3	✓*1	✓*2	✓	✓*3	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes

*1: Timer current value (may not be used for long-word high-speed read)
*2: Counter current value (may not be used for long-word high-speed read)
*3: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

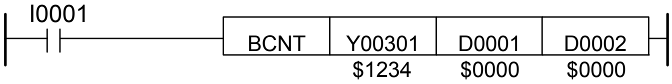
The Bit Counter and Long-word Bit Counter instructions count the number of "1" (ON) bits in 16- and 32-bit data, respectively, and place the number of "1" (ON) bits (bit count) in d1 and the bit position of the least significant "1" (ON) bit in d2.

The least significant bit position is identified by 0 and the most significant bit position by 15 or 31. The least significant bit position is set to -1 ($FFFF) if there is no "1" bit.

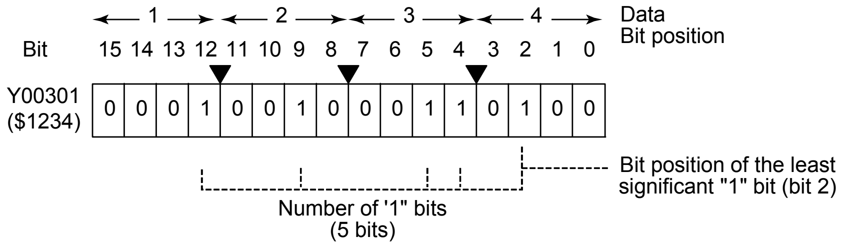
Use the Bit Counter instruction to count the number of bits in 16-bit data and the Long Bit Counter instruction to count the number of bits in 32-bit data.

■ Programming Example

The sample code shown below counts and loads the number of "1" bits in Y00301 ($1234) into D0001 and the bit position of the least significant "1" bit into D0002 if I0001 is on.



Line No.	Instruction	Operands				
0001	LD	I0001				
0002	BCNT	Y00301	D0001	D0002		



		Before Execution	After Execution
Device to be counted	Y00301	$1234	$1234
Number of "1" bits	D0001	$0000	$0005
Bit position of least significant "1" bit	D0002	$0000	$0002

The following figure shows when there is no "1" bit:

		Before Execution	After Execution
Device to be counted	Y00301	$0000	$0000
Number of "1" bits	D0001	$0000	$0000
Bit position of least significant "1" bit	D0002	$0000	$FFFF

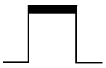
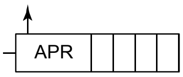
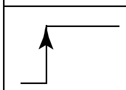
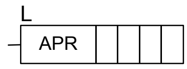
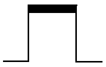
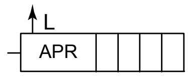
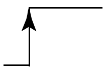
← -1 ($FFFF) is loaded.

F030845.VSD

Figure 3.8.37 Example of a Bit Counter Program

3.8.20 Approximate Broken Line (APR), Long-word Approximate Broken Line (APR L)

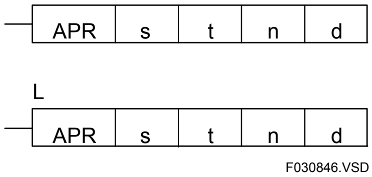
Table 3.8.40 Approximate Broken Line, Long-word Approximate Broken Line

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	158	Approximate Broken Line	APR		✓	—		5	16 bits	—
	158P		↑APR					6		
	158L	Long-word Approximate Broken Line	APR L		✓	—		5	32 bits	—
	158LP		↑APR L					6		

■ Parameter

Approximate Broken Line

Long-word Approximate Broken Line



s : Device number of the first device storing the data to be approximated
t : Device number of the first device storing the broken line data table
n : Number of broken data tables[*1]
d : Device number of the first device for storing the approximation result

*1: - n is handled as a word even in a 32-bit (long-word) instruction.
- The table count can be specified within the range of the number of devices available for a table.
- Assuming that the table is to start at D0001 and that there are data registers in up to D8192, the maximum value of n is 4095 (2047 for the Long-word Approximate Broken Line instruction).

■ Available Devices

Table 3.8.41 Devices Available for the Approximate Broken Line and Long-word Approximate Broken Line Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
t	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓		Yes	Yes
n	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*3	✓*3	✓*3	✓*1	✓*2	✓	✓*3	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes

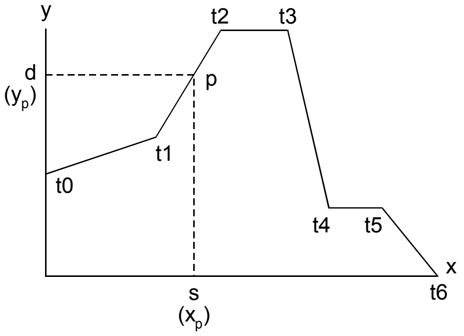
*1: Timer current value (may not be used as s, t, or d in the Long-word Approximate Broken Line instruction)
*2: Counter current value (may not be used as s, t, or d in the Long-word Approximate Broken Line instruction)
*3: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Approximate and Approximate Long-word instructions approximate 16- and 32-bit data, respectively, according to the given broken line table.

Use the Approximate Broken Line instruction to approximate 16-bit data and the Long-word Approximate Broken Line instruction to approximate 32-bit data.

The result of approximation is loaded as integers with an error of ±1.



	x	y
t0	0	10
t1	8	13
t2	14	20
t3	20	20
t4	23	6
t5	26	6
t6	28	0
	x_p	y_p
p	11	17

$$y = \frac{y_2-y_1}{x_2-x_1}(x-x_1)+y_1$$
$$= \frac{20-13}{14-8}(11-8)+13$$
$$= 16.5$$

F030848.VSD

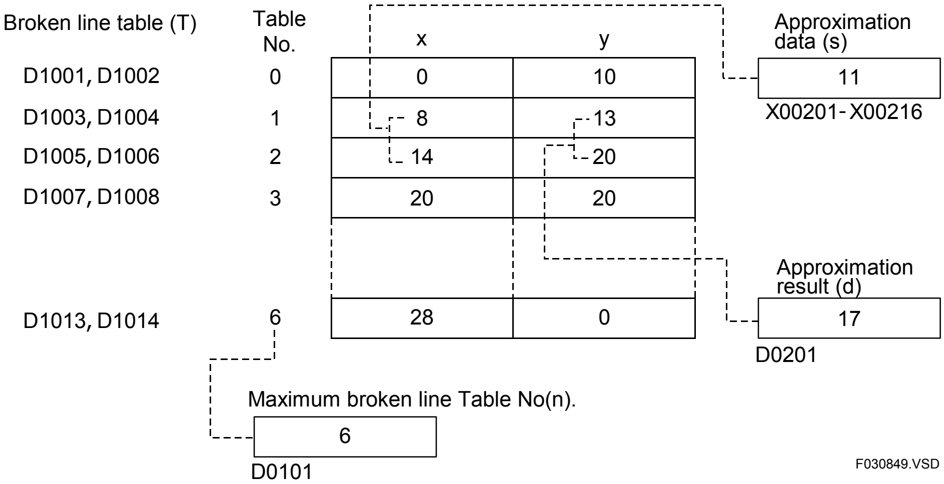
Figure 3.8.38 Example of a Broken Line Approximation

The instruction determines where s(x_p) falls in a range of values in the broken line table and calculates d(y_p) within that value range as a broken line approximation. In the above example, broken line approximation is carried out between t1(8, 13) and t2 (14, 20).

CAUTION

If the approximation data (s) falls within none of ranges of table X, an error is signaled and the special relay M201 is set to ON.

● Broken line table



Broken line table (T)	Table No.	x	y
D1001, D1002	0	0	10
D1003, D1004	1	8	13
D1005, D1006	2	14	20
D1007, D1008	3	20	20
D1013, D1014	6	28	0

Approximation data (s)
11
X00201- X00216

Approximation result (d)
17
D0201

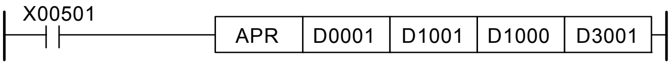
Maximum broken line Table No(n).
6
D0101

F030849.VSD

Figure 3.8.39 Broken Line Table (Approximate)

■ Programming Example

The sample code shown below approximates real number data in D0001 using the broken line table starting at D1001 with D1000 number of table entries, and stores the approximation result in D3001 to D3002 when X00501 is turned on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	APR	D0001	D1001	D1000	D3001	

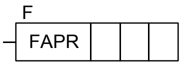
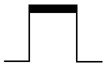
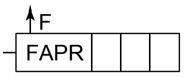
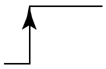
F030850.VSD

Figure 3.8.40 Example of a Floating-Point Broken Line Approximation Program

3.8.21 Float Approximate Broken Line (FAPR)

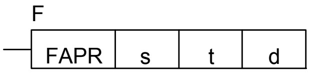
F3SP25 F3SP35	F3SP22 F3SP28 F3SP38	F3SP53 F3SP58 F3SP59	F3SP66 F3SP67	F3SP71 F3SP76

Table 3.8.42 Float Approximate Broken Line

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	918	Float Approximate Broken Line	FAPR	 F FAPR	✓	—		5	32 bits	—
	918P		↑FAPR	 ↑F FAPR				6		

■ Parameter

Float Approximate Broken Line



F030851.VSD

s : Data to be subject to floating-point approximation
or device number of the first device storing the data subject to floating-point approximation

t : Device number of the first device storing the floating-point broken line data table
(the first word contains the number of tables.)

d : Device number of the first device for storing the approximation result

Tables s, d, and t must be represented in the IEEE single-precision floating-point format (32-bit).
The table count can be specified within the range of the number of devices available for a table.
Assuming that the table is to start at D0001 and that there are data registers in up to D8192,
the maximum value of the number of tables is 2047.

■ Available Devices

Table 3.8.43 Devices Available for the Float Approximate Broken Line Instruction

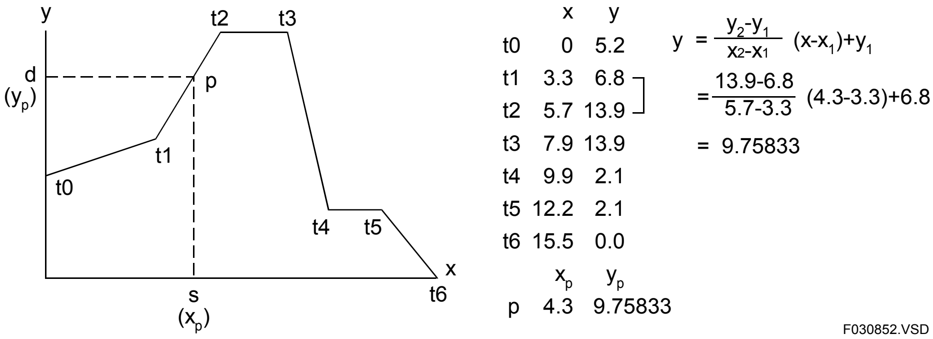
Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
t	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓		Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Float Approximate Broken Line instruction approximates single-precision floating-point data according to the given broken line table.

The result of approximation is loaded as single-precision floating point numbers. The single-precision floating-point numbers are represented in the IEEE format.



	x	y
t0	0	5.2
t1	3.3	6.8
t2	5.7	13.9
t3	7.9	13.9
t4	9.9	2.1
t5	12.2	2.1
t6	15.5	0.0
	x_p	y_p
p	4.3	9.75833

$$y = \frac{y_2-y_1}{x_2-x_1}(x-x_1)+y_1$$

$$= \frac{13.9-6.8}{5.7-3.3}(4.3-3.3)+6.8$$

$$= 9.75833$$

Figure 3.8.41 Example of a Broken Line Approximation

The instruction determines where s (x_p) falls in a range of values in the broken line table and calculates d (y_p) within that value range as a broken line approximation. In the above example, broken line approximation is carried out between t1 (3.3, 6.8) and t2 (5.7, 13.9).

CAUTION

If the approximation data (s) falls within none of ranges of table X, an error is signaled and the special relay M201 is set to ON.

● Broken line table

In a floating-point broken line approximation, the first word of the broken line table must be loaded with the number of tables (maximum table number) and the second and subsequent words with broken line data.

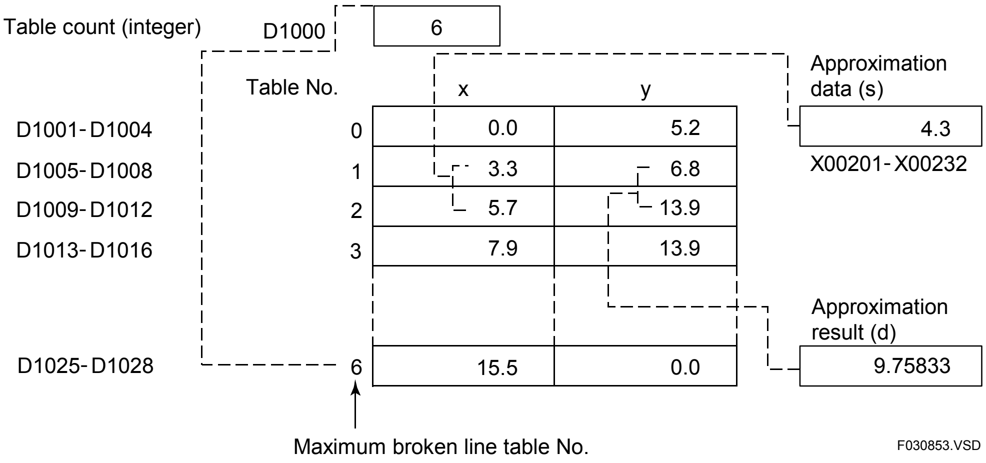


	Table No.	x	y
D1001-D1004	0	0.0	5.2
D1005-D1008	1	3.3	6.8
D1009-D1012	2	5.7	13.9
D1013-D1016	3	7.9	13.9
D1025-D1028	6	15.5	0.0

Figure 3.8.42 Broken Line Table (Float Approximate Broken Line)

In a broken line table, a pair of coordinates (x, y) is represented by two long words. In a pair of coordinates (x, y), specify x and y such that x is always assigned to a smaller device number and y to a larger device number.

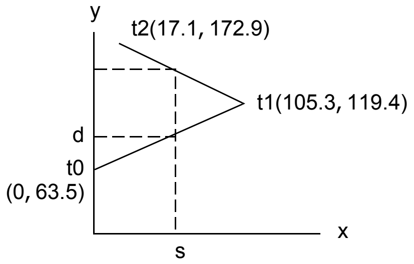
Floating point	x	y
✓	D0001,2	D0003, 4
X	D0003,4	D0001, 2

F030854.VSD

Figure 3.8.43 Broken-line Table (a pair of coordinates)

x's in the tables must be set in the increasing order of magnitude (monotonically increasing). When x's are specified in the increasing order, broken line approximation is carried out within the first value range of s(x_p) that is encountered (table with the smaller table No.).

	Table No.	x	y
D1021- D1024	0	0.0	63.5
D1025- D1028	1	105.3	119.4
D1029- D1032	2	17.1	172.9



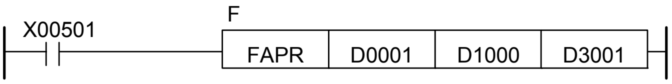
Approximation data X00201- X00232	55.9	Approximation is carried out between t0 and t1 which is given a smaller table number though s with a value of 55.9 falls between t0 and t1 and between t1 and t2.
Approximation result D0201- D0202	93.1753	The result of approximation done using a value range (between t0 and t1) with a smaller table number is loaded.

F030855.VSD

Figure 3.8.44 Broken-line Table (approximation)

■ Programming Example

The sample code shown below determines in which range of the broken line table D1001 (table count stored in D1000) the real (IEEE single-precision floating point) approximation data in D0001 falls, and places the approximation result in the location from D3001 to D3002 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FAPR	D0001	D1000	D3001		

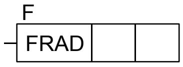
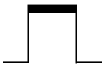
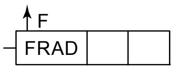
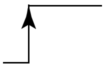
F030856.VSD

Figure 3.8.45 Example of a Floating-point Approximation Program

3.8.22 Convert Degree to Radian (FRAD)

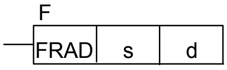
F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.8.44 Convert Degree to Radian

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	905	Convert Degree to Radian	FRAD		✓	—		4	32 bits	—
	905P		↑FRAD					5		

■ Parameter

Convert Degree to Radian



F030857.VSD

s : Angle data (in degrees) or device number of the first device storing the angle data to be converted (source)

d : Device number of the first device for storing the angle data converted to radians (destination)

Both the source (s) and destination (d) must be represented in the IEEE single-precision floating-point format (32 bits).

■ Available Devices

Table 3.8.45 Devices Available for the Convert Degree to Radian Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Convert Degree to Radian instruction converts angle data (IEEE single-precision floating point) represented in degrees to angle data (IEEE single-precision floating point) in radians. The equation is shown below.

$d = s \times \pi / 180$

s : Source (in degrees)

d : Destination (in radians)

The single-precision floating-point numbers are represented in the IEEE format.

● Example

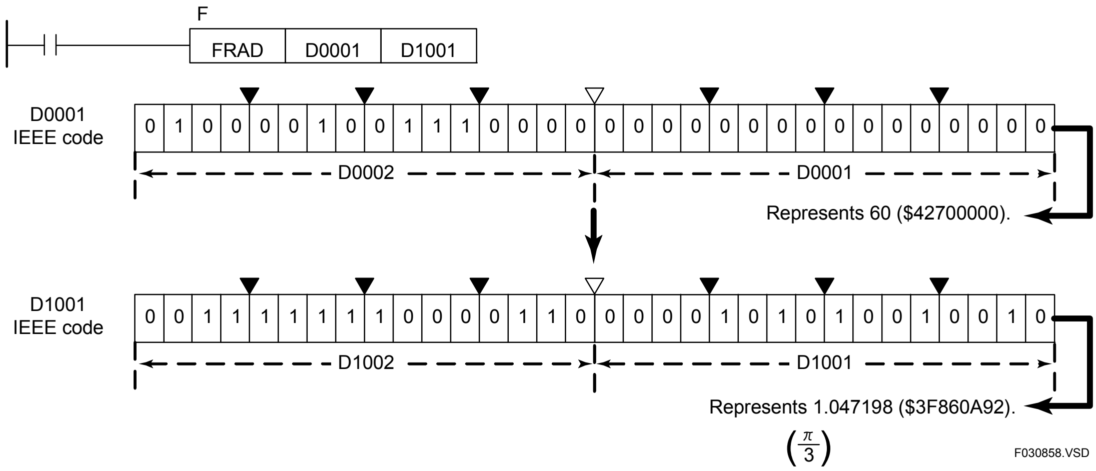
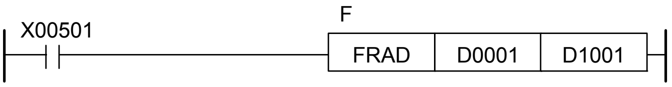


Figure 3.8.46 Example of Degree-to-radian Conversion

■ Programming Example

The sample code shown below converts angle data (in degrees) in the location from D0001 to D0002 to radian data and places the result in the location from D1001 to D1002 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FRAD	D0001	D1001			

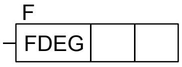
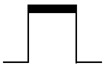
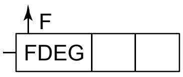
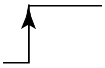
F030859.VSD

Figure 3.8.47 Example of a Degree-to-radian Conversion Program

3.8.23 Convert Radian to Degree (FDEG)

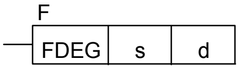
F3SP25 F3SP35	F3SP22 F3SP28 F3SP38	F3SP53 F3SP58 F3SP59	F3SP66 F3SP67	F3SP71 F3SP76

Table 3.8.46 Convert Radian to Degree

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	906	Convert Radian to Degree	FDEG		✓	—		4	32 bits	—
	906P		↑FDEG					5		

■ Parameter

Convert Radian to Degree



F030860.VSD

s : Angle data (in radians) or device number of the first device storing the angle data to be converted (source)

d : Device number of the first device for storing the angle data converted to degrees (destination)

Both the source (s) and destination (d) must be represented in the IEEE single-precision floating-point format (32 bits).

■ Available Devices

Table 3.8.47 Devices Available for the Convert Radian to Degree Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1 : See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Convert Radian to Degree instruction converts angle data (IEEE single-precision floating point) represented in radians to angle data (IEEE single-precision floating point) in degrees. The equation is shown below.

d = s x 180/π

s : Source (in radians)

d : Destination (in degrees)

The single-precision floating-point numbers are represented in the IEEE format.

● Example

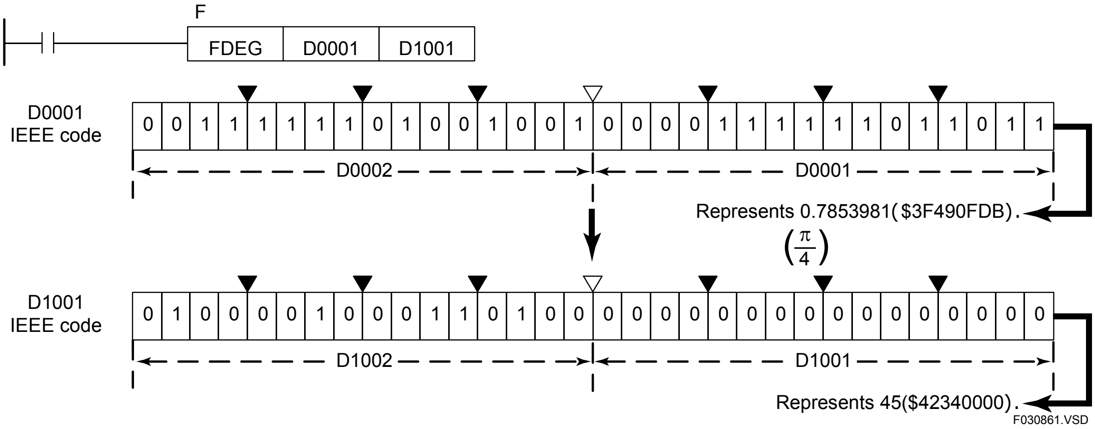
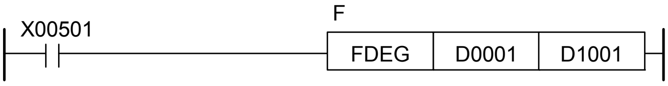


Figure 3.8.48 Example of Radian-to-degree Conversion

■ Programming Example

The sample code shown below converts angle data (in radians) in the location from D0001 to D0002 to degree data and places the result in the location from D1001 to D1002 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	FDEG	D0001	D1001			

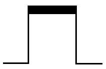
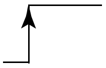
F030862.VSD

Figure 3.8.49 Example of a Radian-to-degree Conversion Program

3.8.24 Extend Sign (SIGN L)

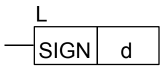
F3SP25
F3SP35 | F3SP22
F3SP28
F3SP38 | F3SP53
F3SP58
F3SP59 | F3SP66
F3SP67 | F3SP71
F3SP76

Table 3.8.48 Extend Sign

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	951	Extend Sign	SIGN L	L SIGN	✓	—		3	32 bits	—
	951P		↑SIGN L	↑L SIGN				4		

■ Parameter

Extend Sign



F030863.VSD

d : Device number of the device storing the data that is to be sign-extended from 1 word to 1 long word

■ Available Devices

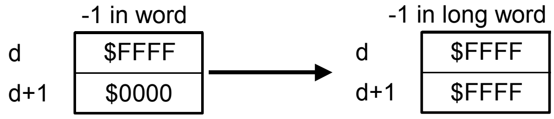
Table 3.8.49 Devices Available for the Extend Sign Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Extend Sign instruction sign-extends 1-word data in d to a 1-long-word data and places the result in d and d+1.



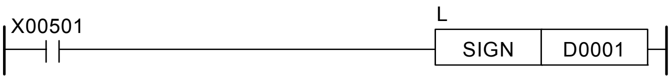
F030864.VSD

Figure 3.8.50 Sign Extension

d+1 may contain any initial value. After the instruction is executed, d+1 is loaded with $0000 if the most significant bit of d is 0 (+) and with $FFFF if the most significant bit is 1 (–).

■ Programming Example

The sample code shown below sign-extends the word data in location D0001 to a long-word data and places the result in the location from D0001 to D0002 if X00501 is ON.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	SIGN L	D0001				

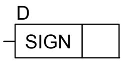
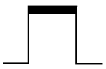
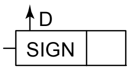
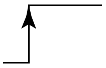
F030865.VSD

Figure 3.8.51 Example of an Extend Sign Program

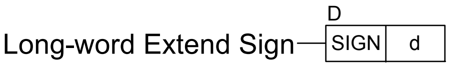
3.8.25 Long-word Extend Sign (SIGN D)

F3SP71
F3SP76

Table 3.8.50 Long-word Extend Sign

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	951D	Long-word Extend Sign	SIGN D		✓	—		3	64 bits	—
	951DP		↑SIGN D					4		

■ Parameter



F3825001.VSD

d : Device number of the device storing the data that is to be long-word sign-extended from 1 long-word to 1 double long-word

■ Available Devices

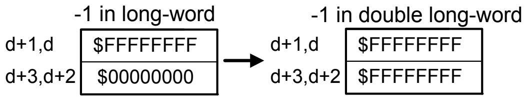
Table 3.8.51 Devices Available for the Long-word Extend Sign Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d									✓	✓	✓*1	✓*1		✓*1			Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Long-word Extend Sign instruction sign-extends 1 long-word data in d, d+1 to a 1 double long-word data and places the result in d, d+1, d+2 and d+3.



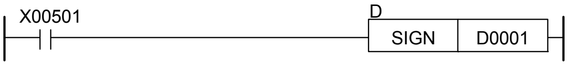
F3825002.VSD

Figure 3.8.52 Double Long-word Sign Extension

d+3 and d+2 may contain any initial value. After the instruction is executed, d+3 and d+2 are loaded with $0000 if the most significant bit of d+1 and d is 0 (+) and with $FFFFFFFF if the most significant bit is 1 (–).

■ Programming Example

The sample code shown below sign-extends the long-word data in location from D0001 to D0002 to a double long-word data and places the result in the location from D0001, D0002, D0003, and D0004 if X00501 is ON.



Line No.	Instruction	Operands			
0001	LD	X00501			
0002	SIGN D	D0001			

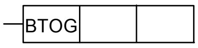
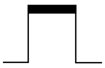
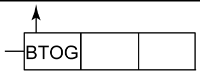
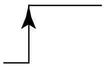
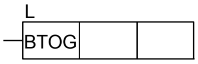
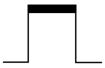
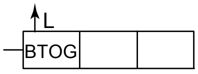
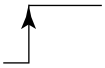
F3825003.VSD

Figure 3.8.53 Example of a Double Long-word Extend Sign Program

3.8.26 Binary to Gray-code (BTOG), Long-word Binary to Gray-code (BTOG L)

F3SP71-4S
F3SP76-7S

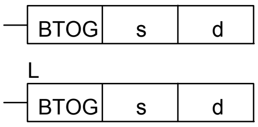
Table 3.8.51 Binary to Gray-code, Long-word Binary to Gray-code

Classi-fication	FUN No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	58	Binary to Gray-code	BTOG		✓	—		4	16 bits	—
	58P		↑BTOG					5		
	58L	Long-word Binary to Gray-code	BTOG L		✓	—		4	32 bits	—
	58LP		↑BTOG L					5		

■ Parameter

Binary to Gray-code

Long-word Binary to Gray-code



T3826005.VSD

s : Device number of the first device storing data to be subject to binary-to-gray-code conversion and long-word-binary-to-gray-code conversion
d : Device number of the first device storing the converted data

■ Available Devices

Table 3.8.52 Devices Available for the Binary to Gray-code and Long-word Binary to Gray-code Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s									✓	✓*1	✓*1	✓*1		✓*1		✓	Yes	Yes
d									✓	✓*1	✓*1	✓		✓			Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

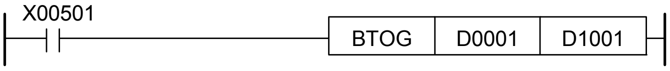
The Binary to Gray-code and Long-word Binary to Gray-code instructions convert 16- and 32-bit binary code (respectively) to gray code, and store the converted data in the specified devices.

Use the Binary to Gray-code instruction to convert 16-bit data and the Long-word Binary to Gray-code instruction to convert 32-bit data.

If the data s to be converted is negative, the special relay M201 is set to ON as an instruction processing error (data error) and the instruction will not be executed.

■ Programming Example

The sample code shown below converts the binary values in location D0001 to gray code and stores the result in the location D1001 if X00501 is ON.



Line No.	Instruction	Operands			
0001	LD	X00501			
0002	BTOG	D0001	D1001		

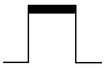
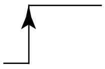
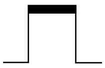
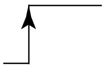
F3826006.VSD

Figure 3.8.54 Example of a Binary-to-Gray-code Conversion Program

3.8.27 Gray-code to Binary (GTOB), Long-word Gray-code to Binary (GTOB L)

F3SP71-4S
F3SP76-7S

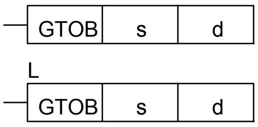
Table 3.8.53 Gray-code to Binary, Long-word Gray-code to Binary

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	59	Gray-code to Binary	GTOB	GTOB	✓	–		4	16 bits	–
	59P		↑GTOB	↑ GTOB				5		
	59L	Long-word Gray-code to Binary	GTOB L	L GTOB	✓	–		4	32 bits	–
	59LP		↑GTOB L	↑L GTOB				5		

■ Parameter

Gray-code to Binary

Long-word Gray-code to Binary



T3827005.VSD

s : Device number of the first device storing data to be subject to gray-code-to-binary conversion and long-word-gray-code-to-binary conversion

d : Device number of the first device storing the converted data

■ Available Devices

Table 3.4.5 Devices Available for the Gray-code to Binary and Long-word Gray-code to Binary Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s									✓	✓*1	✓*1	✓*1		✓*1		✓	Yes	Yes
d									✓	✓*1	✓*1	✓		✓			Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

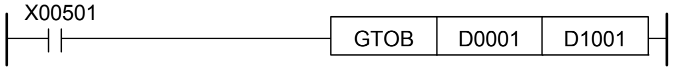
The Gray-code to Binary and Long-word Gray-code to Binary instructions convert 16- and 32-bit gray code (respectively) to binary code, and store the converted data in the specified devices.

Use the Gray-code to Binary instruction to convert 16-bit data and the Long-word Gray-code to Binary instruction to convert 32-bit data.

If the data converted with gray-code-to-binary conversion is out of the range between 0 and 32767 or the data converted with long-word-gray-code-to-binary conversion is out of the range between 0 and 2147483647, the special relay M201 is set to ON as an instruction processing error (data error) and the instruction will not be executed.

■ Programming Example

The sample code shown below converts the gray code values in location D0001 to binary and stores the result in the location D1001 if X00501 is ON.



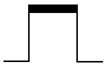
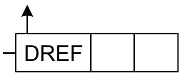
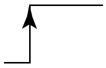
Line No.	Instruction	Operands			
0001	LD	X00501			
0002	GTOB	D0001	D1001		

F3827006.VSD

Figure 3.8.55 Example of a Gray-code-to-Binary Conversion Program

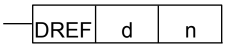
3.9 Direct Refresh Instruction (DREF)

Table 3.9.1 Direct Refresh Instruction

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	60	Direct Refresh	DREF		✓	—		3	—	—
	60P		↑DREF					4		

■ Parameter

Direct Refresh



F030901.VSD

d : Device number of the first device storing the data to be refreshed
n : Number of bits to be refreshed

■ Available Devices

Table 3.9.2 Devices Available for the Direct Refresh Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d	✓	✓															Yes	No
n	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1 : See Section 1.17, "Devices Available As Instruction Parameters."
*2 : Timer current value
*3 : Counter current value

■ Function

If the value of parameter d is an input relay (X), the Direct Refresh instruction immediately inputs the specified number of bits in the middle of a scan.

If the value of parameter d is an output relay (Y), The Direct Refresh instruction immediately outputs the specified number of bits, as well as the channel containing these bits, in the middle of a scan. The channels here refer to groups of sixteen output relays, namely, Y□01 to Y□16, Y□17 to Y□32, Y□33 to Y□48, Y□49 to Y□64. For instance, if parameter d is Y615 and parameter n is 4, the relays specified for output are Y615, Y616, Y617 and Y618, but 32 bits of Y601 to Y632 are actually sent to output immediately.

Since normal input/output is executed all at once at the end of a program, you cannot input or output data from or to the external world in the middle of a scan. The Direct Refresh instruction is used to immediately read or write data in the middle of a scan.

The scope of the DREF instruction is limited to a single module. Consider an example where a 32-point output module is installed in slot No. 3. For this module, you can execute

DREF Y00301 32

but you cannot execute

DREF Y00317 32.

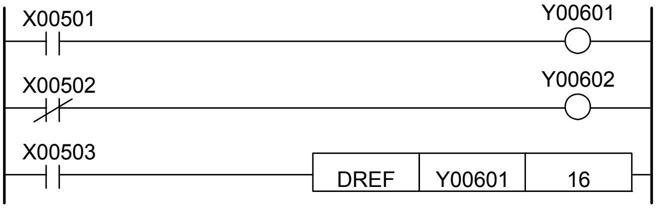
Since Y00301 to Y00332 belong to the same module, you can refresh the 32 points starting at Y00301 but you cannot refresh the 32 points that start at Y00317 as those 32 points go beyond Y00332.

SEE ALSO

- For details on the output modules, see Section 2.2.2 of "Sequence CPU Instruction Manual – Functions (for F3SP22-0N, F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S)" (IM 34M06P13-01E), Section A2.2.2 of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E), or Section A2.2.2 of "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).
- For details on input/output refreshing, see Section 3.4 of "Sequence CPU Instruction Manual – Functions (for F3SP22-0N, F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S)" (IM 34M06P13-01E), Section A3.4 of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E), or Section A3.4 of "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

■ Programming Example

The sample code shown below refreshes 16 bits of data in the location from Y00601 to Y00616 if X00503 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	OUT	Y00601				
0003	LDN	X00502				
0004	OUT	Y00602				
0005	LD	X00503				
0006	DREF	Y00601	16			

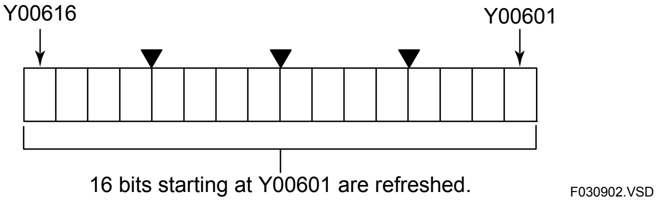
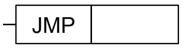
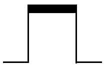
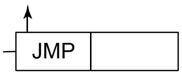
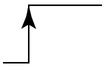


Figure 3.9.1 Example of a Direct Refresh Program

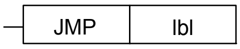
3.10 Program Control Instructions

3.10.1 Jump (JMP)

Table 3.10.1 Jump

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	61	Jump	JMP		✓	—		1	—	—
	61P		↑JMP					2		

■ Parameter

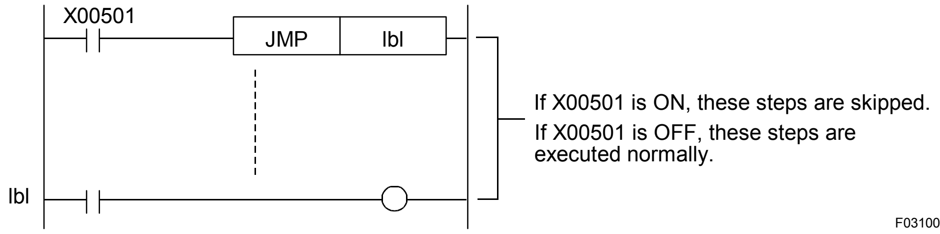
Jump 

F031001.VSD

lbl : Label of the destination to which a jump is to be made
lbl must be a 1- to 6-character alphanumeric string beginning with a letter.

■ Function

The Jump instruction transfers control to the line identified by the given label.


X00501
JMP
lbl
lbl
If X00501 is ON, these steps are skipped.
If X00501 is OFF, these steps are executed normally.

F031002.VSD

Figure 3.10.1 Example of a Jump Operation

An error will be generated if one of the following conditions occurs while you are coding a program using WideField3, WideField2, WideField, or Ladder Diagram Support Program M3:

- A location in a different block is specified as the destination.
  (The destination of a jump must be within the same block.)
- Two or more labels of the same name are specified.
- The label specified in the Jump instruction is not found.

TIP

The label name indicates where control must be transferred when the JMP or CALL instruction is executed. It is a 1- to 6-character alphanumeric string beginning with a letter.

CAUTION

When the program causes a jump into a subroutine, the scan ends with a Subroutine Return (RET) instruction specified in that subroutine. In such a case, an error is signaled and the special relay (subroutine error) is set to ON.

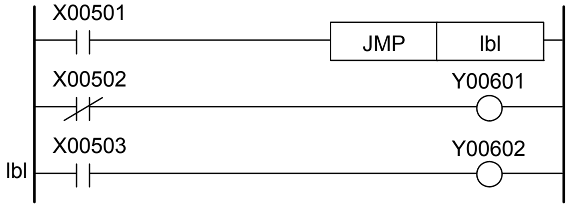
A scan timeout error is generated if an infinite loop is entered as the result of a jump and thus the scan monitoring time is exceeded.

TIP

Scan timeout is a ladder sequence operation error where the actual scan time exceeded the preset scan monitoring time.

■ Programming Example

The sample code shown below causes a jump to the step that is labeled lbl if X00501 is on.



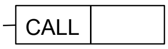
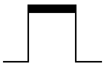
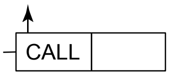
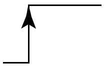
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	JMP	lbl				
0003	LDN	X00502				
0004	OUT	Y00601				
lbl	LD	X00503				
0006	OUT	Y00602				

F031003.VSD

Figure 3.10.2 Example of a Jump Program

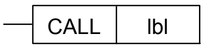
3.10.2 Subroutine Call (CALL), Subroutine Entry (SUB), Subroutine Return (RET)

Table 3.10.2 Subroutine Call, Subroutine Entry, Subroutine Return

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	62	Subroutine Call	CALL		✓	—		1	—	—
	62P		↑CALL					2		
	63	Subroutine Entry	SUB		—	✓	—	1	—	—
	64	Subroutine Return	RET		—	✓	—	1	—	—

■ Parameter

Subroutine Call

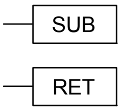


F031004.VSD

lbl : Label of the subroutine to be called
lbl must be a 1- to 6-character alphanumeric string beginning with an alphabetic character.

Subroutine Entry

Subroutine Return



F031005.VSD

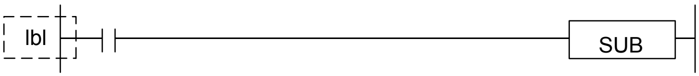
■ Function

(1) Subroutine Call

The Subroutine Call instruction transfers control to the subroutine identified by the given label. When the execution of the specified subroutine ends, control transfers to the step immediately following the step that called the subroutine.

(2) Subroutine Entry

The Subroutine Entry identifies the beginning of a subroutine. A subroutine entry always requires a label.



F031006.VSD

Figure 3.10.3 Label of a Subroutine Entry

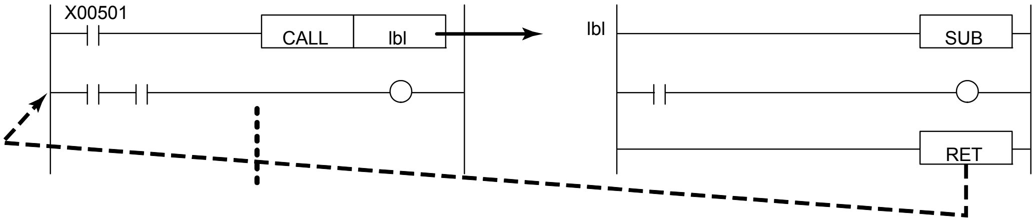
(3) Subroutine Return

The Subroutine Return instruction identifies the end of a subroutine. A subroutine requires at least one Subroutine Return instruction.

TIP

A subroutine is a process invoked by a main routine using a CALL instruction. It may be stored at any location in any block. When the CPU is configured to execute only specified blocks, a subroutine in a block which is not selected for execution may also be executed if indirectly invoked by an executing block.

An example of a subroutine call is shown below.



F031007.VSD

Figure 3.10.4 Example of a Subroutine Call

If X00501 is on, execution transfers to the subroutine that begins with the step with the label lbl. When execution reaches the end of the subroutine (Subroutine Return), the program transfers control to the step immediately following the subroutine call. If X00501 is off, the subroutine that begins with the step with the label lbl is not executed.

Subroutines may appear anywhere in a program. A subroutine is a set of instructions that are executed only when it is invoked with a Subroutine Call instruction (CALL). It is not executed during the normal scans. A subroutine may appear in the same block as the Subroutine Call (CALL) instruction calling it or in a different block.

If a subroutine is located in a different block and the CPU is configured to execute specified blocks only, the block containing the subroutine may be active or inactive. The subroutine can be executed even if its block is inactive.

A program can contain any number of subroutine call (CALL) instructions.

An error will be generated if any one of the following conditions occurs while you are coding a program using WideField3, WideField2, WideField, or Ladder Diagram Support Program M3:

- Two or more labels of the same name are specified.
- The label specified in the Subroutine Call instruction (CALL) is not found.
- The Subroutine Entry (SUB) and Subroutine Return (RET) instructions are specified so that subroutines are nested.
- There are two or more Subroutine Return (RET) instructions in a subroutine.

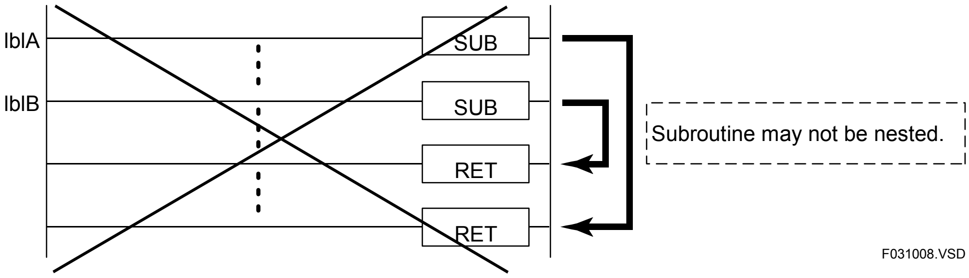


Figure 3.10.5 Nesting Subroutine Entries are Inhibited

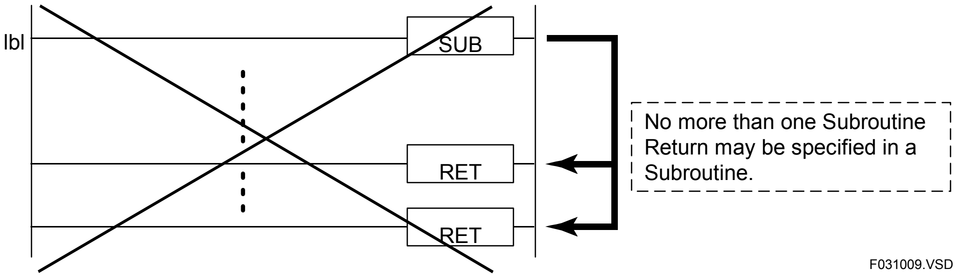

lbl
SUB
RET
RET
No more than one Subroutine Return may be specified in a Subroutine.
F031009.VSD

Figure 3.10.6 Using Two or More Subroutine Return Instructions is Disallowed

An error will be generated and the special relay M201 (instruction processing error) will be set to ON in the following cases:

- A Subroutine Return (RET) instruction is executed before its matching Subroutine Call (CALL) instruction is executed.
- The nesting depth of subroutine calls exceeds 8.

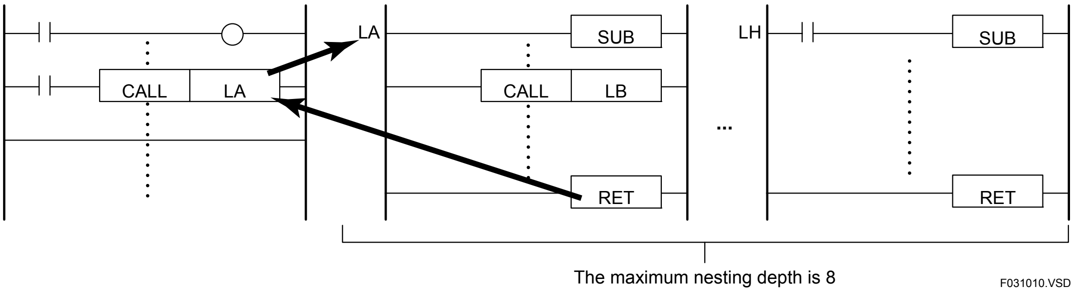


Figure 3.10.7 Subroutine Nesting

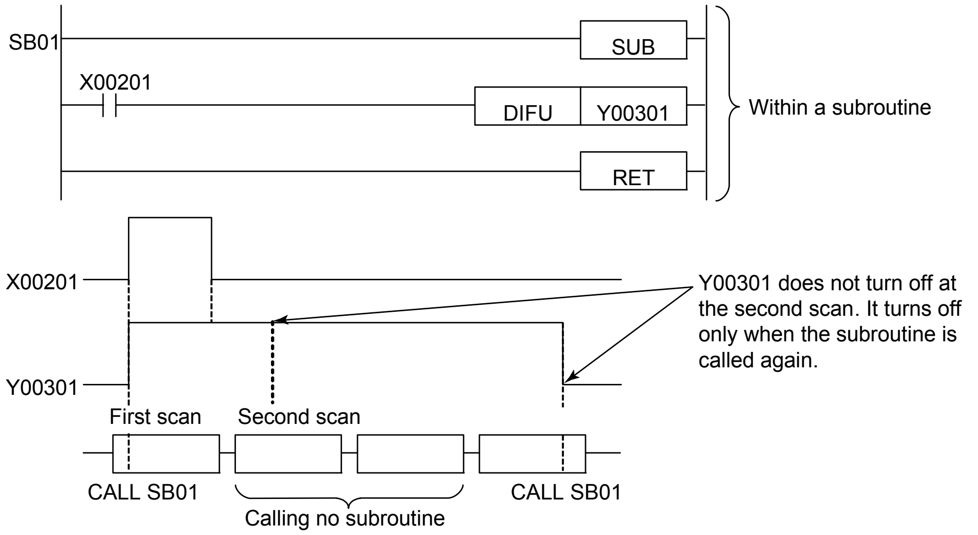
CAUTION

An instruction processing error is generated if the nesting depth of subroutine calls exceeds 8. Make sure that the nesting depth of subroutine calls in your program does not exceed 8.

Care must be taken with the following when using differential type instructions in a subroutine (code between SUB and RET instructions).

CAUTION

- DIFU and DIFD instructions
  These instructions turn on their output on the rising and falling edges of their input condition, respectively. In this case, the output will not turn off until the same subroutine is called next time and DIFU and DIFD are executed.
- Differential up application instruction
  A differential up application instruction will not be executed when it is called after its input condition is switched from OFF to ON state during a scan period during which the subroutine is not called.
- LDU/LDD/UP/DWN/UPX/DWNX instructions
  The result of operation does not equal ON at the next subroutine call even if the input condition or specified device (LDU/LDD instructions) makes an OFF-to-ON or ON-to-OFF transition during a scan in which the subroutine was not called.



F031011.VSD

Figure 3.10.8 Precaution When Using a Differential Type Instruction in a Subroutine (1)

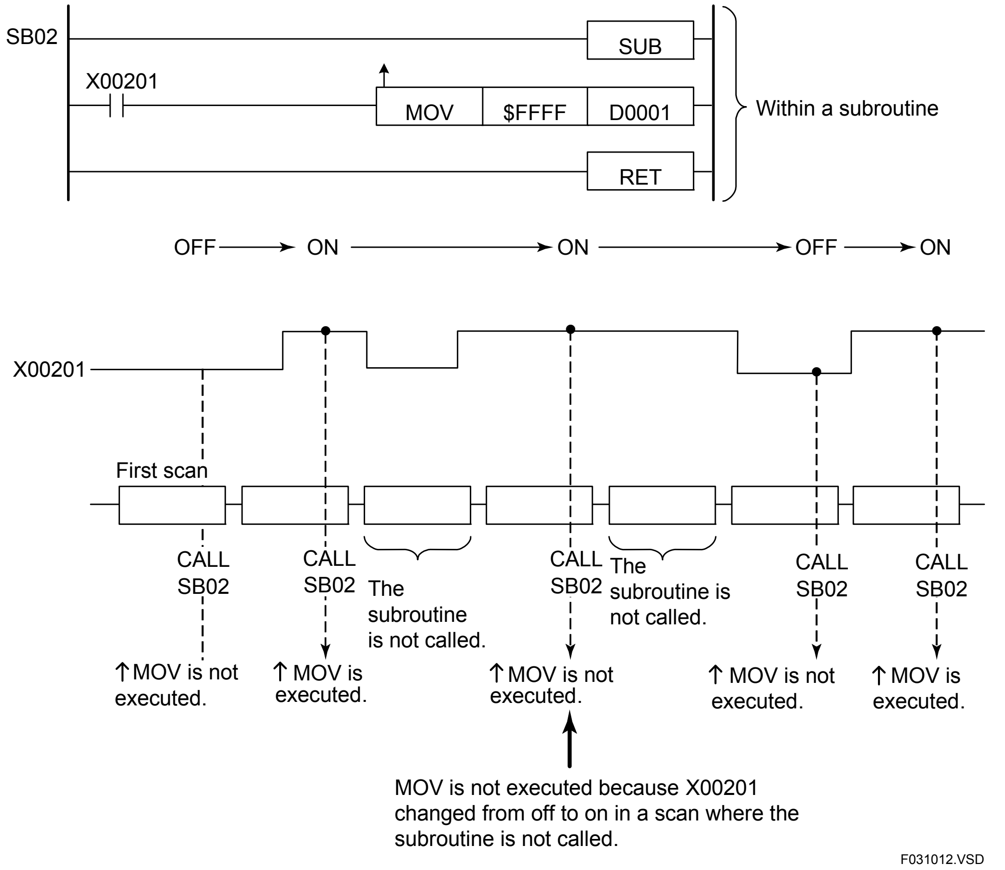
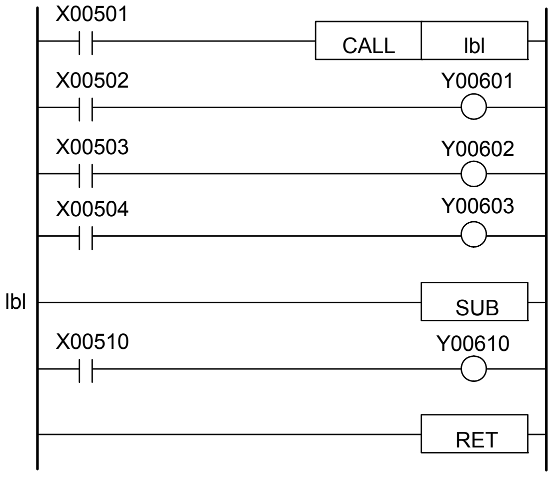


Figure 3.10.9 Precaution When Using a Differential Type Instruction in a Subroutine (2)

■ Programming Example

The sample code shown below transfers control to the step that is labeled lbl if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	CALL	lbl				
0003	LD	X00502				
0004	OUT	Y00601				
0005	LD	X00503				
0006	DREF	Y00602				
0007	LD	X00504				
0008	OUT	Y00603				
0009	SUB					
0010	LD	X00510				
0011	OUT	Y00610				
0012	RET					

F031013.VSD

Figure 3.10.10 Example of a Subroutine Program

3.10.3 Interrupt (INTP), Interrupt Return (IRET)

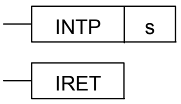
Table 3.10.3 Interrupt, Interrupt Return

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	65	Interrupt	INTP	INTP	—	✓	—	1	—	—
	66	Interrupt Return	IRET	IRET	—	✓	—	1	—	—

■ Parameter

Interrupt

Interrupt Return



F031014.VSD

s : Input relay causing an interrupt

■ Available Devices

Table 3.10.4 Devices Available for Interrupt and Interrupt Return Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓																No	No

■ Function

The Interrupt and Interrupt Return instructions identify the interrupt program that is executed on the rising edge of an input interrupt generated by an input (input/output) module. Note that, although contacts of almost all input (input/output) modules can be used as input interrupts, some input (input/output) modules have no interrupt capability.

Up to 4 input interrupts can be registered.

The input/output processing at input interrupt time is summarized in the following table.

Table 3.10.5 Interrupt-time Input/output Processing

Input / Output	Processing
Input	The result of preceding input/output refreshing is used
Output	Refreshed in the next input/output refresh.

Note: Use the Direct Refresh instruction to generate output prior to the next input/output refresh.

CAUTION

When an input relay causing an interrupt is used in an input interrupt subroutine (between INTP and IRET), the input relay may not be ON in the input interrupt subroutine. This is because the input interrupt program may be executed during the time after the rising edge of the input relay and before execution of input refresh.

Control returns to the step immediately following the step that caused the input interrupt when the specified input interrupt processing ends.

Any input interrupt requests generated during processing of the current input interrupt are executed after the current input interrupt processing is finished (interrupt pending). The maximum number of input interrupt requests that can be held pending at a time is 7, excluding the currently executing input interrupt request.

CAUTION

If the number of pending input interrupt requests exceeds 7, the extra requests are ignored and the special relay M201 (instruction processing error) is set to ON.

Any input interrupts that are generated in the stopped and paused states are all ignored.

An example of input interrupt processing is shown below.

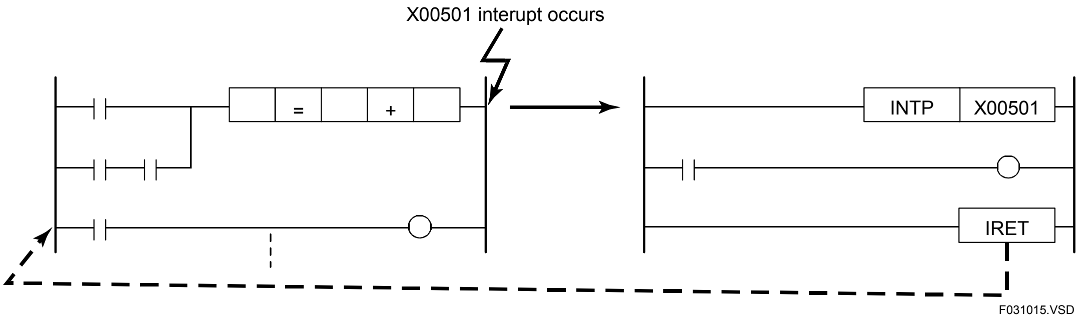


Figure 3.10.11 Outline of Interrupt Processing

If X00501 is on, execution transfers to the beginning of the corresponding interrupt processing routine (INTP). The input interrupt is held pending if an application instruction is being executed. After the execution of the current application instruction ends, control transfers to the input interrupt processing routine.

When execution reaches the end of the input interrupt processing (IRET) routine, control is returned to the step immediately following the step that caused the input interrupt.

Input interrupt processing routines are not executed if X00501 is off.

An input interrupt processing routine may appear anywhere in a ladder sequence program.

An input interrupt processing routine is a set of instructions that are executed only when an input interrupt occurs; it is not executed during normal scans.

If an input interrupt processing routine is located in a block other than the current block and the CPU is configured to execute specified blocks only, the block containing the input interrupt processing routine may be active or inactive.

An input interrupt processing routine can be executed even if its block is inactive.

An error will be generated if one of the following conditions occurs while you are coding a program using WideField3, WideField2, WideField or Ladder Diagram Support Program M3:

- The Interrupt (INTP) and Interrupt Return (IRET) instructions are specified so that their subroutines are nested.

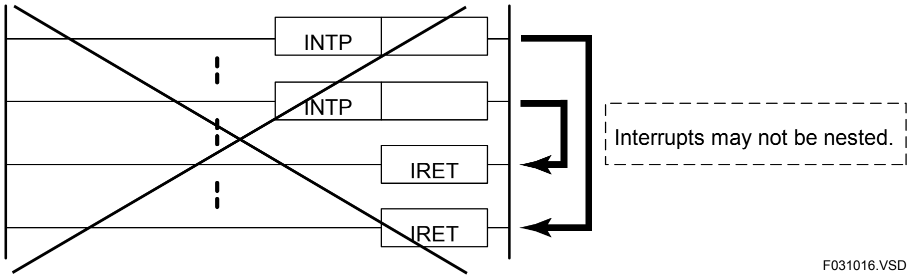


Figure 3.10.12 Nesting of Interrupt Processing Routines is Disallowed

- There are two or more Interrupt Return (IRET) instructions in an interrupt processing routine.

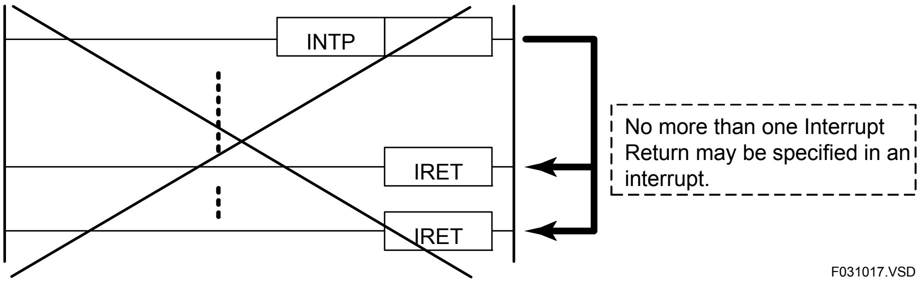

INTP
IRET
IRET
No more than one Interrupt Return may be specified in an interrupt.
F031017.VSD

Figure 3.10.13 Using Two or More Interrupt Return Instructions is Prohibited

CAUTION

An error will be generated in the following case:

An Interrupt Return (IRET) instruction is executed before an Interrupt (INTP) instruction is executed. In this case, the special relay M201 (instruction processing error) will be set to ON, and the steps following the Interrupt Return (IRET) instruction to the last step are not executed.

While an input interrupt is being processed, if the same input interrupt is detected again, it will be held pending.

Input interrupt processing routines are controlled by the Disable Interrupt and Enable Interrupt instructions and are enabled by default.

Care must be taken with the following when using differential type instructions in an input interrupt processing routine (code between INTP and IRET instructions).

CAUTION

- DIFU and DIFD instructions
  These instructions turn on their output on the rising and falling edges of their input condition, respectively. In this case, the output will not turn off until the same input interrupt processing routine is activated next time and DIFU and DIFD are executed.

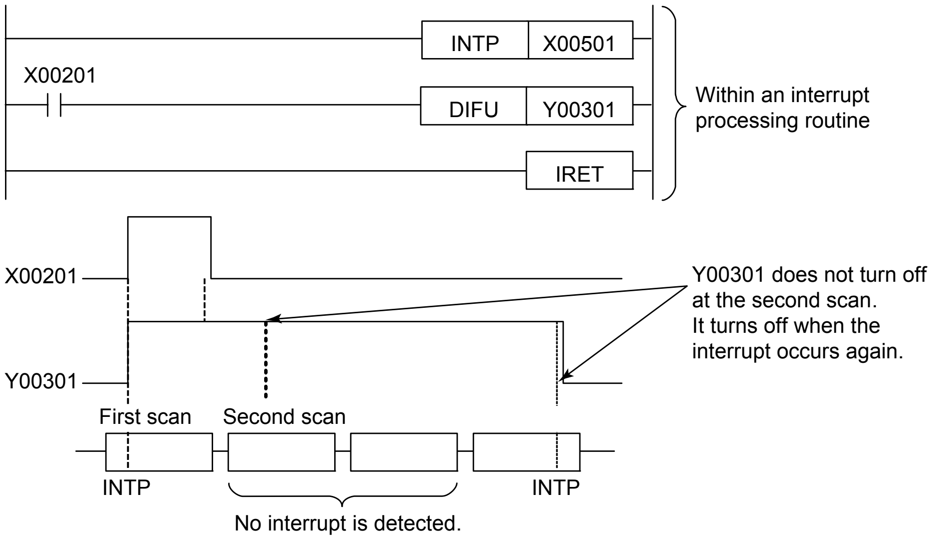

INTP
X00501
X00201
DIFU
Y00301
IRET
Within an interrupt processing routine
X00201
Y00301
Y00301 does not turn off at the second scan.
It turns off when the interrupt occurs again.
First scan
Second scan
INTP
INTP
No interrupt is detected.

Figure 3.10.14 Precaution When Using a Differential Type Instruction in an Interrupt Processing Routine (1)

CAUTION

- Differential up application instruction
  A differential up application instruction will not be executed when it is called after its input condition is switched from OFF to ON state during a scan period during which the input interrupt processing routine is not activated.

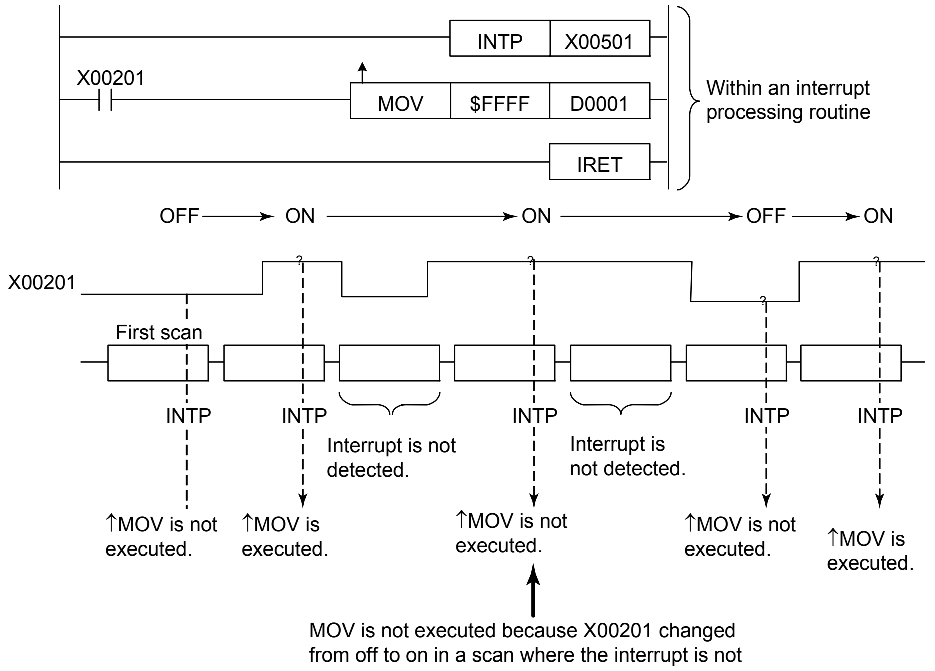
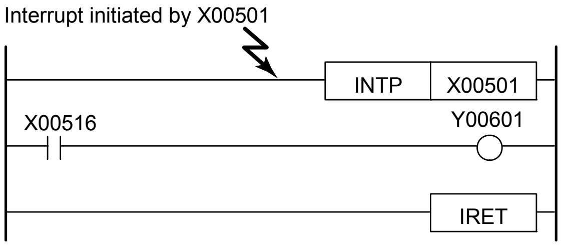

INTP
X00501
X00201
MOV
$FFFF
D0001
IRET
Within an interrupt processing routine
OFF
ON
ON
OFF
ON
X00201
First scan
INTP
INTP
INTP
INTP
INTP
Interrupt is not detected.
Interrupt is not detected.
↑MOV is not executed.
↑MOV is executed.
↑MOV is not executed.
↑MOV is not executed.
↑MOV is executed.
MOV is not executed because X00201 changed from off to on in a scan where the interrupt is not

Figure 3.10.15 Precaution When Using a Differential Type Instruction in an Input Interrupt Processing Routine (2)

- LDU/LDD/UP/DWN/UPX/DWNX instructions
  The result of operation does not equal ON at the next call of input interrupt processing routine even if the input conditions (for UP/DWN/UPX/DWNX instructions) or specified device (for LDU/LDD instructions) makes an OFF-to-ON or ON-to-OFF transition during a scan in which the input interrupt processing routine was not called.

■ Programming Example

The sample code shown below executes an input interrupt processing routine if X00501 is on.



Line No.	Instruction	Operands				
0001	INTP	X00501				
0002	LD	X00516				
0003	OUT	Y00601				
0004	IRET					

F031020.VSD

Figure 3.10.16 Example of an Input Interrupt Processing Program

3.10.4 Disable Interrupt (DI), Enable Interrupt (EI)

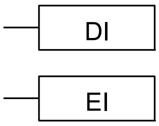
Table 3.10.6 Disable Interrupt, Enable Interrupt

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	67	Disable Interrupt	DI	–[DI]	—	✓	—	1	—	—
	68	Enable Interrupt	EI	–[EI]	—	✓	—	1	—	—

■ Parameter

Disable Interrupt

Enable Interrupt



F031021.VSD

■ Function

Interrupts are enabled initially.

(1) Disable Interrupt

The Disable Interrupt instruction disables input interrupts. When input interrupts are disabled, the program executes normally activating none of the input interrupt processing routines.

Any input interrupts requested while input interrupts are disabled are held pending. Up to 7 input interrupt requests can be held pending at any time. If more than 7 pending input interrupt requests occur, the extra requests are discarded and an interrupt error is generated, and the special relay M201 (instruction processing error) is turned ON.

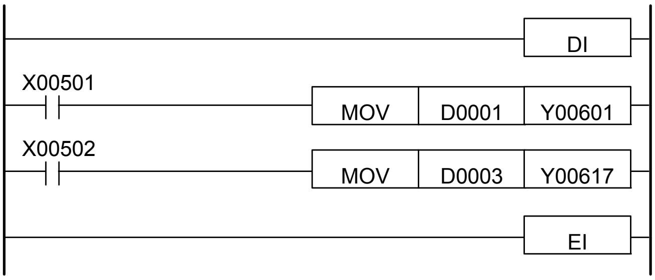
(2) Enable Interrupt

The Enable Interrupt instruction enables input interrupts.

Any input interrupts requested while input interrupts are disabled are processed before normal program execution is restored.

■ Programming Example

The sample code shown below disable input interrupts to prevent the data registers from being modified during data transfer.



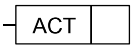
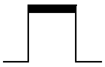
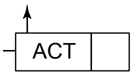
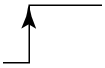
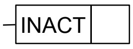
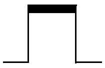
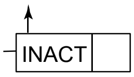
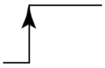
Line No.	Instruction	Operands				
0001	DI					
0002	LD	X00501				
0003	MOV	D0001	Y00601			
0004	LD	X00502				
0005	MOV	D0003	Y00617			
0006	EI					

F031022.VSD

Figure 3.10.17 Example of a Disable Interrupt/Enable Interrupt Program

3.10.5 Activate Block (ACT), Inactivate Block (INACT)

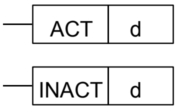
Table 3.10.7 Activate Block, Inactivate Block

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	69	Activate Block	ACT		✓	—		2	—	—
	69P		↑ACT					3	—	—
	70	Inactivate Block	INACT		✓	—		2	—	—
	70P		↑INACT					3	—	—

■ Parameter

Activate Block

Inactivate Block



F031023.VSD

d : Name or number of the block to be activated or inactivated

■ Available Devices

Table 3.10.8 Devices Available for the Activate and Inactivate Block Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Block Name	Indirect Specification, Pointer P
d																✓	No	✓	No

■ Function

The Activate and Inactivate Block instructions activate and inactivate a program block, respectively. These instructions are valid only when the program execution mode is set to "Specified Blocks" during the Configuration function of WideField3, WideField2, WideField or Ladder Diagram Support Program M3. The Inactivate Block instruction is ignored if the program execution mode is set to "All Blocks."

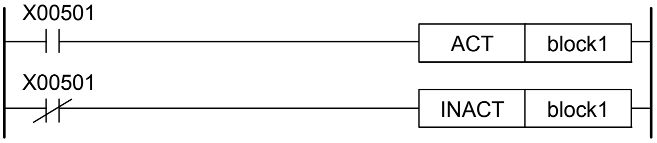
Once a program block is activated or inactivated, it remains in the specified state until it is inactivated or deactivated next time.

TIP

When an ACT instruction is executed in a scan, the block initialization process is done at the end of that scan, and the specified block starts execution at the next scan. When an INACT instruction is executed in a scan, the block initialization process is done at the end of that scan, and the specified block stops execution at the next scan.

■ Programming Example

The sample code shown below activates the program block block1 if X00501 is on and inactivates it if X00501 is off.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	ACT	block1				
0003	LDN	X00501				
0004	INACT	block1				

F031024.VSD

Figure 3.10.18 Example of an Activating/Inactivating Program

3.10.6 For Loop (FOR), Next Loop (NEXT)

Table 3.10.9 For Loop, Next Loop

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	160	For Loop	FOR	FOR	—	✓	—	4	—	—
	161	Next Loop	NEXT	NEXT	—	✓	—	2	—	—

■ Parameter

For Loop FOR d s1 s2

Next Loop NEXT

F031025.VSD

d : Device number of the device storing the loop counter
s1 : Device number of the device storing the initial value of the loop counter (-32768 to 32767)
s2 : Device number of the device storing the limit value of the loop counter (-32768 to 32767)

■ Available Devices

Table 3.10.10 Devices Available for the For Loop and Next Loop Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
s1	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value
*3: Counter current value

■ Function

The FOR and NEXT instructions identify a loop that is executed repeatedly while the counter designated by device d takes values from the initial value designated by s1 to the limit value designated by s2.

d can be referenced within the loop delineated by FOR and NEXT but cannot be updated (written). Normal program execution cannot be guaranteed if d is overwritten. The initial and limit values that are established when the FOR instruction is executed for the first time are used. The number of iterations remains unchanged even if they are altered during the execution of the loop.

Note that these instructions are executed regardless of the input conditions.

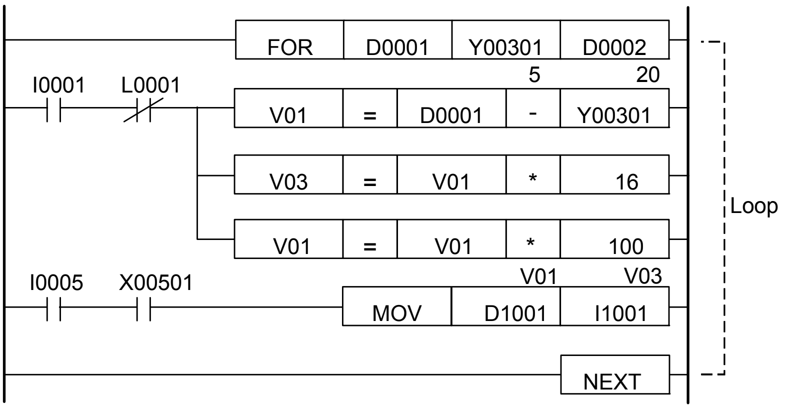
The code between FOR and NEXT is executed only once if s1 (initial value) >= s2 (limit value).

Loops defined by the FOR and NEXT instructions can be nested down to 16 levels.

Use the BRK instruction to force program control out of a FOR-NEXT loop.

■ Programming Example

The sample code shown below repeats the execution of the given steps for 16 iterations while the counter at D0001 takes values from the initial value designated by Y00301 (5) to the limit value designated by D0002 (20).



Line No.	Instruction	Operands				
0001	FOR	D0001	Y00301	D0002		
0002	LD	I0001				
0003	ANDN	L0001				
0004	PUSH					
0005	CAL	V01	=	D0001	–	Y00301
0006	STCRD					
0007	CAL	V03	=	V01	*	16
0008	POP					
0009	CAL	V01	=	V01	*	100
0010	LD	I0005				
0011	AND	X00501				
0012	MOV	D1001;V01	I1001;V03			
0013	NEXT					

F031026.VSD

Figure 3.10.19 Example of a Loop Program

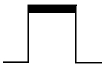
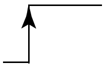
Care must be taken with the following when using differential type instructions in a FOR-NEXT loop.

CAUTION

- DIFU and DIFD instructions
  These instructions turn on their output on the rising and falling edges of their input condition, respectively. The output is turned off when they are executed in a FOR-NEXT loop. The output will then be immediately turned off if the FOR-NEXT loop is executed more once. Consequently, devices whose output is refreshed to an external device at the end of each scan, such as Y (output) relays, may not be turned on at all.
- Differential up type application instructions
  Only the first loop through a FOR-NEXT loop is executed at the rising edge of the input condition. The second and subsequent loops are not executed since the input condition has already been raised.
- LDU/LDD/UP/DWN/UPX/DWNX instructions
  Only the first loop through a FOR-NEXT loop is executed at the rising or falling edge of the input condition (UP/DWN/UPX/DWNX instructions) or specified device (LDU/LDD instructions). The second and subsequent loops are not executed since the input condition has already been raised or lowered. Note that using index modification with the UPX/DWNX instructions allows differential operations for every loop. For details, see the description for the “UPX, DWNX” instructions.

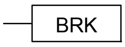
3.10.7 Break Loop (BRK)

Table 3.10.11 Break Loop

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	162	Break Loop	BRK	BRK	✓	—		1	—	—
	162P		↑BRK	↑ BRK				2		

■ Parameter

Break Loop



F031027.VSD

■ Function

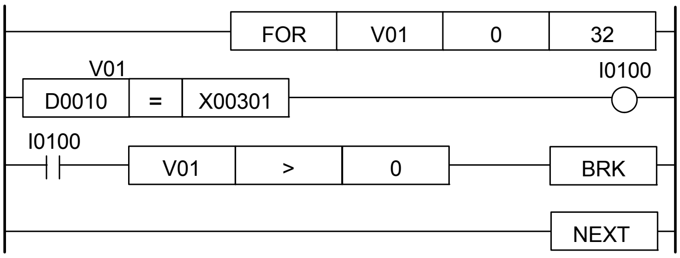
The BRK instruction forces the code between FOR and NEXT instructions to termination and transfers control to the step immediately following the NEXT instruction. This instruction can appear only between FOR and NEXT instructions. If a BRK instruction appears somewhere outside FOR-NEXT loops, an error is signaled and execution skips to the next step.

You cannot use a JMP in place of a BRK. With a JMP, the program regards it as residing within a FOR-NEXT loop and signals an error at the end of the loop.

CAUTION

Code your program so that a BRK instruction is executed only after FOR and NEXT instructions are executed at least once.

- Example program



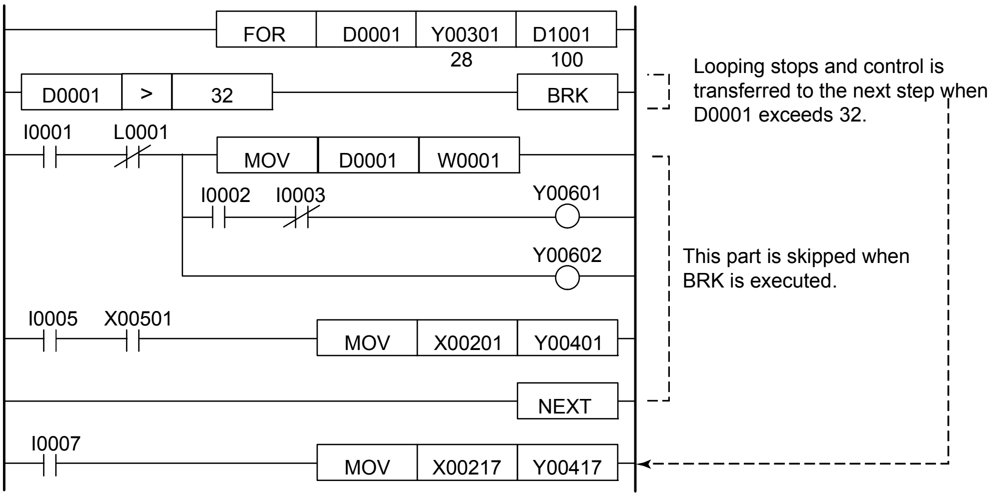
In the code shown above, place dummy data in D0010 so that the value of D0010 does not coincide with that of X00301 when V01=0.

F031028.VSD

Figure 3.10.20 Precautions When Using a BRK Instruction

■ Programming Example

The sample code shown below breaks the loop and transfers control to the next step when the value of the loop counter in D0001 exceeds 32.



Line No.	Instruction	Operands				
0001	FOR	D0001	Y00301	D1001		
0002	CMP	D0001	>	32		
0003	BRK					
0004	LD	I0001				
0005	ANDN	L0001				
0006	PUSH					
0007	MOV	D0001	W0001			
0008	STCRD					
0009	AND	I0002				
0010	ANDN	I0003				
0011	OUT	Y00601				
0012	POP					
0013	OUT	Y00602				
0014	LD	I0005				
0015	AND	X00501				
0016	MOV	X00201	Y00401			
0017	NEXT					
0018	LD	I0007				
0019	MOV	X00217	Y00417			

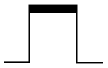
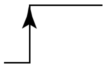
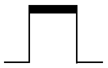
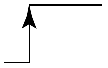
F031029.VSD

Figure 3.10.21 Example of a Program Containing a Break Instruction

3.10.8 Activate Sensor Control Block (CBACT), Inactivate Sensor Control Block (CBINA)

F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.10.12 Activate Sensor Control Block, Inactivate Sensor Control Block

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	71	Activate Sensor Control Block	CBACT	CBACT	✓	—		1	—	—
	71P		↑CBACT	↑ CBACT				2		
	72	Inactivate Sensor Control Block	CBINA	CBINA	✓	—		1	—	—
	72P		↑CBINA	↑ CBINA				2		

■ Parameter

Activate Sensor Control Block	CBACT
Inactivate Sensor Control Block	CBINA

F031030.VSD

■ Function

(1) Activate Sensor Control Block (CBACT)

The CBACT instruction activates a sensor control block. Activation processing is performed when the CBACT instruction is executed.

The CBACT instruction is enabled only if the sensor block has been registered in the component definition of the executable program of WideField3, WideField2 or WideField. If no sensor control block is registered, execution of the instruction is disabled.

By default, the sensor control block is inactive.

(2) Inactivate Sensor Control Block (CBINA)

The CBINA instruction activates a sensor control block. Inactivation processing is performed when the CBINA instruction is executed.

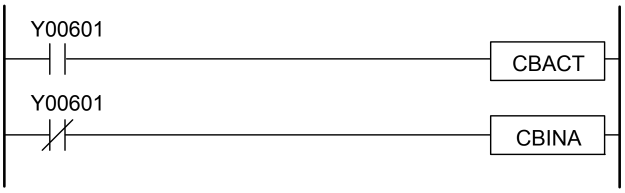
The CBINA instruction is enabled only if the sensor block has been registered in the component definition of the executable program of WideField3, WideField2 or WideField. If no sensor control block is registered, execution of the instruction is disabled.

TIP

A sensor control block is scanned at a high-speed constant scan independently of scanning for normal blocks. Up to one ladder block can be assigned as a sensor control block.

■ Programming Example

The program shown below activates a sensor control block when Y00601 turns ON and inactivates it when Y00601 turns OFF.



Line No.	Instruction	Operands				
0001	LD	Y00601				
0002	CBACT					
0003	LDN	Y00601				
0004	CBINA					

F031031.VSD

Figure 3.10.22 Example Program for Activate and Inactivate Sensor Control Block Instructions

CAUTION

Do not execute the CBACT instruction in an input interrupt routine (code between INTP and IRET instructions). Doing so may fail to activate the sensor control block normally.

SEE ALSO

For details on the specification of the sensor control block, see Section 6.15 of "Sequence CPU Instruction Manual – Functions (for F3SP22-0N, F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S)" (IM 34M06P13-01E), Section A6.15 of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E), Section A6.14 of "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

3.10.9 Disable Sensor Control Block (CBD), Enable Sensor Control Block (CBE)

F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

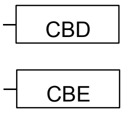
Table 3.10.13 Disable Sensor Control Block, Enable Sensor Control Block

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	73	Disable Sensor Control Block	CBD	CBD	—	✓	—	1	—	—
	74	Enable Sensor Control Block	CBE	CBE	—	✓	—	1	—	—

■ Parameter

Disable Sensor Control Block

Enable Sensor Control Block



F031032.VSD

■ Function

(1) Disable Sensor Control Block

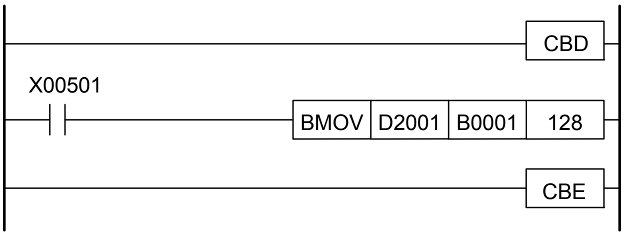
The CBD instruction disables sensor control block execution, even if the block is active.

(2) Enable Sensor Control Block

The CBE instruction enables sensor control block execution. If an interrupt of the sensor control block occurs while sensor control block execution is disabled, the sensor control block is activated immediately after execution is enabled.

■ Programming Example

The program shown below disables sensor control block execution while a table is being set using the BMOV instruction.



Line No.	Instruction	Operands				
0001	CBD					
0002	LD	X00501				
0003	BMOV	D2001	B0001	128		
0004	CBE					

F031033.VSD

Figure 3.10.23 Example of a Program Using Disable and Enable Sensor Control Block

CAUTION

If the sensor control block is disabled for a period that exceeds its execution interval, a sensor control scan timeout error may occur.

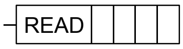
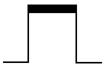
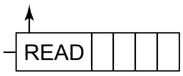
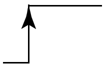
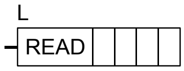
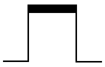
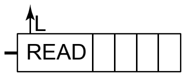
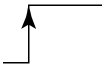
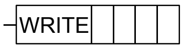
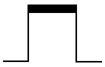
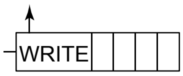
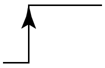
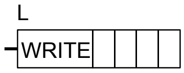
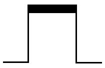
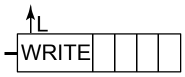
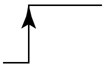
SEE ALSO

For details on the specification of the sensor control block, see Section 6.15 of "Sequence CPU Instruction Manual – Functions (for F3SP22-0N, F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S)" (IM 34M06P13-01E), Section A6.15 of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E), Section A6.14 of "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

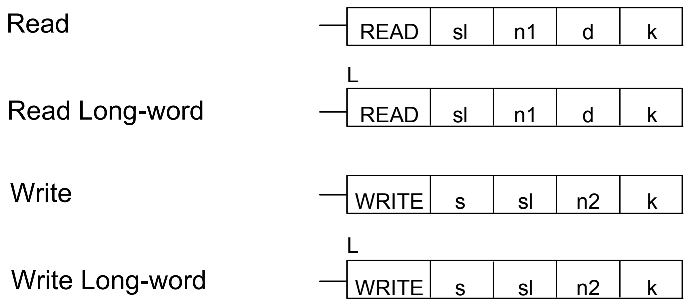
3.11 Special Module Instructions

3.11.1 Read (READ), Read Long-word (READ L), Write (WRITE), Write Long-word (WRITE L)

Table 3.11.1 Read, Write

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	81	Read	READ	 READ	✓	—		5	16 bit	—
	81P		↑READ	 READ				6		
	81L	Read Long-word	READ L	 L READ	✓	—		5	32 bit	—
	81LP		↑READ L	 ↑L READ				6		
	82	Write	WRITE	 WRITE	✓	—		5	16 bit	—
	82P		↑WRITE	 WRITE				6		
	82L	Write Long-word	WRITE L	 L WRITE	✓	—		5	32 bit	—
	82LP		↑WRITE L	 ↑L WRITE				6		

■ Parameter



F031101.VSD

sl : Device number of the first device storing the slot number*1 (3 digits) of the special module
n1 : Device number of the first device storing the first data position number*1 to read
n2 : Device number of the first device storing the first data position number*1 to write
k : Device number of the first device storing the number of words to be transferred
d : Device number of the first device for storing the read data
s : Device number of the first device storing the data to write
*1: sl, n1, n2, and k are handled as a word even in a 32-bit (long-word) instructions.

SEE ALSO

For details on the slot numbers, see Section 1.3.2, "Slot Number," of "Sequence CPU Instruction Manual – Functions (for F3SP22-0N, F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S)" (IM 34M06P13-01E), Section A1.3.2, "Slot Number," of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E), or Section A1.3.2, "Slot Number," of "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

■ Available Devices

(1) Read, Read Long-word

Table 3.11.2 Devices Available for the Read and Read Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
sl	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
n1	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*3	✓*3	✓*3			✓	✓*3	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes
k	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes

*1: Timer current value (may not be used as a long-word parameter)
*2: Counter current value (may not be used as a long-word parameter)
*3: See Section 1.17, "Devices Available As Instruction Parameters."

(2) Write, Write Long-word

Table 3.11.3 Devices Available for the Write and Write Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓			✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
sl	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
n2	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes
k	✓	✓	✓	✓	✓	✓	✓*2	✓*3	✓	✓*1	✓*1	✓	✓	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value (may not be used as a long-word parameter)

*3: Counter current value (may not be used as a long-word parameter)

■ Function

The Read and Write instructions are used to read from and write to special modules. A special module is a module of a type other than contact input, contact output, and contact input/output.

No error is generated even if an attempt is made to read or write to an empty slot.

The usage of the read/write instructions for the special modules is summarized in the table shown below.

Table 3.11.4 Special Modules and Special Module Instructions

Instruction	Special Module Handling 1-word Data	Special Module Handling 2-word Data
Read	✓	—*1
Read Long-word	—*1	✓
Write	✓	—*1
Write Long-word	—*1	✓

*1: Operation is not guaranteed if used.

Examples of data structures of special modules used for read are shown below.

(1) Special module that handles 1-word (16-bit) data

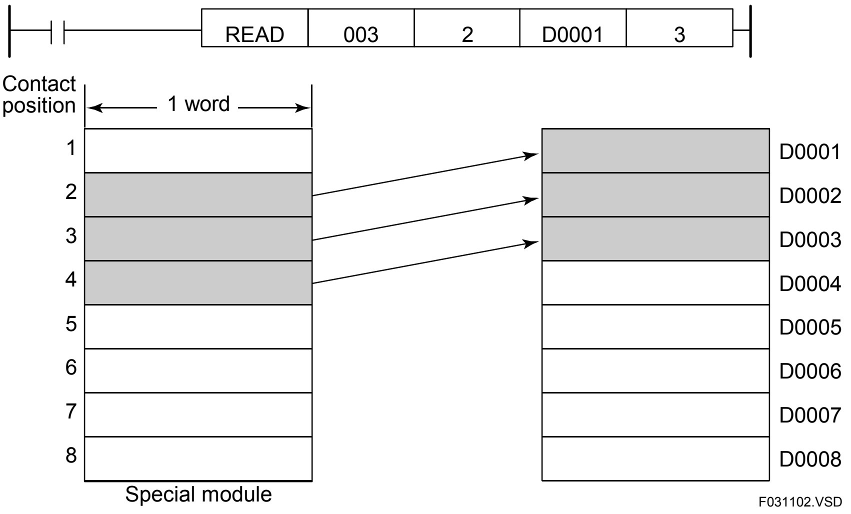


Figure 3.11.1 Reading a Special Module That Handles 1-word Data

(2) Special module that handles 2-word (32-bit) data

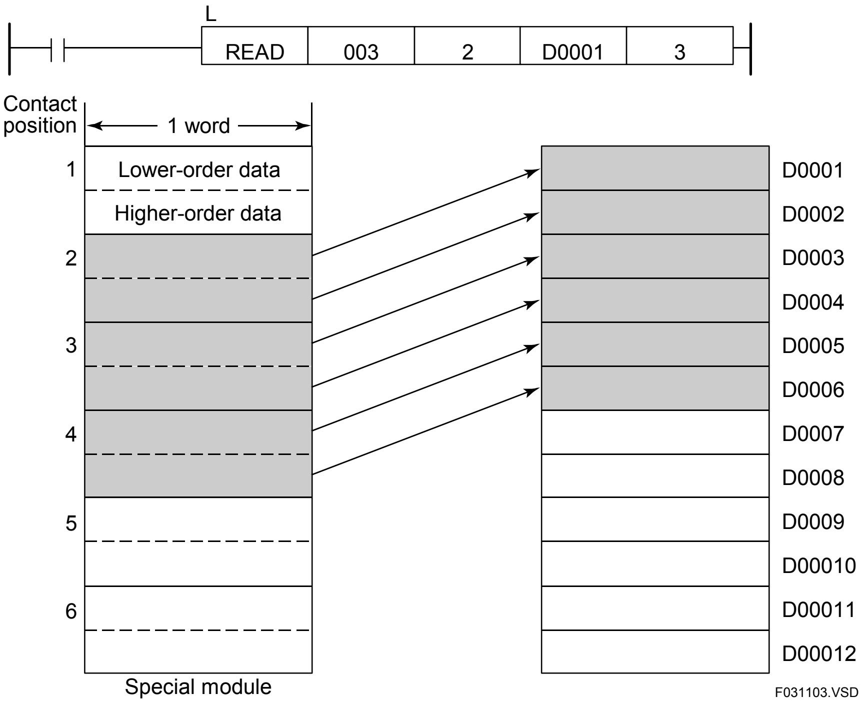


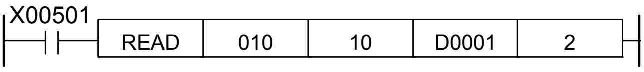
Figure 3.11.2 Reading a Special Module That Handles 2-word Data

SEE ALSO

For the types of data that are handled by special modules, refer to the instruction manual for the individual special modules.

■ Programming Example

The sample code shown below reads 2 words of data from a 1-word handling special module that is installed in slot No. 010 into data registers D0001 and D0002, starting at data position No. 10 if X00501 is on.



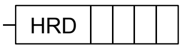
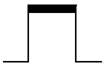
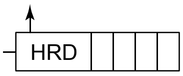
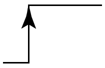
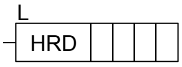
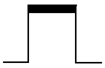
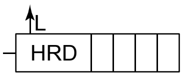
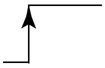
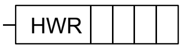
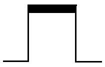
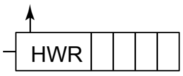
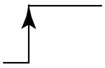
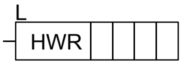
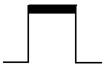
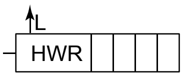
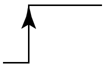
Line No.	Instruction	Operands				
0001	LD	X00501				
0002	READ	010	10	D0001	2	

F031104.VSD

Figure 3.11.3 Example of a Special Module Read Program

3.11.2 High-speed Read (HRD), High-speed Read Long-word (HRD L), High-speed Write (HWR), High-speed Write Long-word (HWR L)

Table 3.11.5 High-speed Read, High-speed Write

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	83	High-speed Read	HRD		✓	—		5	16 bit	—
	83P		↑HRD					6		
	83L	High-speed Read Long-word	HRD L		✓	—		5	32 bit	—
	83LP		↑HRD L					6		
	84	High-speed Write	HWR		✓	—		5	16 bit	—
	84P		↑HWR					6		
	84L	High-speed Write Long-word	HWR L		✓	—		5	32 bit	—
	84LP		↑HWR L					6		

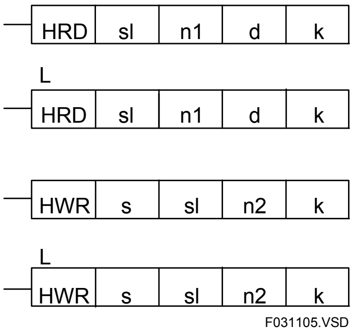
■ Parameter

High-speed Read

High-speed Read Long-word

High-speed Write

High-speed Write Long-word



sl : Device number of the first device storing the slot number (3 digits) of the special module
n1 : Device number of the first device storing the first data position number to read
n2 : Device number of the first device storing the first data position number to write
k : Device number of the first device storing the number of words to be transferred
High-speed Read, High-speed Read Long-word : 1 to 8
High-speed Read Long, High-speed Read Long-word : 1 to 4
d : Device number of the first device for storing the read data
s : Device number of the first device storing the data to be written

SEE ALSO

For details on the slot numbers, see Section 1.3.2, "Slot Number," of "Sequence CPU Instruction Manual – Functions (for F3SP22-0N, F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S)" (IM 34M06P13-01E), Section A1.3.2, "Slot Number," of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E), Section A1.3.2, "Slot Number," of "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

■ Available Devices

(1) High-speed Read, High-speed Read Long-word

Table 3.11.6 Devices Available for the High-speed Read and High-speed Read Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
sl																✓	No	No
n1																✓	No	No
d	✓	✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
k																✓	No	No

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value (may not be used with long-word, high-speed read)
*3: Counter current value (may not be used with long-word, high-speed read)

(2) High-speed Write, High-speed Write Long-word

Table 3.11.7 Devices Available for the High-speed Write and High-speed Write Long-word Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
sl																✓	No	No
n2																✓	No	No
k																✓	No	No

*1: Timer current value (may not be used with long-word, high-speed write)
*2: Counter current value (may not be used with long-word, high-speed write)
*3 : See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The High-speed Read and Write instructions are used to read from and write to special modules at high speed. The basic functionality of these instructions is identical to that of the Read and Write instructions. See also the preceding subsection.

CAUTION

- Differences between the READ/WRITE and HRD/HWR instructions

  HRD and HWR instructions execute at higher speeds than the READ and WRITE instructions. However, because the HRD and HWR instructions access modules at the timing of input/output refreshing, their input and outputs responses are slower than the responses of READ/WRITE instructions. And they are subject to restrictions on the number of data that can be transferred and the type of available devices.

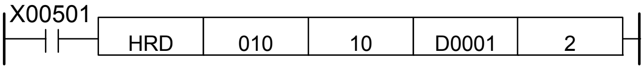
Table 3.11.8 Differences between the READ/WRITE Instructions and HRD/HWR Instructions

Item	READ WRITE	HRD HWR
Execution time *1	Slower than HRD and HWR.	Higher
Special module access timing	Synchronized with the execution of the instruction	Asynchronous with the execution of the instruction (at the timing of input/output refreshing)*3
Number of data words per instruction that can be transferred.	No restriction	Word instruction : 8 maximum Long-word instruction : 4 maximum
Available devices *2	Device may be specified	Only constants are allowed for sl, n1, n2 and k.
Number of instructions that can be used in a program	No restriction	HRD instruction (including long-word instructions): 64 max. HWR instruction (including long-word data instructions): 64 max.

*1: For details, see the appendixes of this manual.
*2: For details, see the description for each instruction.
*3: Eight HRD instructions can be executed in one scan. Therefore, executing 32 HRD instructions require four scans. On the other hand, all of the HWR instructions can be executed in one scan provided the corresponding special modules accept the input of the HWR instruction.

■ Programming Example

The sample code shown below reads 2 words of data at high speed from a 1-word handling special module, which is installed in slot No. 010, starting at data position No. 10, into data registers D0001 and D0002, if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	HRD	010	10	D0001	2	

F031106.VSD

Figure 3.11.4 Example of a High-speed Read Program

CAUTION

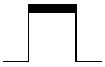
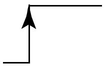
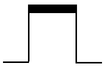
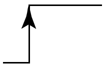
Do not use the HRD and HWR instructions in a sensor control block.

3.12 String Manipulation Instructions

3.12.1 Convert String to Numeric (VAL), Convert String to Long-word Numeric (VAL L)

F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.12.1 Convert String to Numeric, Convert String to Long-word Numeric

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Application Instruction	931	Convert String to Numeric	VAL	VAL	✓	—		5	8 bits	—
	931P		↑VAL	↑ VAL				6		
	931L	Convert String to Long-word Numeric	VAL L	L VAL	✓	—		5	32 bits	—
	931LP		↑VAL L	↑L VAL				6		

■ Parameter

Convert String to Numeric	VAL	n	s	d
Convert String to Long-word Numeric	L VAL	n	s	d

F031201.VSD

n : String format stored as 16-bit integer (0 for auto-detect, 1 for decimal string to binary, 2 for hexadecimal string to binary, 3 for decimal string to BCD, and 4 for sAAAA:BBBB to IEEE single-precision floating point. Option 4 is valid only for long-word instruction)

s : Device number of the first device storing the data to be subject to string-to-numeric conversion

d : Device number of the first device for storing the conversion result

s must be string data, and d must be a 16-bit integer, 32-bit integer, or IEEE single-precision floating-point (32-bit) number.

■ Available Devices

Table 3.12.2 Devices Available for Convert String to Numeric and Convert String to Long-word Numeric Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
n	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓	✓*3	✓	✓	✓	✓	✓	Yes	Yes
s	✓	✓	✓	✓	✓	✓			✓	✓*3	✓*3	✓	✓	✓	✓		Yes	Yes
d		✓	✓	✓*3	✓*3	✓*3	✓*1	✓*2	✓	✓	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes

*1: Timer current value (may not be used as the d parameter for the Convert String to Long-word Numeric function)
*2: Counter current value (may not be used as the d parameter for the Convert String to Long-word Numeric function)
*3: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Convert String to Numeric and Convert String to Long-word Numeric instructions convert a string (terminated with $00) defined by s to numeric data, where the combination of the type of string and that of numeric data is specified by n, and the converted numeric data is stored in d. The types of string that can be converted by these instructions are decimal, hexadecimal, and real number strings. The types of converted numeric data are integer (word or long word) and IEEE single-precision floating-point. Single-precision floating-point data is represented in IEEE format.

● String formats

(1) When n = 0

The type of string to be converted is automatically identified: it is identified as a hexadecimal string and converted to binary data (as if n=2) if it contains one of the characters 'A' to 'F', and it is identified as a decimal string and converted to binary data (as if n=1) if it contains no such characters. Specify n=2 if you want to convert a hexadecimal string containing no such characters to binary data.

(2) When n = 1

A decimal string is converted to a binary number.

The first character of a string may be used as the sign for the data.

If it is '+' ($2B), ' ' ($20), or '0' ($30), or if no sign character is used, the data is positive ('0' may be followed by another '0'); and if it is '-' ($2D), the data is negative.

An error occurs if the converted data is out of the range of a word or long-word format.

Content of s (= -12345)

s	'-' ($2D)	'1' ($31)	
s+1	'2' ($32)	'3' ($33)	
s+2	'4' ($34)	'5' ($35)	
s+3	$00		String terminator (NULL)

F031229.VSD

Figure 3.12.1 Decimal String to Binary Number Conversion (Figure has the same layout as the Device Monitor)

(3) When n = 2

A hexadecimal string is converted to a binary number.

Content of s (=$ABCD)

s	'A' ($41)	'B' ($42)	
s+1	'C' ($43)	'D' ($44)	
s+2	$00		String terminator (NULL)

F031230.VSD

Figure 3.12.2 Hexadecimal String to Binary Number Conversion (Figure has the same layout as the Device Monitor)

(4) When n = 3

A decimal string is converted to BCD representation.

Content of s (=6789)

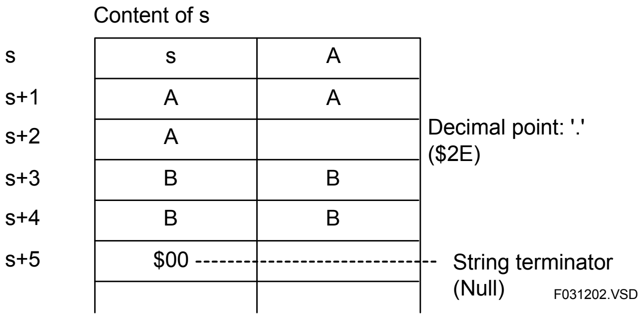
s	'6' ($36)	'7' ($37)	
s+1	'8' ($38)	'9' ($39)	
s+2	$00		String terminator (NULL)

F031231.VSD

Figure 3.12.3 Decimal String to BCD Conversion (Figure has the same layout as the Device Monitor)

A string in sAAAA.BBBB format is converted to an IEEE single-precision floating-point number, where:

s	:	sign +: ‘+’ ($2B), or ‘ ‘ ($20) (space) -: ‘-’ ($2D)
AAAA	:	Integer part (4 digits) A ‘0’ to ’9’ ($30 to $39)
BBBB	:	Fractional part (4 digits)
B	:	‘0’ to ’9’ ($30 to $39)



Content of s

s	s	A
s+1	A	A
s+2	A	
s+3	B	B
s+4	B	B
s+5	$00	

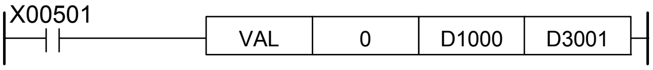
Figure 3.12.4 String to IEEE Floating-point Number Conversion (Figure has the same layout as the Device Monitor)

CAUTION

If a string to be converted contains characters that cannot be converted or if the converted data exceeds the range that can be represented as a word or a long word, an instruction processing error occurs and the special relay M201 turns on. In this case, the Convert String to Numeric or Convert String to Long-word Numeric instruction is not executed.

■ Programming Example

The sample code shown below automatically determines the format of the string starting at D1000, converts it to numeric data, and places the result in the devices starting at D3001 if X00501 is on.



Line No.	Instruction	Operands					
0001	LD	X00501					
0002	VAL	0	D0001	D3001			

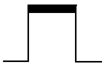
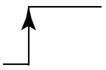
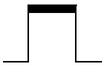
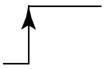
F031203.VSD

Figure 3.12.5 Example of a String to Numeric Conversion Program

3.12.2 Convert Numeric to String (STR), Convert Long-word Numeric to String (STR L)

F3SP25
F3SP35 | F3SP22
F3SP28
F3SP38 | F3SP53
F3SP58
F3SP59 | F3SP66
F3SP67 | F3SP71
F3SP76

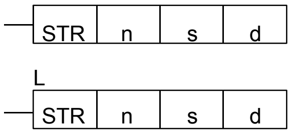
Table 3.12.3 Convert Numeric to String, Convert Long-word Numeric to String

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Application Instruction	932	Convert Numeric to String	STR	STR	✓	—		5	8 bits	—
	932P		↑STR	↑ STR				6		
	932L	Convert Long-word Numeric to String	STR L	L STR	✓	—		5	32 bits	—
	932LP		↑STR L	↑L STR				6		

■ Parameter

Convert Numeric to String

Convert Long-word Numeric to String



F031204.VSD

n : String format stored as 16-bit integer (0 is the same as 1, 1 for binary to decimal string, 2 for binary to hexadecimal, 3 for decimal string to BCD, and 4 for IEEE single-precision floating point to sAAAA:BBBB, and 4 is valid only for long-word instruction)

s : Device number of the first device storing the data to be subject to numeric-to-string conversion

d : Device number of the first device for storing the conversion result

s must be a 16-bit integer, 32-bit (long word) integer, or IEEE single-precision floating-point (32 bits) number, and d must be string data.

■ Available Devices

Table 3.12.4 Devices Available for the Convert Numeric to String and Convert Long-word Numeric to String Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
n	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓	✓*3	✓	✓	✓	✓	✓	Yes	Yes
s	✓	✓	✓	✓	✓	✓			✓	✓*3	✓*3	✓	✓	✓	✓		Yes	Yes
d		✓	✓	✓*3	✓*3	✓*3	✓*1	✓*2	✓	✓	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes

*1: Timer current value (may not be used as the s parameter for the Convert Long-word Numeric to String function)
*2: Counter current value (may not be used as the s parameter for the Convert Long-word Numeric to String function)
*3: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The Convert Numeric to String and Convert Long-word Numeric to String instructions convert numeric data, s, to string data of the format designated by n, and places the result in d. The types of numeric data that can be converted by these instructions are integer (word or long word) and IEEE single-precision floating-point, and the types of conversion result are decimal, hexadecimal, and real number strings.

A single-precision floating-point data must be represented in the IEEE format.

● String formats

(1) When n = 0 (same as n = 1)
Binary numeric data is converted to a decimal string.

The decimal string starts with a space character (' ', $20) for a zero or positive number; it starts with a minus character ('-', $2D) for a negative number.

(2) When n = 2
Binary numeric data is converted to a hexadecimal string.

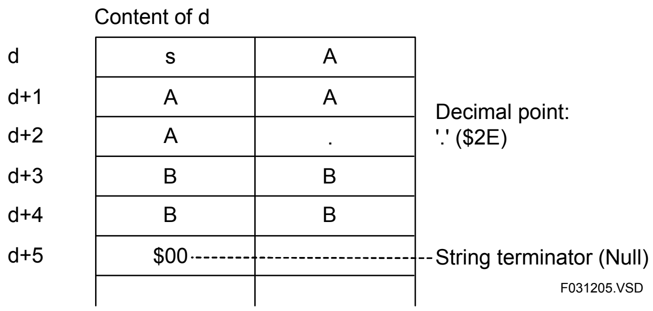
(3) When n = 3
BCD numeric data is converted to a decimal string.

(4) When n = 4 (valid only for the long-word instruction)
An IEEE single-precision floating-point number is converted to a string in sAAAA.BBBB format, where:

s : Sign
‘ ‘ (space; $20 hexadecimal) for a positive number
'-' (minus; $2D hexadecimal) for a negative number

AAAA : Integer part (4 digits)
A: ‘0’ to ’9’ ($30 to $39)

BBBB : Fractional part (4 digits)
B: ‘0’ to ’9’ ($30 to $39)



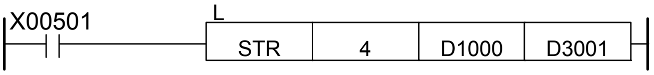
Content of d

d	s	A
d+1	A	A
d+2	A	.
d+3	B	B
d+4	B	B
d+5	$00	

Figure 3.12.6 Example of a Numeric to String Conversion Program

■ Programming Example

The sample code shown below converts the floating-point data in the location from D1000 to D1001 to a string of the format sAAAA.BBBB and places the result in the devices starting at D3001 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	STR L	4	D1000	D3001		

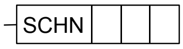
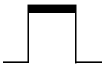
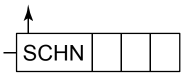
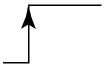
F031206.VSD

Figure 3.12.7 Example of a Numeric to String Conversion Program

3.12.3 String Chain (SCHN)

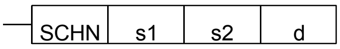
F3SP25
F3SP35 | F3SP22
F3SP28
F3SP38 | F3SP53
F3SP58
F3SP59 | F3SP66
F3SP67 | F3SP71
F3SP76

Table 3.12.5 String Chain

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Application Instruction	933	String Chain	SCHN	 SCHN	✓	—		5	8 bits	—
	933P		↑SCHN	 SCHN				6		

■ Parameter

String Chain



F031207.VSD

s1 : Concatenation data (first half) or device number of the first device storing the data to be concatenated
s2 : Concatenation data (last half) or device number of the first device storing the data to be concatenated
d : Device number of the first device for storing the concatenation result
s1, s2, and d are string data

■ Available Devices

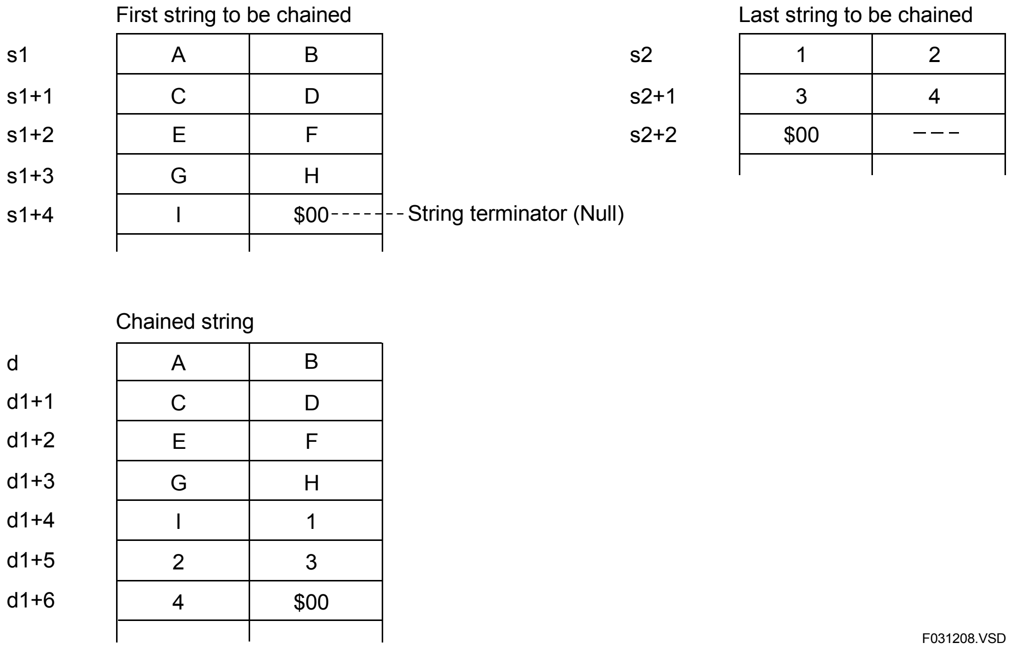
Table 3.12.6 Devices Available for the String Chain Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s1	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1 : See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The String Chain instruction concatenates strings s1 and s2 and places the result in d.



First string to be chained

s1	A	B
s1+1	C	D
s1+2	E	F
s1+3	G	H
s1+4	I	$00

Last string to be chained

s2	1	2
s2+1	3	4
s2+2	$00	---

Chained string

d	A	B
d1+1	C	D
d1+2	E	F
d1+3	G	H
d1+4	I	1
d1+5	2	3
d1+6	4	$00

F031208.VSD

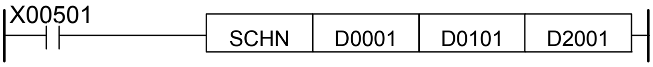
Figure 3.12.8 String Chain

CAUTION

If the character string is longer than 2047 characters, the concatenation result string is longer than 2047 characters, or the concatenation result is 0 characters, an instruction processing error occurs and the special relay M201 turns on. In this case, the String Chain instruction is not executed.

■ Programming Example

The sample code shown below concatenates string starting at D0101 to the end of the string starting at D0001 and places the result in the devices starting at D2001 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	SCHN	D0001	D0101	D2001		

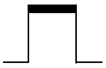
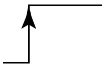
F031209.VSD

Figure 3.12.9 Example of a String Chain Program

3.12.4 String Move (SMOV L)

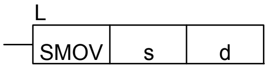
F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.12.7 String Move

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Application Instruction	934	String Move	SMOV L	L SMOV	✓	—		4	8 bits	—
	934P		↑SMOV L	↑L SMOV				5		

■ Parameter

String Move



F031210.VSD

s : Source string or device number of the first device storing the data to move
d : Device number of the first destination device
s and d are string data.

■ Available Devices

Table 3.12.8 Devices Available for the String Move Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

Source string

s1	A	B
s1+1	C	D
s1+2	E	F
s1+3	G	H
s1+4	$00	– – –

String terminator (Null)

Destination string

d1	A	B
d+1	C	D
d+2	E	F
d+3	G	H
d+4	$00	– – –

F031211.VSD

Figure 3.12.10 String Terminator

s may be a literal string of 1 to 4 bytes. Unlike the MOV instruction, the SMOV instruction appends a terminator character (Null ($00)) to the end of the destination string. Use the SMOV instruction to move a string and the MOV or BMOV instruction to move numeric data.

SEE ALSO

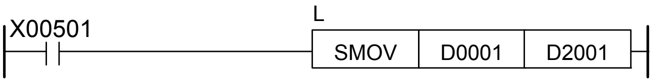
For details, see Section 1.8, "String Manipulation."

CAUTION

If the character string is 0 characters or longer than 2047 characters, an instruction processing error occurs and the special relay M201 turns on. In this case, the String Move instruction is not executed.

■ Programming Example

The sample code shown below moves the string starting at D0001 to the devices starting at D2001 if X00501 is on



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	SMOVL	D0001	D2001			

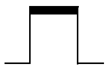
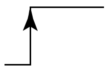
F031212.VSD

Figure 3.12.11 Example of a String Move Program

3.12.5 String Length Count (SLEN)

F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.12.9 String Length Count

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Application Instruction	935	String length count	SLEN	SLEN	✓	—		4	8 bits	—
	935P		↑SLEN	↑ SLEN				5		

■ Parameter

String length count

SLEN	s	d

F031213.VSD

s : String whose length is to be calculated or device number of the first device storing that data
d : Length of the string in bytes
s must be a string and d must be an integer.

■ Available Devices

Table 3.12.10 Devices Available for the String Length Count Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value
*3: Counter current value

■ Function

The String Length Count instruction calculates the length (in bytes) of the string s and places the result in d.

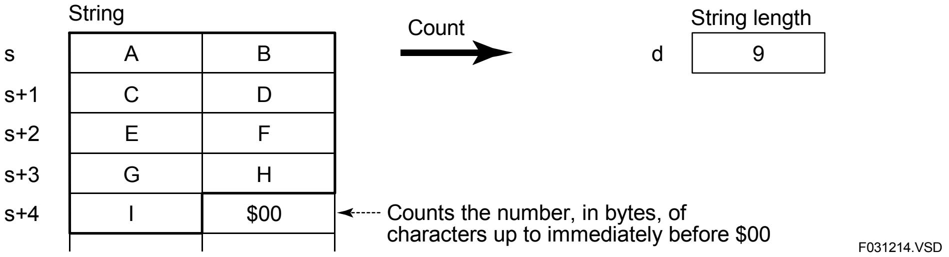

String
s
s+1
s+2
s+3
s+4
A
B
C
D
E
F
G
H
I
$00
Count
String length
d
9
Counts the number, in bytes, of characters up to immediately before $00
F031214.VSD

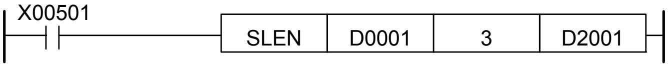
Figure 3.12.12 String Length Count Operation

CAUTION

If the character string is longer than 2047 characters, an instruction processing error occurs and the special relay M201 turns on. In this case, the SLEN instruction is not executed.

■ Programming Example

The sample code shown below loads the length of the string starting at D0001 into D2001 if X00501 is on.



Line No.	Instruction	Operands			
0001	LD	X00501			
0002	SLEN	D0001	3	D2001	

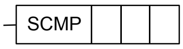
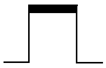
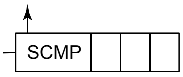
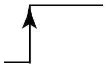
F031225.VSD

Figure 3.12.13 Example of a String Length Count Program

3.12.6 Compare String (SCMP)

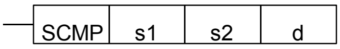
F3SP25
F3SP35 | F3SP22
F3SP28
F3SP38 | F3SP53
F3SP58
F3SP59 | F3SP66
F3SP67 | F3SP71
F3SP76

Table 3.12.11 Compare String

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Application Instruction	936	Compare String	SCMP		✓	—		5	8 bits	—
	936P		↑SCMP					6		

■ Parameter

Compare String



F031216.VSD

s1: String to be compared or device number of the first device storing the data to be compared
s2: String to be compared or device number of the first device storing the data to be compared
d: Device number of the device for storing the comparison result

s1 and s2 must be string data and d must be a 1-bit relay or the least significant bit of a register.

■ Available Devices

Table 3.12.12 Devices Available for the Compare String Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s1	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value
*3: Counter current value

■ Function

The Compare String instruction compares strings s1 and s2 and places the result in d. d is a relay device or the least significant bit of a register device. The bit d is set to ON if s1 and s2 matches and to OFF if the strings differ.

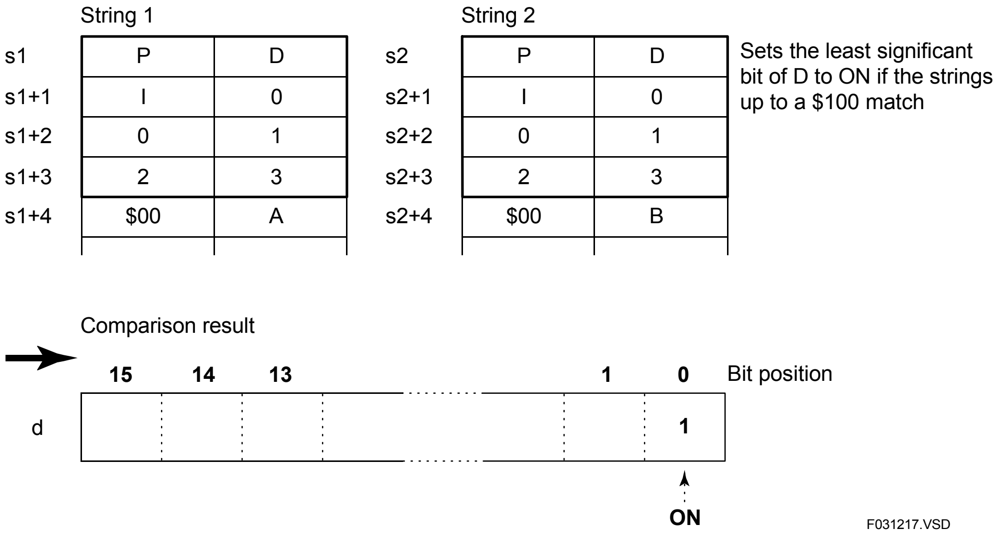


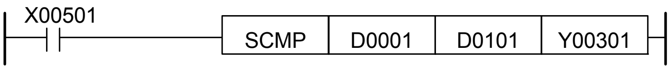
Figure 3.12.14 String Comparison

CAUTION

If each character string to be compared is longer than 2047 characters or if both character strings to be compared are 0 characters, an instruction processing error occurs and the special relay M201 turns on. In this case, the Compare String instruction is not executed.

■ Programming Example

The sample code shown below compares the string starting at D0001 with the string starting at D0101 and sets Y00301 to ON if they match and to OFF otherwise if X00501 is on.



Line No.	Instruction	Operands			
0001	LD	X00501			
0002	SCMP	D0001	D0101	Y00301	

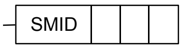
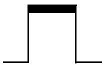
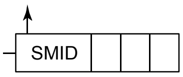
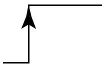
F031218.VSD

Figure 3.12.15 Example of a String Comparison Program

3.12.7 String Middle (SMID)

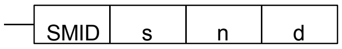
F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.12.13 String Middle

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Application Instruction	937	String Middle	SMID		✓	—		5	8 bits	—
	937P		↑SMID					6		

■ Parameter

String Middle



F031219.VSD

s : Source string or device number of the first device storing the source data
n : Start position and the number of character (in bytes) to be extracted
d : Device number of the first device for storing the extracted substring

s and d must be a string and n must be a 2-word integer (the first word specifies the start position and the second byte specifies the number of characters to extract).

■ Available Devices

Table 3.12.14 Devices Available for the String Middle Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
n	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓		Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1			✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1 : See Section 1.17, "Devices Available As Instruction Parameters."

Function

The String Middle instruction extracts a substring of (n + 1) characters long from the string s starting at the character position n and places the substring in d. This instruction is equivalent to the MID$ statement of BASIC.

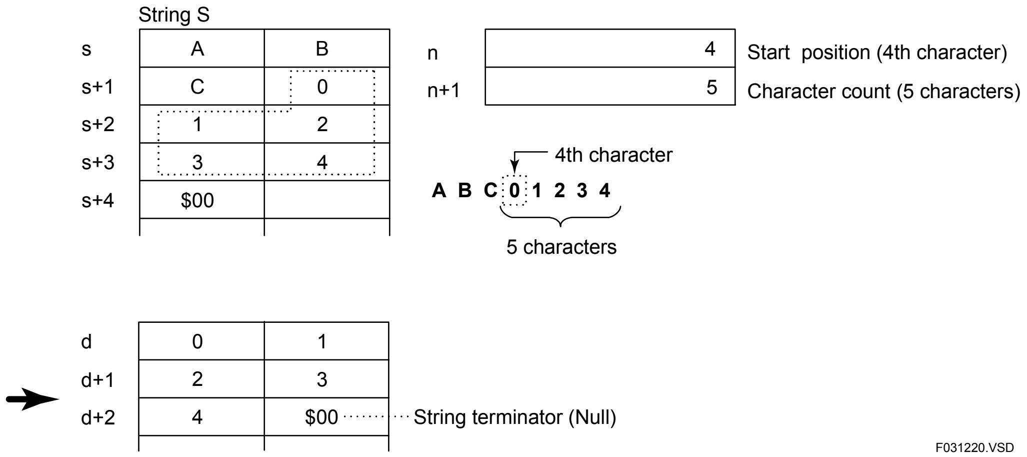


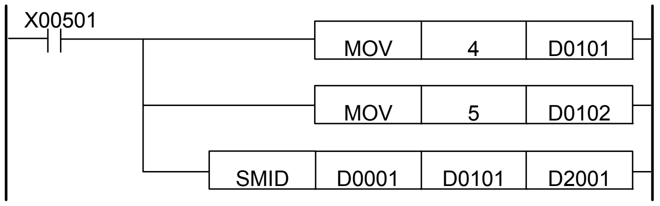
Figure 3.12.16 Substring Extraction

CAUTION

If a String Middle instruction specifies to extract a substring from a string, which is 0 character long, shorter than the substring length, longer than 2047, or starts with a character that is longer than the source string length, an instruction processing error occurs, and the special relay M201 turns on. In this case, the String Middle instruction is not executed.

Programming Example

The sample code shown below extracts 4 characters from the string at D0001 starting at the 4th character position and places the result in devices starting at D2001 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	PUSH					
0003	MOV	4	D0101			
0004	STCRD					
0005	MOV	5	D0102			
0006	POP					
0007	SMID	D0001	D0101	D2001		

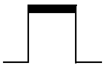
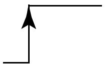
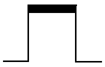
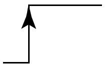
F031221.VSD

Figure 3.12.17 Example of a Substring Program

3.12.8 String Left (SLFT), String Right (SRIT)

F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.12.15 String Left, String Right

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Application Instruction	938	String Left	SLFT	SLFT	✓	—		5	8 bits	—
	938P		↑SLFT	↑ SLFT				6		
	939	String Right	SRIT	SRIT	✓	—		5	8 bits	—
	939P		↑SRIT	↑ SRIT				6		

■ Parameter

String Left — SLFT | s | n | d

String Right — SRIT | s | n | d

F031222.VSD

s : Source string or device number of the first device storing the source data
n : Number of characters to extract (in bytes)
d : Device number of the first device for storing the extracted substring
s and d must be a string and n must be an integer.

■ Available Devices

Table 3.12.16 Devices Available for the String Left and String Right Instructions

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓			✓	✓	✓*3	✓	✓	✓	✓	✓	Yes	Yes
n	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓	✓*3	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*3	✓*3	✓*3			✓	✓	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes

*1: Timer current value
*2: Counter current value
*3 : See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

(1) String Left

The String Left instruction extracts n characters from the left end of string s and places the substring in d. This instruction is equivalent to the LEFT$ statement of BASIC.

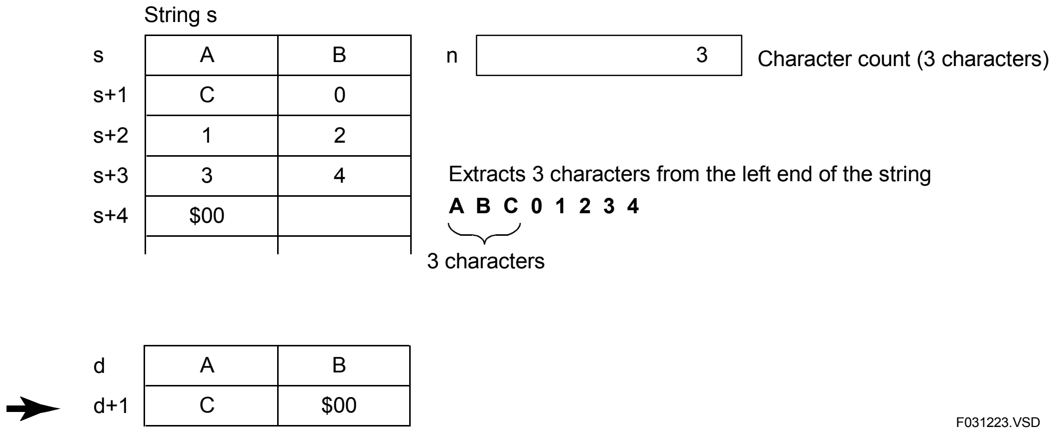

String s
s
A
B
s+1
C
0
s+2
1
2
s+3
3
4
s+4
$00
n
3
Character count (3 characters)
Extracts 3 characters from the left end of the string
A B C 0 1 2 3 4
3 characters
d
A
B
d+1
C
$00
F031223.VSD

Figure 3.12.18   Left Substring Extraction

(2) String Right

The String Right instruction extracts n characters from the right end of string s and places the substring in d. This instruction is equivalent to the RIGHT$ statement of BASIC.

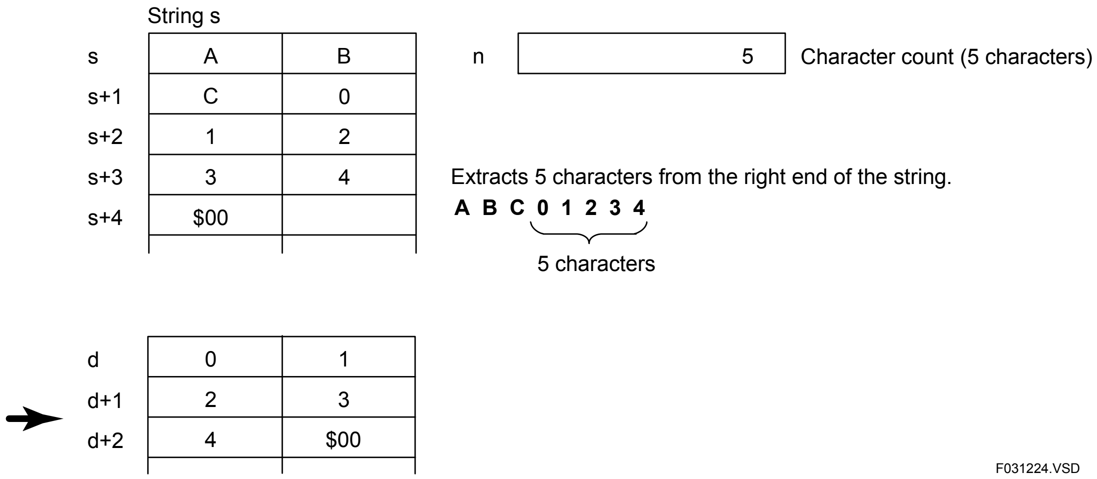


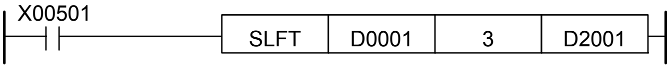
Figure 3.12.19   Right Substring Extraction

CAUTION

If the String Left or String Right instruction specifies to extract a substring from a string that is 0 characters, shorter than the substring, longer than 2047, or starts with a character that is longer than the source string length, an instruction processing error occurs, and the special relay M201 turns on. In this case, the String Left or String Light instruction is not executed.

■ Programming Example

The sample code shown below extracts 3 characters from the left end of the string at D0001 and places the result in devices starting at D2001 if X00501 is on.



Line No.	Instruction	Operands			
0001	LD	X00501			
0002	SLFT	D0001	3	D2001	

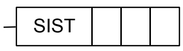
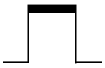
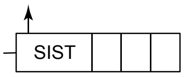
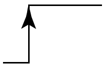
F031225.VSD

Figure 3.12.20 Example of a String Left Instruction

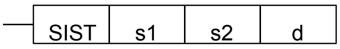
3.12.9 String Search (SIST)

F3SP25 F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.12.17 String Search

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Application Instruction	940	String Search	SIST		✓	—		5	8 bits	—
	940P		↑SIST					6		

■ Parameter

String Search 

F031226.VSD

s1 : String to be searched or device number of the first device storing the string to be searched
s2 : String to search for
d : Device number of the device for storing the search result
s1 and s2 must be string data and d must be an integer.

■ Available Devices

Table 3.12.18 Devices Available for the String Search Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s1	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓			✓	✓	✓*1	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1 : See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value
*3: Counter current value

■ Function

The String Search instruction searches string s1 for substring s2 and, if a match is found, places, in d, the character position (number of bytes) of the first matching character in the original string.

d is set to 1 (first byte) if s2 matches s1 starting at the beginning of s1. d is set to 0 if s1 does not contain s1.

This instruction is equivalent to the INSTR statement of BASIC (except that the starting position for the search is not specified).

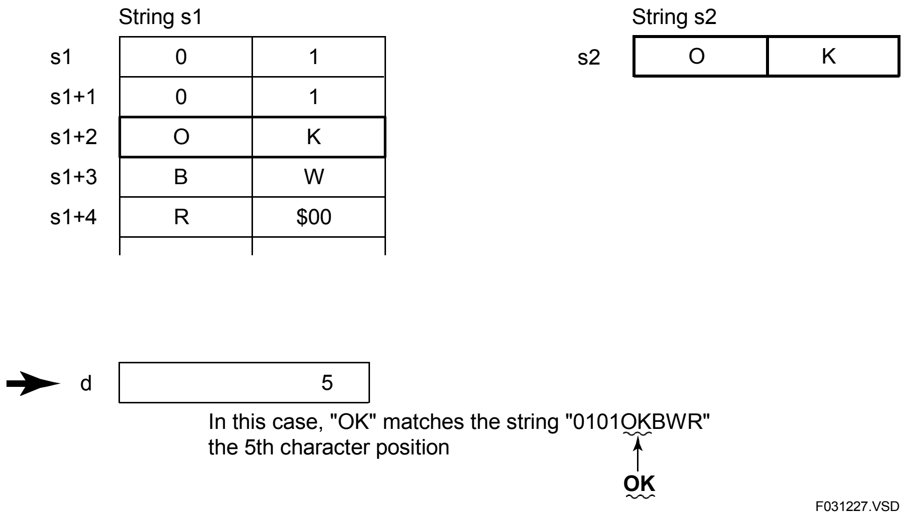
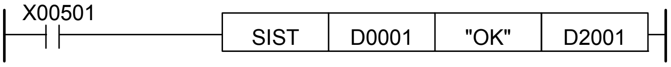

String s1
s1
0
1
s1+1
0
1
s1+2
O
K
s1+3
B
W
s1+4
R
$00
String s2
s2
O
K
d
5
In this case, "OK" matches the string "0101OKBWR"
the 5th character position
OK
F031227.VSD

Figure 3.12.21 String Search

■ Programming Example

The sample code shown below searches the string starting at D0001 for substring “OK” and places the character position of the first matching character in D2001 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	SIST	D0001	"OK"	D2001		

F031228.VSD

Figure 3.12.22 Example of a String Search Program

3.13 Structures and Macro Instructions

3.13.1 Structure Pointer Declaration (STRCT)

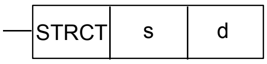
F3SP22-0S F3SP28-3S F3SP38-6S | F3SP53-4S F3SP58-6S F3SP59-7S | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.13.1 Structure Pointer Declaration

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Application Instruction	986	Structure Pointer Declaration	STRCT	STRCT	—	✓	—	4	—	—

■ Parameter

Structure Pointer Declaration



F031416.VSD

s : Structure pointer (Q01 or Q02)
d : Structure type name

■ Available Devices

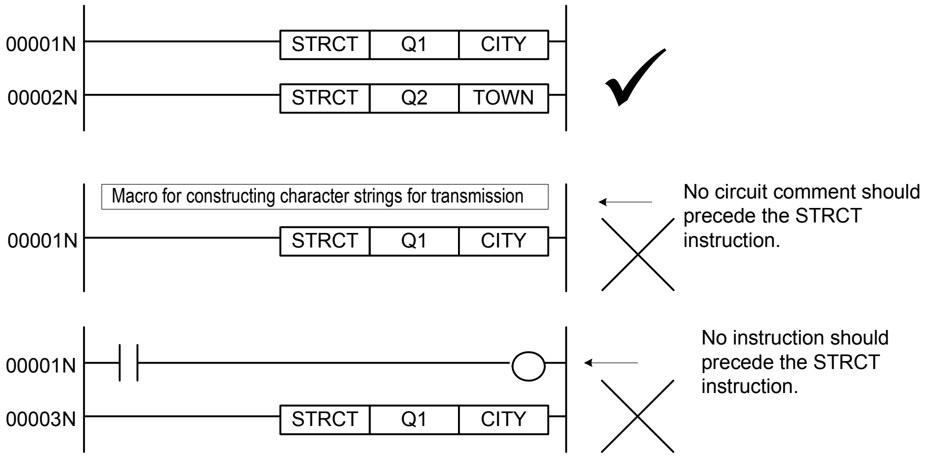
Table 3.13.2 Parameters Available for Structure Pointer Declaration

Parameter	Constant	Structure Type	Structure Name	Structure Pointer	Array Constant Specification	Array Device Specification
s				✓	No	No
d		✓			No	No

■ Function

The Structure Pointer Declaration instruction declares the type of structures to be passed to structure macro instructions.

It must be coded at the beginning of structure macro instructions.


00001N
STRCT
Q1
CITY
00002N
STRCT
Q2
TOWN
✓
Macro for constructing character strings for transmission
00001N
STRCT
Q1
CITY
No circuit comment should precede the STRCT instruction.
00001N
00003N
STRCT
Q1
CITY
No instruction should precede the STRCT instruction.

F031417.VSD

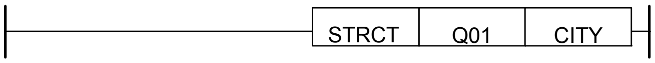
Figure 3.13.1 Position of Structure Pointer Declaration Instruction

SEE ALSO

For more details on structures, see "FA-M3 Programming Tool WideField3" (IM 34M06Q16-□□E) or "FA-M3 Programming Tool WideField2" (IM 34M06Q15-01E).

■ Programming Example

The following sample code declares the structure pointer Q1 to be of structure type CITY:



Line No.	Instruction	Operands				
0001	STRCT	Q01	CITY			

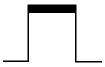
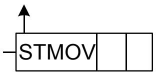
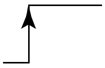
F031418.VSD

Figure 3.13.2 Example of a Structure Pointer Declaration Program

3.13.2 Structure Move (STMOV)

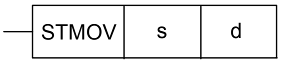
F3SP22-0S
F3SP28-3S
F3SP38-6S | F3SP53-4S
F3SP58-6S
F3SP59-7S | F3SP66
F3SP67 | F3SP71
F3SP76

Table 3.13.3 Structure Move

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	987	Structure Move	STMOV	STMOV	✓	–		26	–	–
	987P		↑ STMOV	 STMOV				27		

■ Parameter

Structure Move



F031419.VSD

s : Name of source structure
d : Name of destination structure

SEE ALSO

For more details on structures, see "FA-M3 Programming Tool WideField3" (IM 34M06Q16-□□E) or "FA-M3 Programming Tool WideField2" (IM 34M06Q15-01E).

■ Available Devices

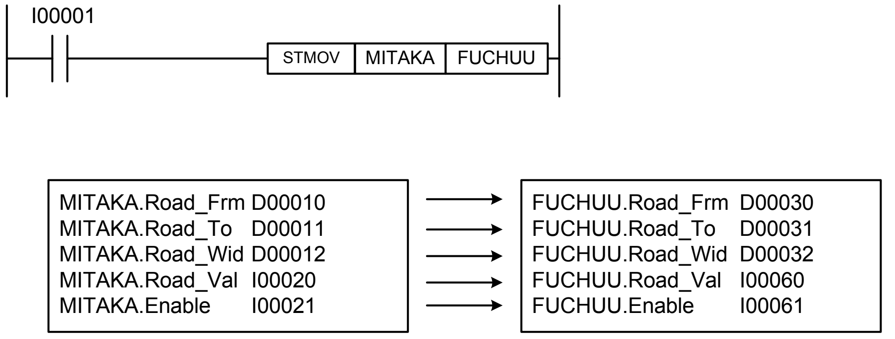
Table 3.13.4 Parameters Available for Structure Move

Parameter	Constant	Structure Type	Structure Name	Structure Pointer	Array Constant Specification	Array Device Specification
s			✓	✓	Yes	Yes
d			✓	✓	Yes	Yes

■ Function

The Structure Move instruction moves the content of structure designated as s to structure designated as d.

s and d structures must be of the same structure type.



F031420.VSD

Figure 3.13.3 Structure Move

The value of member ".Road_Frm" of structure "MITAKA" is moved to member ".Road_Frm" of structure "FUCHUU." Likewise, the values of the other members, from ".Road_To" to ".Enable" of structure "MITAKA" are moved to respective members of structure "FUCHUU."

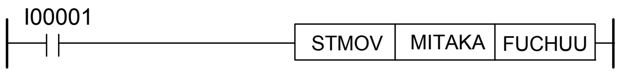
Values of X and Y relay members of a structure are also moved.

CAUTION

When specifying an array index using devices, do not exceed the array boundary. The CPU module does not perform a range check. However, if the range of the array is exceeded, it may result in modification of unintended devices so proper operation is not guaranteed. A negative array index generates an instruction processing error and the instruction will not be executed.

■ Programming Example

The sample code below moves the content of the structure MITAKA to the structure FUCHUU when I00001 is turned on.



Line No.	Instruction	Operands			
0001	LD	I00001			
0002	STMOV	MITAKA	FUCHUU		

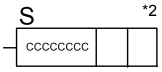
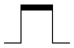
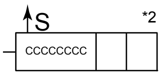
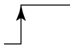
F031421.VSD

Figure 3.13.4 Example of a Structure Move Program

3.13.3 Structure Macro Instruction Call (SCALL)

F3SP22-0S F3SP28-3S F3SP38-6S | F3SP53-4S F3SP58-6S F3SP59-7S | F3SP66 F3SP67 | F3SP71 F3SP76

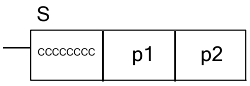
Table 3.13.5 Structure Macro Instruction Call

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	985	Structure Macro Instruction Call	SCALL*1		✓	–		35	1/16 bits	–
	985P		↑ SCALL*1					36		

*1: Must be entered in mnemonics.
*2: cccccccc: Name of a macro instruction to be called (Alphanumeric string up to 8 characters long, beginning with two letters).

■ Parameter

Structure Macro Instruction Call



F031412.VSD

cccccccc : Name of input macro instruction to call
(Alphanumeric string up to 8 characters long, beginning with two letters)
p1 : Structure 1 to be passed to structure macro
p2 : Structure 2 to be passed to structure macro

CAUTION

Both parameters 1 and 2 must be specified. Enter a constant 0 as dummy parameter if a parameter is not to be used in the structure macro instruction object.

SEE ALSO

For details on structures, see "FA-M3 Programming Tool WideField3" (IM 34M06Q16-☐☐E) or "FA-M3 Programming Tool WideField2" (IM 34M06Q15-01E).

■ Available Devices

Table 3.13.6 Parameters Available for Structure Macro Instruction Call

Parameter	Constant	Structure Type	Structure Name	Structure Pointer	Array Constant Specification	Array Device Specification
p1	✓		✓		Yes	No
p2	✓		✓		Yes	No

■ Function

When this instruction is executed, the registered structure macro instruction is executed.

This instruction differs from the Macro Call (MCALL) instruction in that it passes to a macro instruction all data of a specified structure collectively.

A macro instruction called by a SCALL instruction is also known as a structure macro instruction.

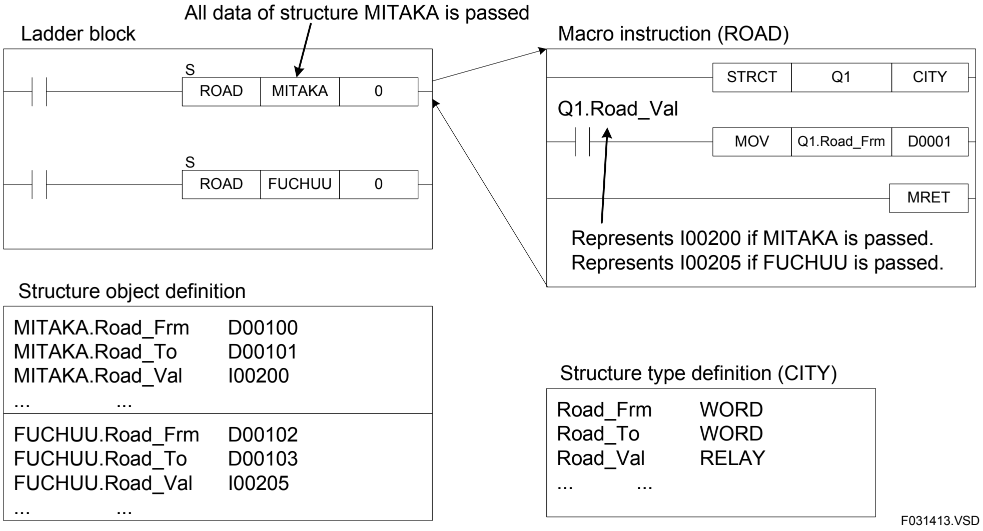


Figure 3.13.5 Structure Macro Instructions

CAUTION

- A structure macro instruction accepts only a structure. It does not accept devices directly.
- You can enter a constant for an unused parameter, but may not utilize constants as arguments.
- When a Structure Macro Instruction Call is executed, parameters 4 to 8 passed using the Parameter instruction (PARA) will be destroyed. Therefore, if you want to pass parameters to a Macro Call or Input Macro Instruction Call using the Parameter instruction (PARA), execute the Parameter instruction (PARA) immediately before the Macro Call or Input Macro Instruction Call.
- Ensure that the structure to be passed and the STRCT instruction to be called have the same structure type. Otherwise a parameter error may occur in the macro instruction.

CAUTION

(1) Exercise caution when using differential type instructions in a macro instruction object (called object).

- When using DIFU or DIFD instructions:
  The output turns on at a rising or falling edge of the input condition.
  Once the output turns on, it stays on until the same macro instruction is called and the DIFU or DIFD is executed again.
- When using differential up application instructions:
  If the input condition changes from off to on during a scan period in which the macro instruction is not executed, the differential application instruction is not executed even if the macro instruction is executed in the next scan period.
- When using LDU/LDD/UP/DWN/UPX/DWNX instructions:
  If the input condition (for UP/DWN/UPX/DWNX instructions) or the specified device (for LDU/LDD instructions) rises (or falls) during a scan period in which the macro instruction is not executed, the operation result does not turn on even if the macro instruction is executed in the next scan period.

(2) The maximum number of HRD instructions, HWR instructions, or labels that can be used in an executable program is limited by the total computed over all program and macro instruction objects (called object).

CAUTION

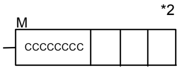
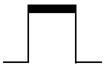
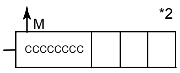
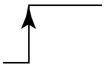
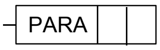
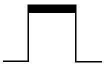
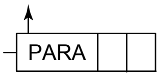
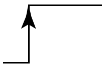
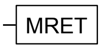
When an interrupt input of the input module rises, an interrupt program (program codes beginning with an INTP instruction and ending with an IRET instruction) is executed, regardless of whether a macro instruction object (called object) is being executed.

To avoid execution of interrupt programs during macro instruction execution, use the Disable Interrupt (DI) and Enable Interrupt (EI) instructions, as described later.

3.13.4 Macro Call (MCALL), Parameter (PARA), Macro Return (MRET)

F3SP25
F3SP35 | F3SP22 F3SP28 F3SP38 | F3SP53 F3SP58 F3SP59 | F3SP66 F3SP67 | F3SP71 F3SP76

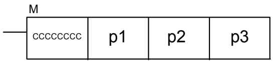
Table 3.13.7 Macro Call, Parameter, and Macro Return

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	996	Macro Call	MCALL*1		✓	—		5	1/16/32 bits	—
	996P		↑ MCALL*1					6		
	995	Parameter	PARA		✓	—		4	1/16/32 bits	—
	995P		↑ PARA					5		
	998	Macro Return	MRET		—	✓	—	1	—	—

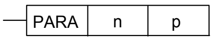
*1: Must be entered in mnemonics.
*2: cccccccc: Name of macro instruction to be called (alphanumeric character string up to 8 characters long, beginning with two letters).

■ Parameter

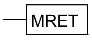
Macro Call



Parameter



Macro Return



F031315.VSD

p1 : Parameter 1 to be passed to macro instruction
p2 : Parameter 2 to be passed to macro instruction
p3 : Parameter 3 to be passed to macro instruction
n : Parameter number to be passed to macro instruction (4 ≤ n ≤ 16)
p : Parameter to be passed to macro instruction

CAUTION

- Up to three parameters may be passed to a macro instruction directly. To pass more than three parameters to a macro instruction, use the Parameter instruction.
- All three parameters (p1, p2 and p3) must be specified. You may assign a zero constant or any dummy value to parameters, which are not used within the macro instruction (called program).

■ Available Devices

Table 3.13.8 Devices Available for Macro Call and Parameter

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
p1	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓	✓	✓	✓	✓	✓	✓	Yes	Yes
p2	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓	✓	✓	✓	✓	✓	✓	Yes	Yes
p3	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓	✓	✓	✓	✓	✓	✓	Yes	Yes
n	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓	✓	✓	✓	✓	✓	✓	Yes	Yes
p	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓	✓	✓	✓	✓	✓	✓	Yes	Yes

*1 : Timer current value
*2 : Counter current value
Note: See Section 1.17, "Devices Available As Instruction Parameters."

CAUTION

You may not pass time-out relays (T) or end-of-count relays (C) directly as parameters to a macro instruction. Instead, copy the relay value to an internal relay (I) and then pass the internal relay as a parameter.

You may not pass a long-word constant (e.g., IEEE single-precision floating-point constant) directly as a parameter to a macro instruction. Instead, assign the long word value to a data register, and then pass the register as a parameter.

TIP

The macro function allows a user to define multiple instructions requiring multiple processing steps as a single instruction. These macros can then be assigned names and be used like any other ordinary instruction.

■ Function

(1) Macro Call

Transfers control to a specified macro instruction.

When the specified macro instruction completes execution, control is returned to the step immediately following the Macro Call instruction.

If a 16-bit device (e.g. data register (D)) is passed as a parameter to a macro instruction and is used in the macro instruction object as bit data for bit instructions (e.g. Load (LD) instruction), the Instruction Processing Error special relay (M201) turns on.

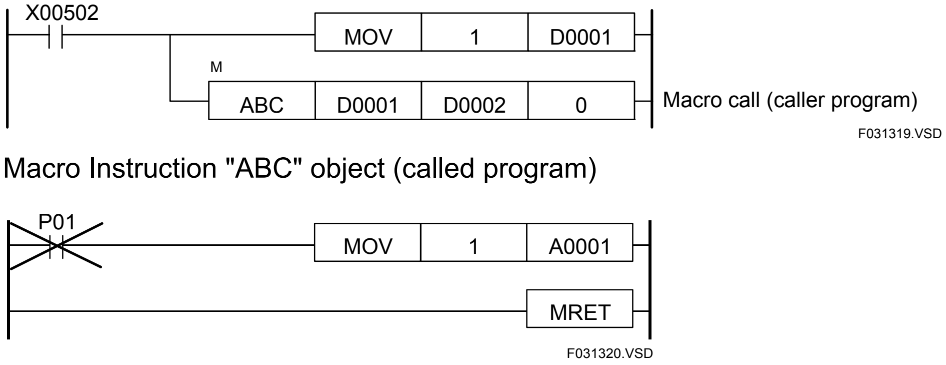

X00502
MOV
1
D0001
M
ABC
D0001
D0002
0
Macro call (caller program)
F031319.VSD
Macro Instruction "ABC" object (called program)
P01
MOV
1
A0001
MRET
F031320.VSD

Figure 3.13.6 Macro Instruction Call

CAUTION

Exercise caution when using link relays and registers as macro instruction parameters. Within a macro instruction, link refreshing is not performed for transfers and processing of these parameters and may result in incorrect processing.

You should instead transfer the content of the link relay or register to another device and then use this device as macro instruction parameter.

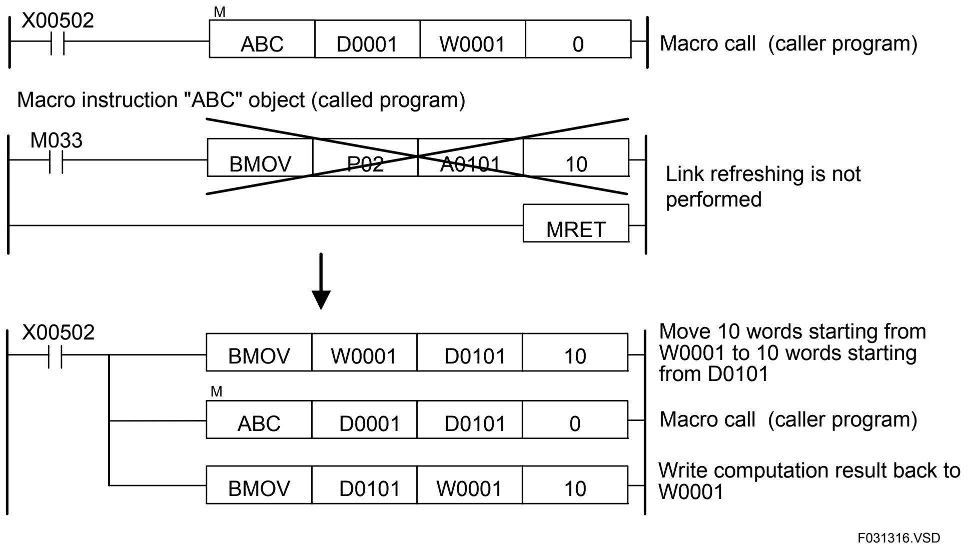

X00502
M
ABC
D0001
W0001
0
Macro call (caller program)
Macro instruction "ABC" object (called program)
M033
BMOV
~~P02~~
~~A0101~~
10
Link refreshing is not performed
MRET
X00502
BMOV
W0001
D0101
10
Move 10 words starting from W0001 to 10 words starting from D0101
M
ABC
D0001
D0101
0
Macro call (caller program)
BMOV
D0101
W0001
10
Write computation result back to W0001
F031316.VSD

Figure 3.13.7 Precautions about Macro Instruction Parameters

SEE ALSO

For details on link refreshing, see Section 3.10.2 of "Sequence CPU Instruction Manual – Functions (for F3SP22-0N, F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S)" (IM 34M06P13-01E), Section A3.10.2 of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E), or Section A3.10.2 of "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

(2) Parameter Instruction

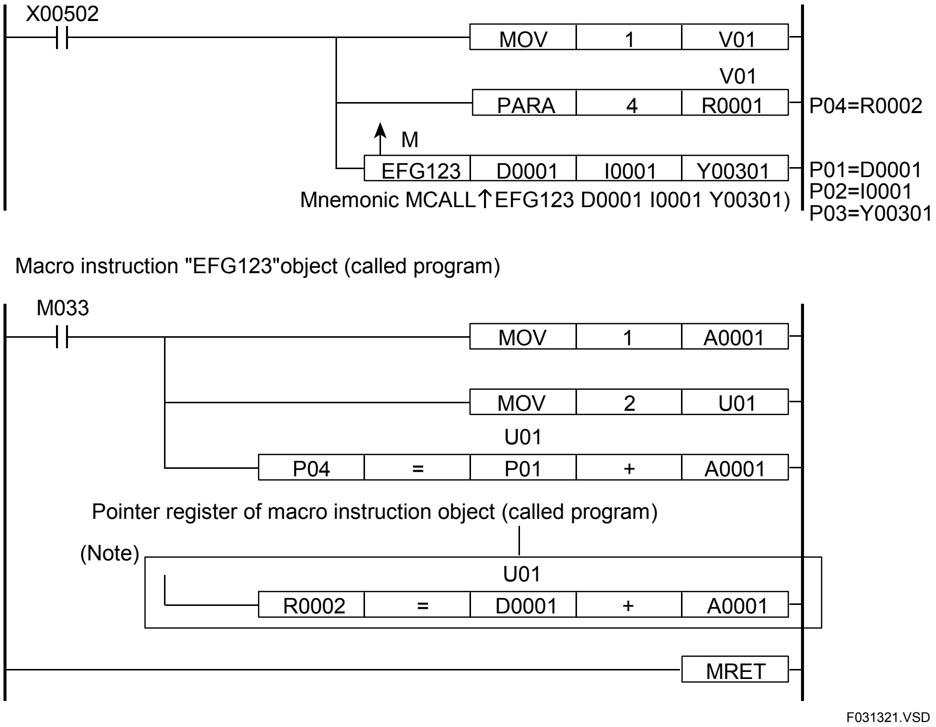
Passes a device designated by parameter P as parameter n to a macro instruction.

It is used when passing more than three parameters to a macro instruction or an input macro instruction.

The Parameter instruction must be executed before a macro instruction. Referring to parameter number 4 (P4) or greater before executing a Parameter instruction turns on the Instruction Processing Error (M201) special relay.

Pass parameters 1 to 3 directly to a macro instruction as parameters of a Macro Call (see (1) Macro Call above) instruction.

If parameter n is less than 4 or greater than 16, an instruction processing error is generated.



Note: Can be used in a macro instruction object (called program).

Figure 3.13.8 Example Use of a Pointer Register

CAUTION

- Passing an index-modified device as macro instruction parameter actually passes the device after index modification.
- In the example shown in the above figure, since V01=1, so "R0001; V01"=R0002.
- Index modification of a pointer register acts on the parameter passed.
- In the example shown in the above figure, since P01=D0001 and U01=2, so "P01; U01"=D0003.
- Parameters 4 to 8 passed using a Parameter instruction (PARA) are destroyed when a Structure Macro Instruction Call is executed. Therefore, to pass parameters to a Macro Call or Input Macro Instruction Call using a Parameter instruction (PARA), place the Parameter instruction (PARA) immediately before the Macro Call or Input Macro Instruction Call.

(3) Macro Return

Designates the end of a macro instruction, input macro instruction, or structure macro instruction.

Put a Macro Return at the end of every macro instruction object (called object). Do not place any program code after a Macro Return instruction.

CAUTION

(1) Exercise caution when using differential type instructions in a macro instruction object (called program).

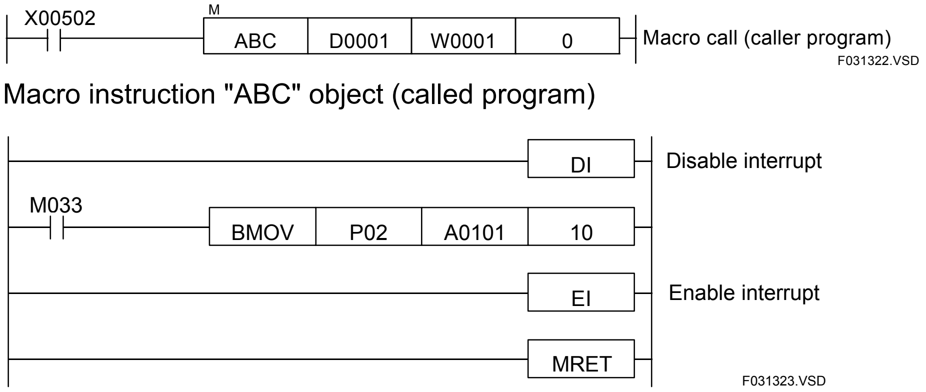
- When using DIFU or DIFD instructions:
  The output turns on at a rising or falling edge of the input condition.
  Once the output turns on, it stays on until the same macro instruction is called and the DIFU or DIFD is executed again.
- When using differential up application instructions:
  If the input condition changes from off to on during a scan period in which the macro instruction is not executed, the differential application instruction is not executed even if the macro instruction is executed in the next scan period.
- When using LDU/LDD/UP/DWN/UPX/DWNX instructions:
  If the input condition (for UP/DWN/UPX/DWNX instructions) or the specified device (for LDU/LDD instructions) rises (or falls) during a scan period in which the macro instruction is not executed, the operation result does not turn on even if the macro instruction is executed in the next scan period.

(2) The maximum number of HRD instructions, HWR instructions, or labels that can be used in an executable program is limited by the total computed over all program and macro instruction objects (called program).

CAUTION

When an interrupt input of the input module changes from OFF to ON, an interrupt program (program codes beginning with an INTP instruction and ending with an IRET instruction) is executed, regardless of whether a macro instruction object (called program) is being executed.

To avoid execution of interrupt programs during macro instruction execution, use the Disable Interrupt (DI) and Enable Interrupt (EI) instructions, as shown below.



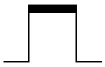
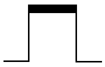
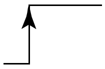
Macro instruction "ABC" object (called program)

Figure 3.13.9 Example for Disabling/Enabling Interrupt Program Execution

3.13.5 Input Macro Instruction Call (NCALL), Output of Input Macro (NMOUT)

F3SP22-0S F3SP28-3S F3SP38-6S | F3SP53-4S F3SP58-6S F3SP59-7S | F3SP66 F3SP67 | F3SP71 F3SP76

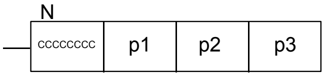
Table 3.13.9 Input Macro Instruction Call and Output of Input Macro

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	981	Input Macro Instruction Call	NCALL	N *1 cccccccc	—	✓		5	1/16/32 bits	—
	309	Output of Input Macro	NMOUT	NMOUT	✓	—		2	1/16 bits	—
	309P		↑ NMOUT	↑ NMOUT				3		

*1: "cccccccc" represents the name of a macro instruction to be called (alphanumeric string of up to 8 characters, beginning with two letters).

■ Parameter

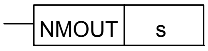
Input Macro Instruction Call



F031324.VSD

cccccccc : Name of a macro instruction to be called (alphanumeric string of up to 8 characters, beginning with 2 letters).
p1 : Parameter 1 to be passed to input macro
p2 : Parameter 2 to be passed to input macro
p3 : Parameter 3 to be passed to input macro

Output of Input Macro



F031325.VSD

s : Device representing the logical operation result of an input macro
For a constant, 0=off, non-zero=on.
For a relay, 0=off, 1=on.
For a register, 0=off, non-zero=on.

CAUTION

All three parameters (p1, p2 and p3) must be specified. You may assign a zero constant or any dummy value to parameters, which are not used within the macro instruction (called program).

■ Available Devices

Table 3.13.10 Devices Available for Input Macro Instruction Call

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
p1	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓	✓	✓	✓	✓	✓	✓	Yes	Yes
p2	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓	✓	✓	✓	✓	✓	✓	Yes	Yes
p3	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓	✓	✓	✓	✓	✓	✓	Yes	Yes

*1 : Timer current value
*2 : Counter current value
Note: See Section 1.17, "Devices Available As Instruction Parameters."

Table 3.13.11 Devices Available for Output of Input Macro

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓	✓	✓	✓	✓	✓	✓	Yes	Yes

*1 : Timer current value
*2 : Counter current value
Note: See Section 1.17, "Devices Available As Instruction Parameters."

CAUTION

- You may not pass time-count relays (T) or end-of-count relays (C) directly as parameters to a macro instruction. Instead, copy the relay value to an internal relay (I) and then pass the internal relay as a parameter.
- You may not pass a long-word constant and floating-point constant directly as a parameter to a macro instruction. Instead, assign the long word value to a data register (D), and then pass the register as a parameter.

■ Function

● Input Macro Instruction Call (NCALL)

Executing this instruction executes the registered macro instruction.

When the input macro instruction completes execution, control is returned to the step immediately following the Input Macro Instruction Call.

The NCALL instruction differs from the Macro Call (MCALL) instruction in that an NCALL instruction can be coded at positions for input instructions (such as Load, Compare) to call and execute a macro instruction. Using NCALL together with the Output of Input Macro (NMOUT) instruction allows you to output the logical operation result to the instruction following the NCALL instruction.

A macro instruction called using an NCALL instruction is also known as an input macro instruction.

TIP

Up to three parameters may be passed to an input macro instruction directly. To pass more than three parameters to an input macro instruction, use the Parameter instruction (PARA).

● Output of Input Macro (NMOUT)

Specifies the logical operation result of an input macro instruction. The logical operation result sent as output to the instruction immediately following the Input Macro Instruction Call depends on the status of the source device contained in this instruction.

Table 3.13.12 Output of Input Macro (NMOUT)

Input Parameter Devices	Logical Operation Results of Input Macro (Device status = output)
Constants	0 = off, non-zero = on
Relay devices	0 = off, 1 = on
Register devices	0 = off, non-zero = on

If NMOUT is executed more than once, the last execution takes precedence.

If NMOUT is not executed, the logical operation result of the input macro is off.

CAUTION

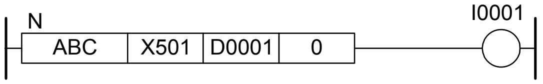
The NMOUT instruction is only valid in an input macro invoked using the NCALL instruction.

An NMOUT instruction executed in a macro invoked using the MCALL instruction is ignored.

■ Programming Example

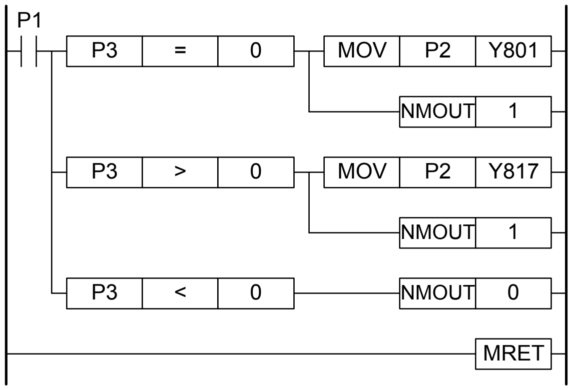
The sample code below calls input macro "ABC," and turns on I0001 if the logical operation result of the input macro is on.

In this example, I0001 turns on if P1 (X501) is on and P3 (D0002) is 0 or positive, and turns off if P3 is negative.



F031326.VSD

Figure 3.13.10 Example Input Macro Program (on the program block side)



F031327.VSD

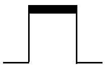
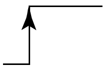
Figure 3.13.11 Example of an Input Macro Program (on the macro side)

3.14 Indirect Specification Instructions

3.14.1 Indirect Address Set (SET@)

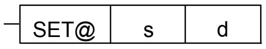
F3SP22-0S
F3SP28-3S
F3SP38-6S | F3SP53-4S
F3SP58-6S
F3SP59-7S | F3SP66
F3SP67 | F3SP71
F3SP76

Table 3.14.1 Indirect Address Set

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	510	Indirect Address Set	SET@	SET@	✓	–		3	3 words	–
	510P		↑ SET@	↑ SET@				4		

■ Parameter

Indirect Address Set



F031401.VSD

s : Number of device to be converted
d : First device number for storing the address after conversion. (always prefixed with @))

■ Available Devices

Table 3.14.2 Devices Available for Indirect Address Set

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s	✓	✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes
d									✓	✓	✓*1						No	–*4

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value
*3: Counter current value
*4: input is in indirect specification representation, but involves no indirect specification.

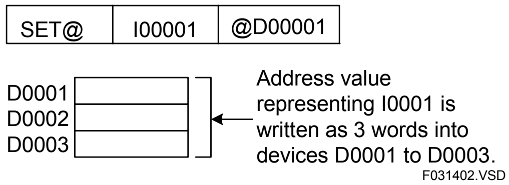
■ Function

Stores the value representing the address of device s in three words starting with register designated by s.

Indirect specification is not allowed for parameters of Indirect Address Set instructions. Although device d is represented with indirect specification, the direct address value is written to the device.

If device s is modified by an index, then the address after index modification will be stored in the device.

If a timer or counter is specified, the address for its current value is stored.

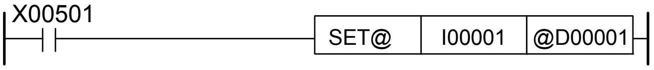

SET@
I00001
@D00001
D0001
D0002
D0003
Address value representing I0001 is written as 3 words into devices D0001 to D0003.

F031402.VSD

Figure 3.14.1 Indirect Address Set

■ Programming Example

Stores the address of I00001 in D00001, D00002, and D00003 when X00501 turns on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	SET@	I0001	@D0001			

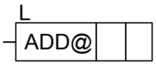
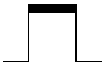
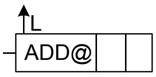
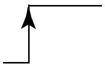
F031403.VSD

Figure 3.14.2 Example of an Indirect Address Set Program

3.14.2 Indirect Address Add (ADD@)

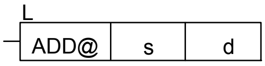
F3SP22-0S F3SP28-3S F3SP38-6S F3SP53-4S F3SP58-6S F3SP59-7S F3SP66 F3SP67 F3SP71 F3SP76

Table 3.14.3 Indirect Address Add

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	511	Indirect Address Add	ADD@		✓	—		3	2 words	—
	511P		↑ ADD@					4		

■ Parameter

Indirect Address Add



F031404.VSD

s : First address of storage area for the indirect address (always prefixed with @)
d : Data, or device number, representing a value to be added

■ Available Devices

Table 3.14.4 Devices Available for Indirect Address Add

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s									✓	✓*1	✓*1						No	—*2
d	✓	✓	✓	✓*1	✓*1	✓*1			✓	✓*1	✓*1	✓*1	✓*1	✓	✓	✓	Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Input is in indirect specification representation, but involves no indirect specification.

■ Function

Adds the (signed) long-word value designated by device d to the address stored in device s and stores the resultant address in 3 words starting with device s.

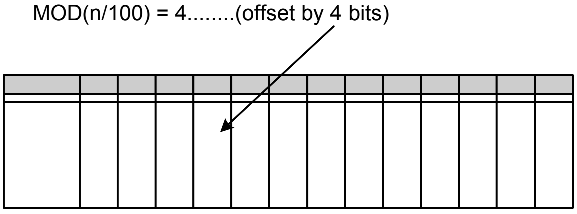
Although device s is represented with indirect specification, addition is performed on the device itself.

Setting device d to a value n adds n to the indirect specified device address.

To perform address subtraction, assign a negative value to device d and perform addition.

If the address designated by device s is an input or output relay, the value of device d is converted to a slot-based value before addition. A negative device d value is in case causes an error.

If @D0001 = X00201 and n=204, the address designated by @D0001 after addition will be X00405, as shown in the following computation.
INT(n/100) = 2..........(offset by 2 slots)
MOD(n/100) = 4........(offset by 4 bits)

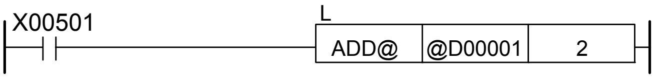


F031405.VSD

Figure 3.14.3 Indirect Address Add for Input/Output Relays

■ Programming Example

The sample code below adds 2 to the address stored in three words starting with D00001 when X00501 turns on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	ADD@	@D0001	2			

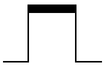
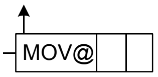
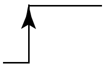
F031406.VSD

Figure 3.14.4 Example of an Indirect Address Add Program

3.14.3 Indirect Address Move (MOV@)

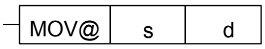
F3SP22-0S F3SP28-3S F3SP38-6S | F3SP53-4S F3SP58-6S F3SP59-7S | F3SP66 F3SP67 | F3SP71 F3SP76

Table 3.14.5 Indirect Address Move

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	512	Indirect Address Move	MOV@	MOV@	✓	—		3	3 words	—
	512P		↑MOV@	 MOV@				4		

■ Parameter

Indirect Address Move



F031407.VSD

s : First device number for transfer source data (always prefixed with @)
d : First device number for destination (always prefixed with @)

■ Available Devices

Table 3.14.6 Devices Available for Indirect Address Move

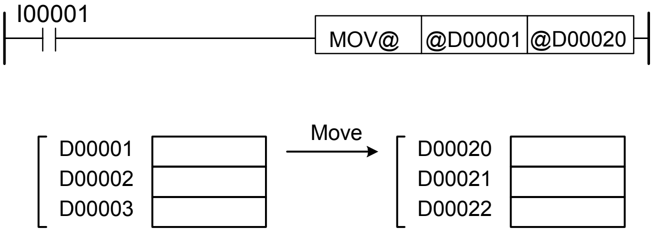
Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s									✓	✓	✓						No	—*1
d									✓	✓	✓						No	—*1

Note: See Section 1.17, "Devices Available As Instruction Parameters."
*1: Input is in indirect specification representation, but involves no indirect specification.

■ Function

Transfers the indirect address stored in three words starting with device s to three words starting with device d.

Use this instruction to transfer indirect addresses.

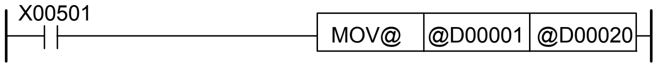


F031408.VSD

Figure 3.14.5 Indirect Address Move

■ Programming Example

The sample code below transfers the address stored in three words starting with D00001 to three words starting with D00020 when X00501 is ON.



Line No.	Instruction	Operands			
0001	LD	X00501			
0002	MOV@	@D00001	@D00020		

F031409.VSD

Figure 3.14.6 Example of an Indirect Address Move Program

3.15 Disk Operation Instructions

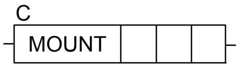
Of the file operation instruction group and disk operation instruction group, only instructions from one group can be executed at any one time. If you attempt to execute multiple instructions, the instruction that is executed later will be terminated with a redundant use of function error (error code -3001). Before executing a disk operation instruction, check to ensure that the File/Disk Operation Group Busy relay (M1025) is OFF.

3.15.1 Mount Memory Card (MOUNT)

F3SP66 F3SP71
F3SP67 F3SP76

Mounts the memory card inserted in the card slot so that it is ready for use by programs and various services.

Table 3.15.1 Mount Memory Card

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Mount Memory Card	MOUNT		✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

Mount Memory Card

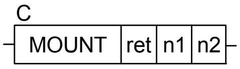


Table 3.15.2 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1 to 32767 (x 100 ms)]
n2	Card slot number (W) [always 1]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.15.3 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.15.4 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.15.5 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1025	File/Disk Operation Group Busy	Execute instruction only if the "File/Disk Operation Group Busy" relay is OFF.

■ Function

Mounts the memory card inserted in the card slot so that it is ready for use by programs and various services. It is usually not necessary to execute this instruction as the CPU module automatically recognizes and mounts a memory card when it is inserted into the memory card slot. This instruction can be used however in situations where there is a need to mount and unmount a memory card which remains inserted in the memory card slot. A possible scenario would be to unmount the memory card in the day but to mount and access the memory card in the night.

If the memory card is successfully mounted with normal exit, the **SD** LED located on the front panel of the module lights up. Conversely, the **SD** LED is not lit if the memory card is unmounted.

Always unmount the memory card either by executing the Unmount Memory card (UNMOUNT) instruction or using the rotary switch before removing the memory card from the memory card slot.

The card slot number is fixed to 1.

CAUTION

- Inserting a memory card automatically mounts it without the need to execute this instruction.
- Do not remove a memory card without first unmounting it. Otherwise, data may be damaged or lost.
- Execution of this instruction is highly likely to complete even in the presence of a timeout or cancel event.

■ Programming Example

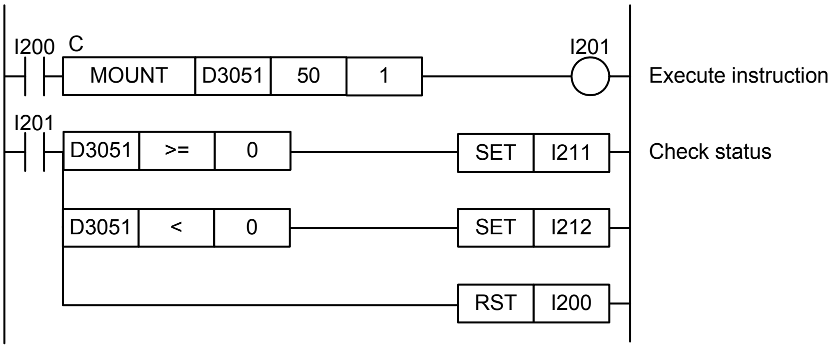


Figure 3.15.1 Example of a Mount Memory Card Program

This sample code mounts memory card CARD1. It specifies timeout interval as 50 (5 s).

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

3.15.2 Unmount Memory Card (UNMOUNT)

F3SP66 F3SP71
F3SP67 F3SP76

Unmounts the memory card, which is inserted and mounted in the card slot.

Table 3.15.6 Unmount Memory Card

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Unmount Memory Card	UNMOUNT	C UNMOUNT	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

Unmount Memory Card C UNMOUNT ret n1 n2

Table 3.15.7 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1 to 32767 (x 100 ms)]
n2	Card slot number (W) [always 1]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.15.8 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.15.9 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

Resource Relays

Table 3.15.10 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1025	File/Disk Operation Group Busy	Execute instruction only if the "File/Disk Operation Group Busy" relay is OFF.

Function

Unmounts the memory card, which is inserted and mounted in the card slot. A memory card in unmounted state can be safely removed, but does not allow access by programs or via FTP.

If the memory card is successfully unmounted with normal exit, the **SD** LED located on the front panel of the module turns off. Conversely, the **SD** LED is lit if the memory card is mounted.

The card slot number is fixed to 1.

CAUTION

- Do not remove a memory card without first unmounting it. Otherwise, data may be damaged or lost.
- Execution of this instruction is highly likely to complete even in the presence of a timeout or cancel event.

Programming Example

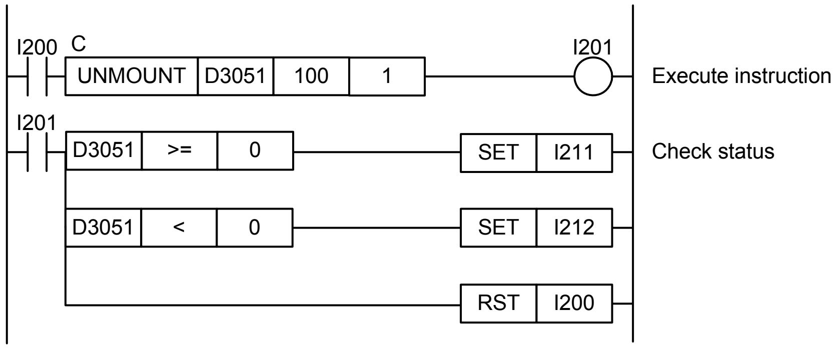


Figure 3.15.2 Example of an Unmount Memory Card Program

This sample code unmounts memory card CARD1. It specifies timeout interval as 100 (10 s).

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

3.15.3 Format Disk (FORMAT)

F3SP66 F3SP71
F3SP67 F3SP76

Formats a specified disk.

Table 3.15.11 Format Disk

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Format Disk	FORMAT	C ┤FORMAT │ │ ├	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

```
            C
Format Disk ┤FORMAT│ret│ n ├
```

Table 3.15.12 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Disk selection (W) [1=\RAMDISK 2=\CARD1]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.15.13 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.15.14 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, " Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.15.15 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1025	File/Disk Operation Group Busy	Execute instruction only if the "File/Disk Operation Group Busy" relay is OFF.

■ Function

Formats a specified disk.

If the disk to be formatted is in use, formatting cannot be done. Close all files before executing this instruction.

An SD memory card, which is not in a supported format, cannot be mounted even if it is inserted. In this case, this instruction can be executed to format the memory card to a supported format.

Talbe 3.15.16 Supported Formats

CPU	Memory Type	Memory Capacity	Format
F3SP71-4N F3SP76-7N F3SP71-4S F3SP76-7S	SD Memory Card	Up to 2GB	FAT16
	SDHC Memory Card	4GB to 32GB	FAT32
F3SP66-4S F3SP67-6S	SD Memory Card	Up to 1GB	FAT16
	SDHC Memory Card	Not supported	Not supported

CAUTION

- This instruction removes all information on the specified disk so be careful when executing the instruction.
- A disk cannot be accessed while formatting in progress.
- This instruction does not have a timeout interval parameter because formatting takes quite a while. Once executed, the instruction ignores cancel requests, if any.

■ Programming Example

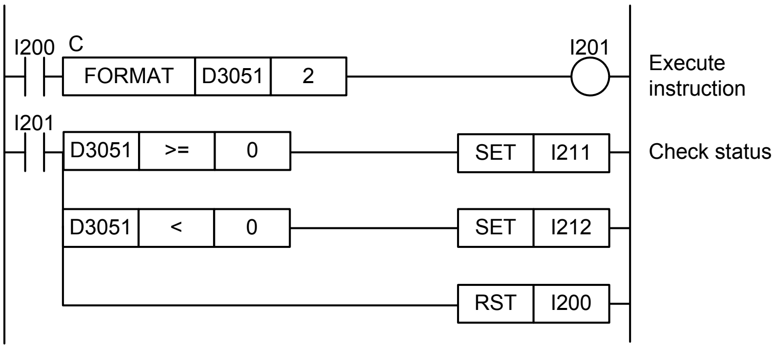


Figure 3.15.3 Example of a Format Disk Program

This sample code formats memory card CARD1.

The table below shows an example of the returned status data (ret), assuming normal exit.

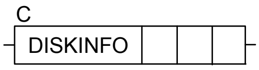
Device	Value	Table Parameter
ret=D3051	0	Status

3.15.4 Disk Info (DISKINFO)

F3SP66 F3SP71
F3SP67 F3SP76

Gets information on free space and capacity of a specified disk.

Table 3.15.17 Disk Info

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Disk Info	DISKINFO		✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

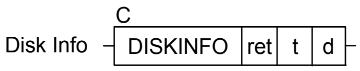


Table 3.15.18 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	Disk selection (W) [1=\RAMDISK 2=\CARD1 3=\CARD1(SDHC)[*2]]
d[*3]		Device for storing disk information (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".
*2: Select when using a SD memory card from 4GB to 32GB for SP71/76.
*3: d (device for storing disk information) is table data. For details, see Tables 3.15.22 and 3.15.23, "Returned Information".

■ Status (Return Value)

Table 3.15.19 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.15.20 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes
d									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.15.21 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1025	File/Disk Operation Group Busy	Execute instruction only if the "File/Disk Operation Group Busy" relay is OFF.

■ Function

Gets the following information about a specified disk.

Table 3.15.22 Returned Information (When \RAMDISK or \CARD1 is specified)

Information		Offset (word)	Size (word)	Description
Free space	Low word	+0	2	Free space [0 to 4294967295 (bytes)]
	High word	+1		
Capacity	Low word	+2	2	Capacity [0 to 4294967295 (bytes)]
	High word	+3		

Table 3.15.23 Returned Information (When \CARD1 (SDHC) is specified)*1

Information		Offset (word)	Size (word)	Description
Free space	Low word	+0	4	Free space [0 to 18446744073709551615 (bytes)]
		+1		
	High word	+2		
		+3		
Capacity	Low word	+0	4	Capacity [0 to 18446744073709551615 (bytes)]
		+1		
	High word	+2		
		+3		

*1: The maximum capacity of returned information is equal to the capacity of a SDHC (4GB to 32GB).

CAUTION

- Handle the returned information as unsigned 32-bit or 64-bit data.
- When \RAMDISK or \CARD1 is specified, there is an upper limit on the capacity of returned information. The upper limit value is 4294967295 bytes (18446744073709551615 bytes for SDHC).

■ Programming Example

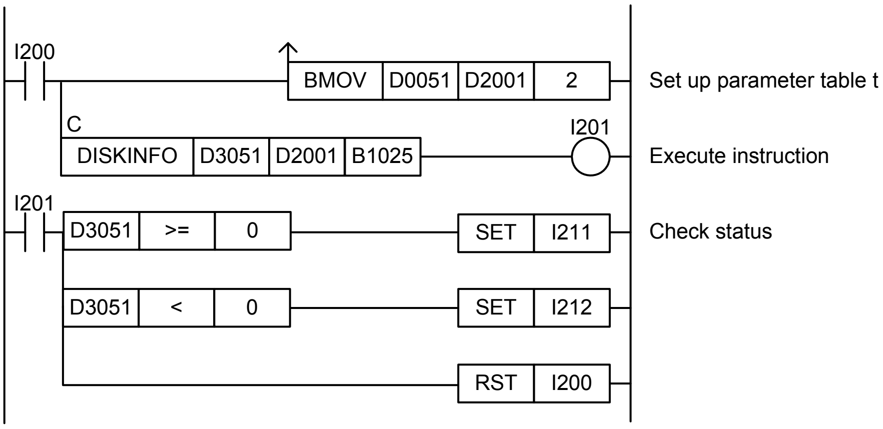


Figure 3.15.4 Example of a Disk Info Program

This sample code gets and stores disk information about memory card CARD1 to device, starting from B1025.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	50	Timeout interval (= 5 s)
D2002	2	Disk selection (=CARD1)

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

The table below shows an example of the data stored to d (B1025) for a 1 gigabyte memory card.

Device	Value	Table Parameter
d=B1025	990019584	Free space
B1026		
B1027	990642176	Capacity
B1028		

3.16 File Access Instructions

You use an FOPEN instruction to get access right to a file in the form of a file ID, which can then be used to read from and write to the file using FREAD, FWRITE and other file access instructions. Finally, you use an FCLOSE instruction to release the file ID.

Table 3.16.1 Terminology Description for File Access Instructions

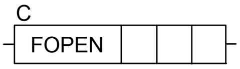
Term	Description
Binary file	A file containing binary data, with no delimiters.
CSV formatted file	A text file in which ASCII coded data elements are delimited by comma (,) characters or TAB characters. A CSV file can be displayed directly in Excel. Conversely, an Excel file can be converted to a CSV formatted file with some limitations. A newline is also considered as a delimiter. Beware that if a newline and a contiguous delimiter character is treated as one field.
Field	A field is one data element in a CSV formatted file.
Record	A record (one line) in a CSV formatted file is delimited by a newline code. One record contains 1 to n fields.

3.16.1 Open File (FOPEN)

F3SP66 F3SP71
F3SP67 F3SP76

Opens a specified file according to the specified open mode so that it is ready for use. At the same time, secures exclusive access control to the file against access by other instructions.

Table 3.16.2 Open File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Open File	FOPEN		✓	–	6	16 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

Open File

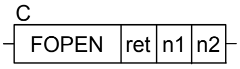


Table 3.16.3 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
n2	Open mode (W) [0 = Read-only mode (read only) 1 = Write mode[*2] (read and write) 2 = Append mode (read and write)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)"
*2: When an existing file is opened in write mode, the contents of the file are deleted.

Table 3.16.4 Text Parameter

Parameter		Description
1	n3	Pathname of file to be opened

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Subsection 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.16.5 Status (Return Value)

Offset (word)		Description	
ret	ret+0	≥ 0	File ID (W) [0-15] (File is opened successfully)
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.16.6 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.16.7 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1026	No Unused File ID	Execute FOPEN instruction if "No Unused File ID" is OFF.
	M1041 to M1056	File ID Open	
	M1057 to M1072	File ID Busy	

■ Function

Opens a specified file according to the specified open mode so that it is ready for use. At the same time, secures exclusive access control to the file against access by other instructions.

If the file is opened successfully, the instruction returns a file ID as return value. This file ID can be used subsequently in file access instructions to access the open file.

There are three types of file open mode, which are described in the table below. Specify the appropriate open mode as required. Files are always opened in binary mode.

Table 3.16.8 Open Modes

Open Mode	Mode No.	Description	Initial File Pointer Position
Read-only mode	0	The read-only mode grants access right for reading a file. Multiple file IDs of read-only mode can be allocated for the same file. File ID of read-only mode can be allocated even for a file that has been opened in write or append mode.	0
Write mode	1	The write mode grants access right for reading and writing a file. When an existing file is opened in write mode, the module deletes the file and creates a new file. For purpose of exclusive control, only one file ID of write mode is allowed for a file at any one time.	End of file
Append mode	2	The append mode grants access right for appending data to an existing file. If the file does not exist, a new file is created. For purpose of exclusive control, only one file ID of append mode is allowed for a file at any one time.	End of file

CAUTION

- Up to 16 files can be opened concurrently. This maximum limit may be reduced by file operations executed via FTP or other interfaces.
- Do not access the status (return value, offset: +0) during instruction execution processing as it is used by the system.

■ Programming Example

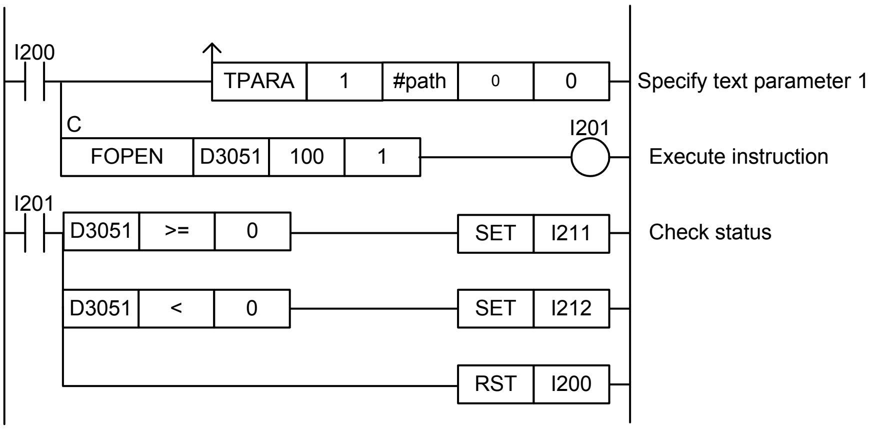


Figure 3.16.1 Example of an Open File Program

This sample code opens "\RAMDISK\myfile.txt" (=#path) in write mode. It specifies D3051 as the device for storing the return status, and specifies the timeout interval as 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

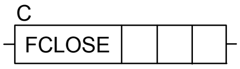
Device	Value	Table Parameter Name
ret=D3051	2	Status (file ID)

3.16.2 Close File (FCLOSE)

F3SP66 F3SP67	F3SP71 F3SP76

Closes a file opened with a specified file ID, and releases access right to the file.

Table 3.16.9 Close File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Close File	FCLOSE		✓	–	6	16 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

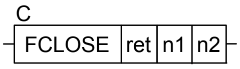
Close File 

Table 3.16.10 Parameter

Parameter	Description
ret*1	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
n2	File ID (W) [0-15]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.16.11 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.16.12 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.16.13 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Closes a file opened with a specified file ID, and releases access right to the file.

■ Programming Example

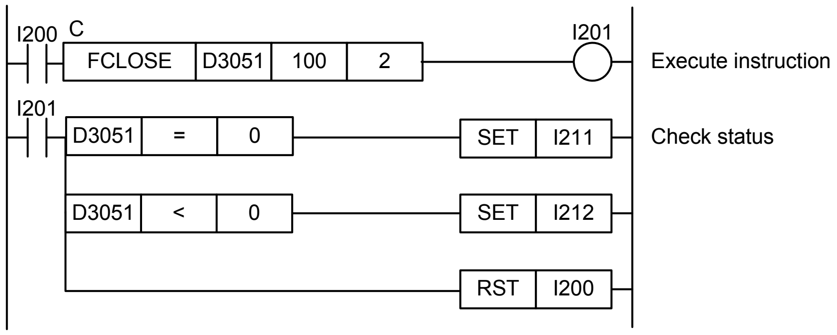


Figure 3.16.2 Example of a Close File Program

This sample code closes the open file designated by file ID 2. It specifies D3051 as the device for storing the return status, and the timeout interval as 10s.

The table below shows the returned status data, assuming normal exit.

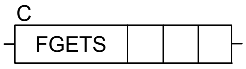
Device	Value	Table Parameter Name
ret=D3051	0	Status

3.16.3 Read File Line (FGETS)

F3SP66 F3SP71
F3SP67 F3SP76

Reads one line (up to a newline) from the file associated with a specified file ID.

Table 3.16.14 Read File Line

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Read File Line	FGETS		✓	–	6	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

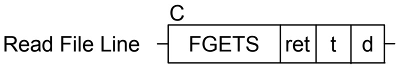


Table 3.16.15 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2	Newline option (W) [0=CRLF 1=LF]
	t+3	Delete Newline option (W) [0=No 1=Yes]
	t+4	Append NULL option (W) [0 = No 1 = Yes]
	t+5	Read size limit (W) [1-128 words][*2]
d		First device for storing string (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".
*2: Limit includes appended NULL and newline bytes.

■ Status (Return Value)

Table 3.16.16 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	No. of bytes read (W) [1-256][*1]	
	ret+2	No. of words written to device (W) [1-128][*1*2]	

*1: Excluding appended NULL, if any.
*2: Rounded up for odd number of bytes.

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.16.17 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes
d									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.16.18 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Reads one line (up to a newline) from the file designated by a specified file ID. It includes an option to delete the newline code from read data.

It also includes an option to append a NULL byte at the end of the read string. When the option is selected, a NULL byte is appended and stored to device.

Up to 256 bytes (128 words) are allowed in one line. This length limit includes appended NULL and newline bytes.

TIP

The file pointer movement is as follows:

- Start position for reading = current file pointer position
- File pointer position after instruction execution = the byte following the last byte that was read

■ Programming Example

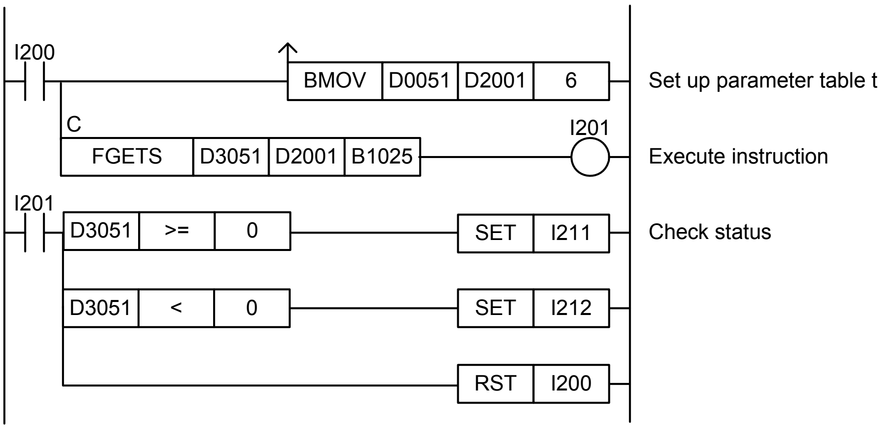


Figure 3.16.3 Example of a Read File Line Program

This sample code reads one line of data from the open file associated with file ID 2. It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as follows.

Device	Value	Table Parameter
t=D2001	10	Timeout interval (= 1 s)
D2002	2	File ID (= 2)
D2003	0	Newline option (= CRLF)
D2004	1	Delete newline option (= Yes)
D2005	1	Append NULL option (= Yes)
D2006	128	Read size limit (= 128 words)

Assuming normal exit and 50 bytes are read, the following data is stored in ret.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	50	No. of bytes read
D3053	25	No. of words written to device

3.16.4 Write File Line (FPUTS)

F3SP66 F3SP71
F3SP67 F3SP76

Writes one line of text to the file associated with a specified file ID.

Table 3.16.19 Write File Line

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Write File Line	FPUTS	C FPUTS	✓	–	5	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

```
                  C
Write File Line -| FPUTS | ret | t |-
```

Table 3.16.20 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2	Append newline option (W) [0 = None 1 = Append CRLF 2 = Append LF]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.16.21 Text Parameter

Parameter		Description
1	s	Write string

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Subsection 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.16.22 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	No. of bytes written to file (W) [1-258][*1]	

*1: Including appended newline bytes.

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.16.23 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.16.24 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Writes one line of text to the file associated with a specified file ID. Writes characters starting from the specified device up to a NULL character as one line of text to the file. The NULL at the end is not written to the file.

It includes options for appending a newline.

Up to 256 bytes can be written as a line.

TIP

The file pointer movement is as follows:

- Start position for writing = current file pointer position
- File pointer position after instruction execution = the byte following the last written byte

■ Programming Example

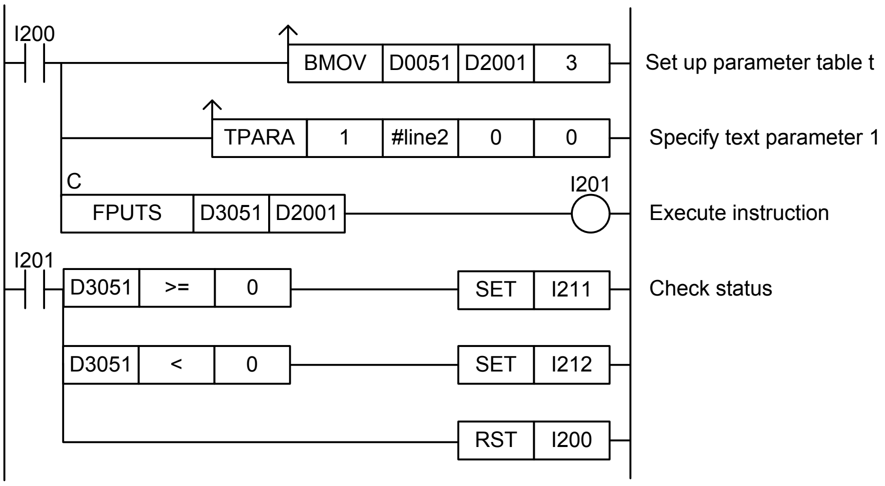


Figure 3.16.4 Example of a Write File Line Program

This sample code writes the string defined by constant #line2 into a file opened as file ID 5.

It specifies ret(=D3051) and t(=D2001), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	10	Timeout interval (= 1 s)
D2002	5	File ID (= 5)
D2003	1	Append newline option (= Append CRLF)

The table below shows the content of ret, assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	34	No. of bytes written to file

3.16.5 Read File Block (FREAD)

F3SP66 F3SP71
F3SP67 F3SP76

Reads [block size] x [No. of blocks] bytes from the file associated with a specified file ID.

Table 3.16.25 Read File Block

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Read File Block	FREAD	C FREAD	✓	–	6	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

Read File Block C FREAD | ret | t | d

Table 3.16.26 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2	Block size low word (L) [1-524288 (byte)]
	t+3	(Block size high word)
	t+4	No. of blocks low word (L) [0-262144]
	t+5	(No. of blocks high word)
	t+6	Read size limit low word (L) [1-262144 (words)]
	t+7	(Read size limit high word)
d		First device for storing read string (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.16.27 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	No. of blocks read low word (L) [1-262144]	
	ret+2	(No. of blocks read high word)	

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.16.28 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes
d									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.16.29 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Reads [block size] x [No. of blocks] bytes from the file associated with a specified file ID. You can specify the number of text blocks to be read.

If an odd number is specified for block size and multiple blocks are read from the file, the blocks will be stored to device with a one-byte gap between blocks. Each block is stored to device, beginning on a word boundary.

If the data read from the file is less than one block size, a block size error (error code -2004) will be generated, and that block will be not included in the number of blocks read stored to status.

TIP

The file pointer movement is as follows:

- Start position for reading = current file pointer position
- File pointer position after instruction execution = the byte following the last byte that was read

CAUTION

Pay attention to device boundary as this instruction may transfer a large amount of data.

■ Programming Example

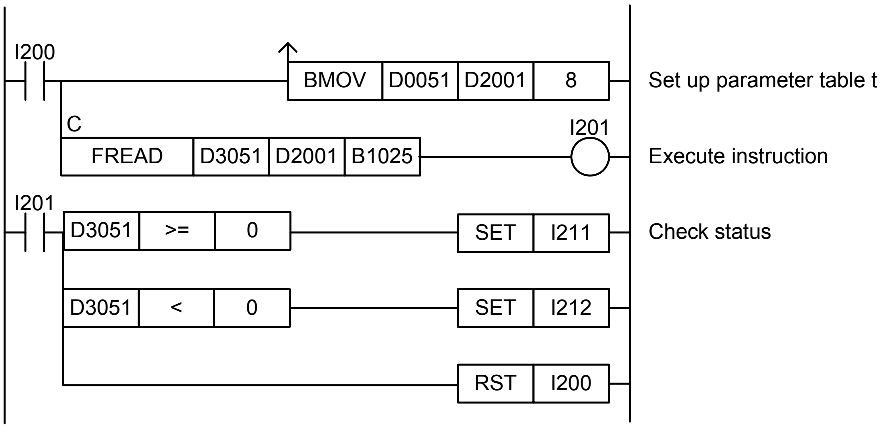


Figure 3.16.5 Example of a Read File Block Program

This sample code reads data blocks from the open file associated with file ID 3.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	10	Timeout interval (= 1 s)
D2002	3	File ID (=3)
D2003	512	Block size (= 512 bytes)
D2004		
D2005	10	No. of blocks (= 10)
D2006		
D2007	10000	Read size limit (= 10000 words)
D2008		

The table below shows an example of the returned status data (ret), assuming normal exit.

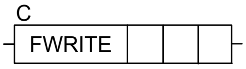
Device	Value	Table Parameter
ret=D3051	0	Status
D3052	10	No. of blocks read
D3053		

3.16.6 Write File Block (FWRITE)

F3SP66 F3SP71
F3SP67 F3SP76

Write [block size] x [No. of blocks] bytes into the file associated with a specified file ID.

Table 3.16.30 Write File Block

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Write File Block	FWRITE		✓	–	6	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

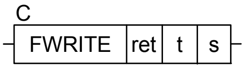
Write File Block


Table 3.16.31 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2	Block size low word (L) [1-524288(bytes)]
	t+3	(Block size high word)
	t+4	No. of blocks low word (L) [0-262144]
	t+5	(No. of blocks high word)
s		First device storing write data

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.16.32 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	No. of written blocks low word (L) [1 to 262144]	
	ret+2	(No. of written blocks high word)	

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.16.33 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes
s									✓	✓	✓	✓		✓	✓	#	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."
#: Only a defined constant can be specified. Specifying a normal constant is not allowed.

■ Resource Relays

Table 3.16.34 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Writes [block size] x [No. of blocks] bytes to the file associated with a specified file ID. You can specify the number of text blocks to be written.

If an odd number is specified for block size and multiple blocks are written to the file, each data block will begin on a word boundary.

TIP

The file pointer movement is as follows:

- Start position for writing = current file pointer position
- File pointer position after instruction execution = the byte following the last written byte

CAUTION

Pay attention to device boundary as this instruction may transfer a large amount of data.

■ Programming Example

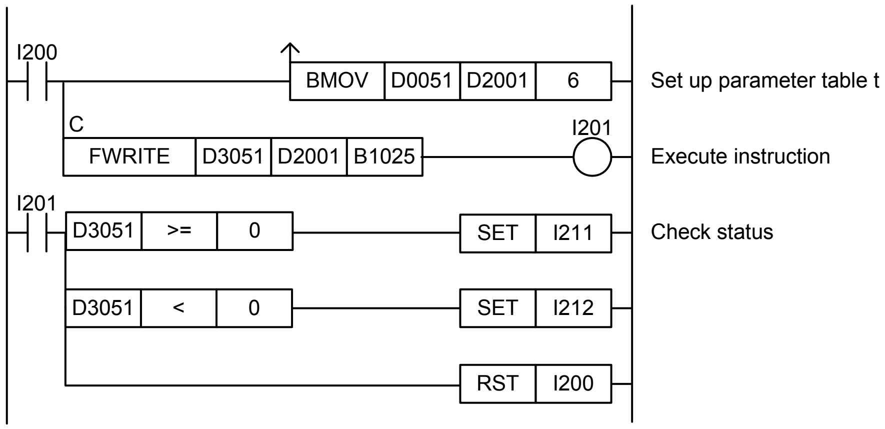


Figure 3.16.6 Example of a Write File Block Program

This sample code writes data blocks stored in device, beginning at B1025 to the file associated with file ID 2.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	10	Timeout interval (= 1 s)
D2002	2	File ID (=2)
D2003	128	Block size (= 128 bytes)
D2004		
D2005	5	No. of blocks (= 5)
D2006		

The table below shows an example of the returned status data (ret), assuming normal exit.

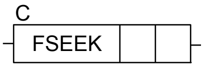
Device	Value	Table Parameter
ret=D3051	0	Status
D3052	5	No. of written blocks
D3053		

3.16.7 File Seek (FSEEK)

F3SP66 F3SP71
F3SP67 F3SP76

Moves the file pointer of the file associated with a specified file ID.

Table 3.16.35 File Seek

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	File Seek	FSEEK		✓	–	5	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

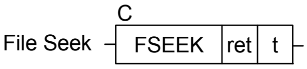


Table 3.16.36 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2	Origin (W) [0-4] [*2]
	t+3	Offset low word (L) [-2147483648 to 2147483647 (bytes)][*3] [0 to 4294967295 (bytes)][*4]
	t+4	(Offset high word)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

*2: "0 to 2" and "0,2,3,4" are available for F3SP66/67 and F3SP71/76 respectively.
Use "3, 4" for a file larger than 2GB but not exceeding 4GB

*3: This is the range of offset values available for a file of 2GB or smaller (signed number of bytes).

*4: This is the range of offset values available for a file larger than 2GB but not exceeding 4GB (unsigned number of bytes).

■ Status (Return Value)

Table 3.16.37 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	File pointer offset from file start after execution (low word) (L) [1-2147483647][*1] [1-4294967295][*2]	
	ret+2	(File pointer offset from file start after execution (high word))	

*1: This is the offset values after move for a file of 2GB or smaller.

*2: This is the offset values after move for a file larger than 2GB but not exceeding 4GB.

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.16.38 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.16.39 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Moves the file pointer of the file associated with a specified file ID. A file pointer marks the beginning position for reading and writing.

The file pointer destination is specified as an origin and an offset from the origin. The offset is specified as a signed number of bytes for a file of 2GB or smaller, and unsigned number of bytes for a file larger than 2GB but not exceeding 4GB.

Table 3.16.40 Origin

Description	Value of Origin
File start	0
Current position *1	1
File end	2
Current position *2 (When moving toward the file start)	3
Current position *3 (When moving toward the file end)	4

*1: Select this when moving the file pointer of a file of 2GB or smaller from the current position toward the file start or end (available offset range: -2147483648 to 2147483647)

*2: Select this when moving the file pointer of a file larger than 2GB but not exceeding 4GB from the current position toward the file start (available offset range: 0 to 4294967295).

*3: Select this when moving the file pointer of a file larger than 2GB but not exceeding 4GB from the current position toward the file end (available offset range: 0 to 4294967295).

To determine the current file pointer position, execute the instruction with origin=1 and offset=0 and check the return value.

To determine the file size (file end position), execute the instruction with origin=2 and offset=0 and check the return value.

An attempt to move the file pointer before file start will move the file pointer to file start. An attempt to move the file pointer after file end will move the file pointer to file end. Neither of the above two cases generate an error.

■ Programming Example

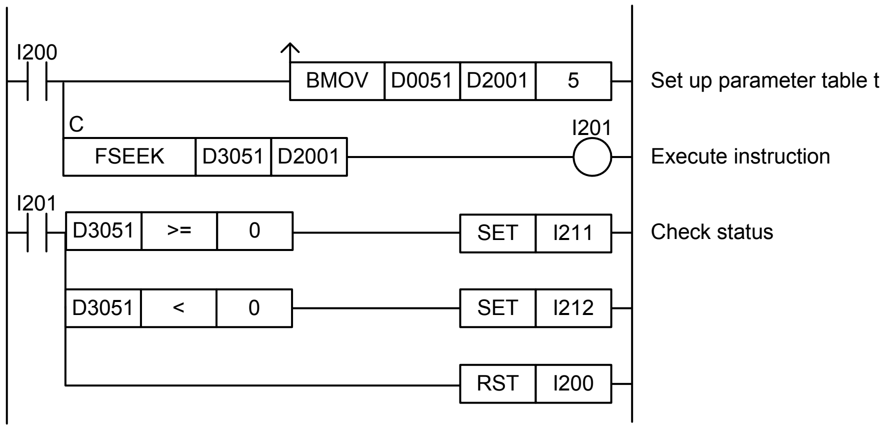


Figure 3.16.7 Example of a File Seek Program

This sample code moves the file pointer of the file opened as file ID 6 to a position, which is 500 bytes from the beginning of the file.

It specifies ret(=D3051) and t(=D2001), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	10	Timeout interval (= 1 s)
D2002	6	File ID (=6)
D2003	0	Origin (= file start)
D2004	500	Offset (= 500 bytes)
D2005		

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	500	File pointer offset from the specified origin after execution.
D3053		

3.16.8 File Text Search (FSEARCHT)

F3SP66 F3SP71
F3SP67 F3SP76

Searches for a user-specified string within the file associated with a specified file ID.

Table 3.16.41 File Text Search

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	File Text Search	FSEARCHT	C FSEARCHT	✓	–	5	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

File Text Search C FSEARCHT | ret | t

Table 3.16.42 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.16.43 Text Parameter

Parameter		Description
1	s	Search string

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Subsection 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.16.44 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1, ret+2	Match location [>0 : Final byte location of matching text from file start (L) -1 : No match found]	

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.16.45 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.16.46 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Searches for a user-specified string within the file associated with a specified file ID.

TIP

The file pointer movement is as follows:

- Starting position for search = current file pointer position
- File pointer position after instruction execution (if a match is found) = the byte following matched data
- File pointer position after instruction execution (if no match is found) = file pointer position before the search

CAUTION

If the instruction is cancelled or transition is made to Stop mode during execution and file size is large (megabytes to gigabytes), it may take some time (from several seconds to several minutes) for processing to be completed.

■ Programming Example

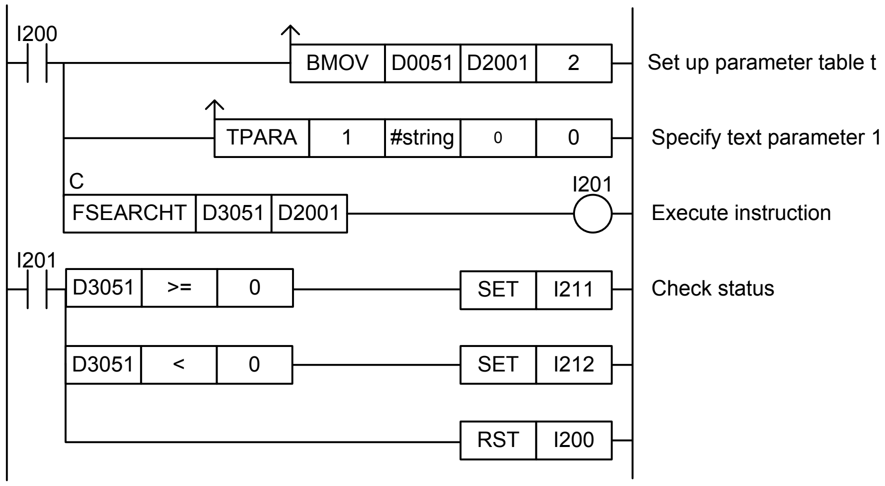


Figure 3.16.8 Example of a File Text Search Program

This sample code searches for the string defined by constant name #string within a file opened as file ID 2.

It specifies ret(=D3051) and t(=D2001), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	100	Timeout interval (= 10 s)
D2002	2	File ID (= 2)

The table below shows an example of the returned status (ret), assuming a match is found at byte offset 3044 from the beginning of the file.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	3044	Match location
D3053		

3.16.9 File Binary Search (FSEARCHB)

F3SP66 F3SP71
F3SP67 F3SP76

Searches for user-specified binary data within the file associated with a specified file ID.

Table 3.16.47 File Binary Search

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	File Binary Search	FSEARCHB	C FSEARCHB	✓	–	6	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

File Binary Search

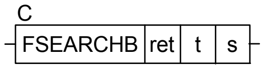


Table 3.16.48 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2	Binary search data size (W) [0-32767 (bytes)]
s		First device of binary search data (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.16.49 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1, ret+2	Match location [>0 : Final byte location of matching text from file start (L) -1 : No match found]	

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.16.50 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes
s									✓	✓	✓	✓		✓	✓	#	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."
#: Only a defined constant can be specified. Specifying a normal constant is not allowed.

■ Resource Relays

Table 3.16.51 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Searches for user-specified binary data within the file associated with a specified file ID.

TIP

The file pointer movement is as follows:

- Starting position for search = current file pointer position
- File pointer position after instruction execution (if a match is found) = the byte following matched data
- File pointer position after instruction execution (if no match is found) = file pointer position before the search

CAUTION

If the instruction is cancelled during execution and file size is large (megabytes to gigabytes), it may take some time (from several seconds to several minutes) for the transition to stop mode to be completed.

■ Programming Example

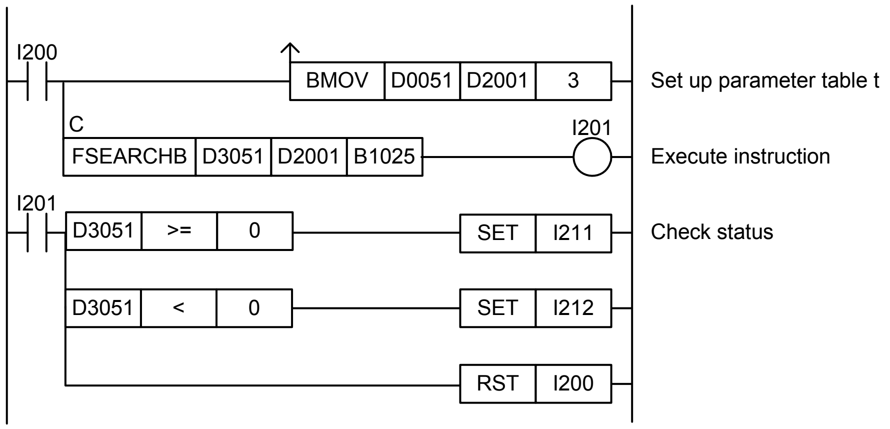


Figure 3.16.9 Example of a File Binary Search Program

This sample code searches for 20 bytes of device data starting at B1025 within the file opened as file ID 2.

It specifies ret(=D3051), t(=D2001) and s(=B1025), with t set up as follows.

Device	Value	Table Parameter
t=D2001	100	Timeout interval (= 10 s)
D2002	2	File ID (=2)
D2003	20	Size of search data (= 20)

The table below shows an example of the returned status (ret), assuming a match is found at byte offset 3044 from the beginning of the file.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	3044	Match location
D3053		

3.16.10 Convert CSV File to Device (F2DCSV)

F3SP66 F3SP71
F3SP67 F3SP76

Converts data in CSV formatted file to binary data and writes the data to contiguous devices.

Table 3.16.52 Convert CSV File to Device

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Convert CSV File to Device	F2DCSV	C F2DCSV	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

Convert CSV File to Device

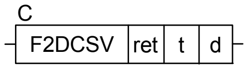


Table 3.16.53 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2, t+3	No. of fields to be read (L) [-1 = until file end 0 - 4194304 (if device unit = bit) 0 - 524288 (if device unit = byte) 0 - 262144 (if device unit = word) 0 - 131072 (if device unit = long word)]
	t+4	Field representation type (W) [0 = Decimal 1 = Hexadecimal 2 = Floating-point representation A ([-]d.dddd e[+/-]ddd form) 3 = Floating-point representation B ([-]dddd.dddd form)]
	t+5	Device unit (W) [0 = Bit 1 = Byte 2 = Word 3 = Long word]
	t+6	Sign extension[*2] (W) [0 = Pad with zeros 1 = Extend sign]
	t+7	Delimiter option (W) [0 = Comma (,) 1 = TAB]
	t+8	Newline option [0 = CRLF 1 = LF]
	t+9, t+10	Write limit in words (L) [1-262144 (words)]
d		First device for writing (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

*2: If the field representation type is set to hexadecimal, the sign extension setting should be specified.

■ Status (Return Value)

Table 3.16.54 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	No. of fields read (low word) (L) [0-2147483647 (fields)]	
	ret+2	(No. of fields read high word)	
	ret+3	No. of written words low word (L) [0-2147483647 (words)]	
	ret+4	(No. of written words high word)	

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.16.55 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes
d		✓	✓	✓	✓				✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.16.56 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

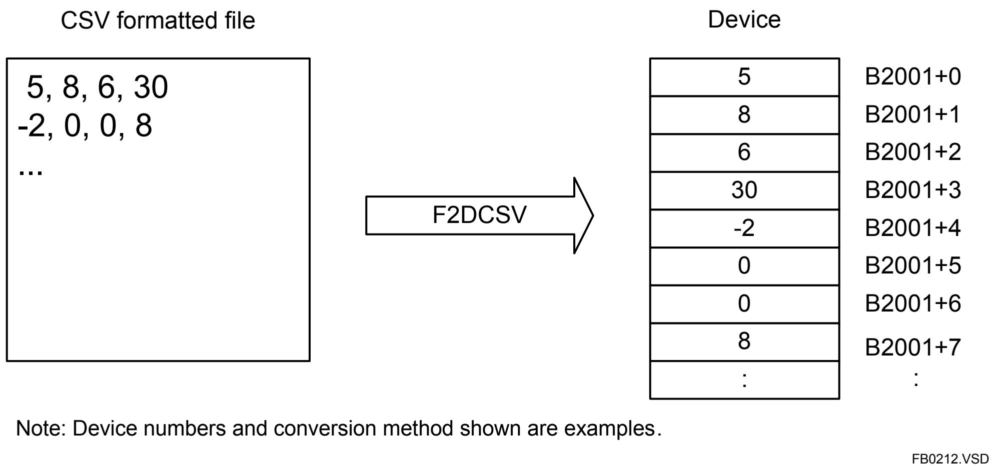
Converts data in CSV formatted file to binary data and writes the data to contiguous devices.

- Text in decimal, hexadecimal or floating-point representation can be converted to device data. Floating-point representation is converted to IEEE single-precision floating-point representation.

  Decimal ("-128" to "255", "-32768" to "65535", "-2147483648" to "4294967295")

  Hexadecimal ("0x0" to "0xFFFFFFFF", "0" to "FFFFFFFF")

  Floating-point ([-]d.dddd e[+/-]ddd, [-]dddd.dddd, Infinite "-INF"/"+INF")
- Available device unit options are bit, byte, word and long word. You can also specify whether to perform sign extension.
- Available field delimiter options are the comma (,) and Tab characters.
- Comments can be included in the file. If a field begins with a double-quote ("), single-quote ('), two slashes (//), or a slash and an asterisk (/*), the instruction skips over all characters until it encounters a delimiter character or newline.
- Newline can be specified as CRLF (standard for Windows) or LF.



Note: Device numbers and conversion method shown are examples.

Figure 3.16.10 CSV Formatted File to Device Conversion

TIP

Reading of File

- If end-of-file is encountered before the required number of fields is read, execution ends without error.
- A newline ends a record, and thus always ends a field.
- Within a field, any and all space characters preceding the data string are ignored but any space character following the data string results in a field conversion error.
- Double slashes (""//") and other comment mark characters must always be coded at the beginning of a field. Otherwise, a conversion error will be generated.
- If NULL or other invalid binary code is encountered, execution ends with a file interpretation error.

Conversion Error and Interpretation Error

- If a conversion error is detected, 0 is written to the device. If a conversion error is detected in a field during conversion, an error is generated but processing continues.
- When the converted numeric value of a field exceeds the range of the device unit, a conversion error is generated.
- Non-numeric representation "NaN" of D2FCSV generates a conversion error.

Data Conversion and Writing to Device

- You can specify to pad with '0's or extend the sign when the converted value of a field is smaller than the size of the device unit.
- If the device unit is specified as bit, 0 is stored for a zero value while 1 is stored for any other value.
- If you specify the field representation type as floating-point representation, you must specify the device unit as long word.
- Writing to device spans multiple scan cycles.

File Pointer Movement

- Start position for reading = current file pointer position
- File pointer position after instruction execution = the byte following the last byte that was read

CAUTION

Pay attention to the size of data read when specifying "until file end" as the number of fields to be read.

■ Programming Example

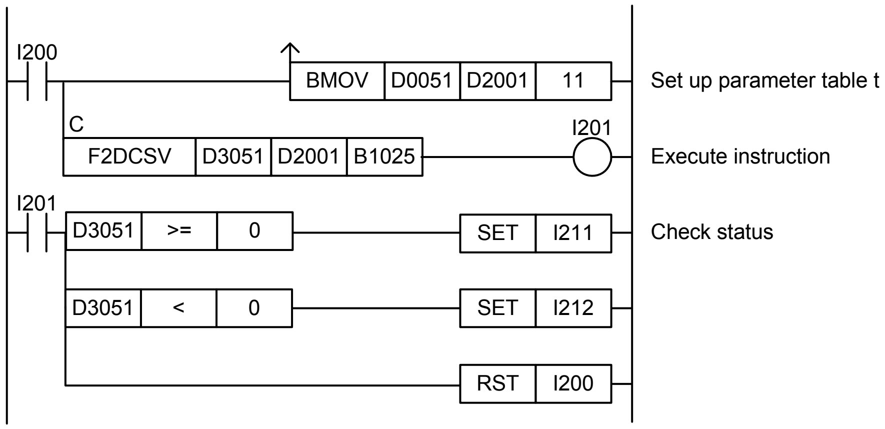


Figure 3.16.11 Example of a Convert CSV File to Device Program

This sample code reads the data contained in a CSV formatted file opened as file ID 4 according to the conditions set up in parameter table t.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	100	Timeout interval (= 10 s)
D2002	4	File ID (= 4)
D2003	2048	No. of fields to be read (= 2048 fields)
D2004		
D2005	0	Field representation type (= decimal)
D2006	2	Device unit (= word)
D2007	1	Sign extension (= Extend sign)
D2008	0	Delimiter option (= comma)
D2009	0	Newline option (= CRLF)
D2010	10000	Write limit in words (= 10000 words)
D2011		

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	2048	No. of fields read
D3053		
D3054	2048	No. of written words
D3055		

3.16.11 Convert Device to CSV File (D2FCSV)

F3SP66 F3SP71
F3SP67 F3SP76

Converts device data to text and outputs a CSV formatted file.

Table 3.16.57 Convert Device to CSV File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Convert Device to CSV File	D2FCSV	C –[D2FCSV | | |]–	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

Convert Device to CSV File C –[D2FCSV | ret | t | s]–

Table 3.16.58 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2	Device unit (W) [0 = bit 1 = byte 2 = word 3 = long word]
	t+3, t+4	No. of data units to be read (L) [0 - 4194304 (if device unit = bit) 0 - 524288 (if device unit = byte) 0 - 262144 (if device unit = word) 0 - 131072 (if device unit = long word)]
	t+5	Field representation type (W) [0 = Decimal 1 = Hexadecimal 2 = Floating-point representation A ([-]d.dddd e[+/-]ddd form) 3 = Floating-point representation B ([-]dddd.dddd form)]
	t+6	Field length (W) [0 = automatic (as required after conversion) 1-13 = fixed field length in characters]
	t+7	Field space handling (W) [[*2] 0 = Pad with spaces 1 = Pad with zeros]
	t+8	Delimiter option (W) [0 = comma (,) 1 =TAB]
	t+9	Newline option (W) [0 = CRLF 1 = LF]
	t+10	Newline insertion position (W) [0 = Do not insert newline 1 - 32767 = Insert newline after n fields]
s		First device for reading (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

*2: If the field length is specified as 0 (automatic), this parameter is ignored but a valid dummy value (say, 0) must still be specified.

■ Status (Return Value)

Table 3.16.59 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	No. of data units written to file (low word) (L) [0-4194304 (data count)]	
	ret+2	(No. of data units written to file (high word))	
	ret+3	No. of bytes written to file (low word) (L) [0-2147483647 (bytes)]	
	ret+4	(No. of bytes written to file (high word))	

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.16.60 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes
s	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		Yes	Yes

Note: See Section 1.17 "Devices Available As Instruction Parameters."

■ Resource Relays

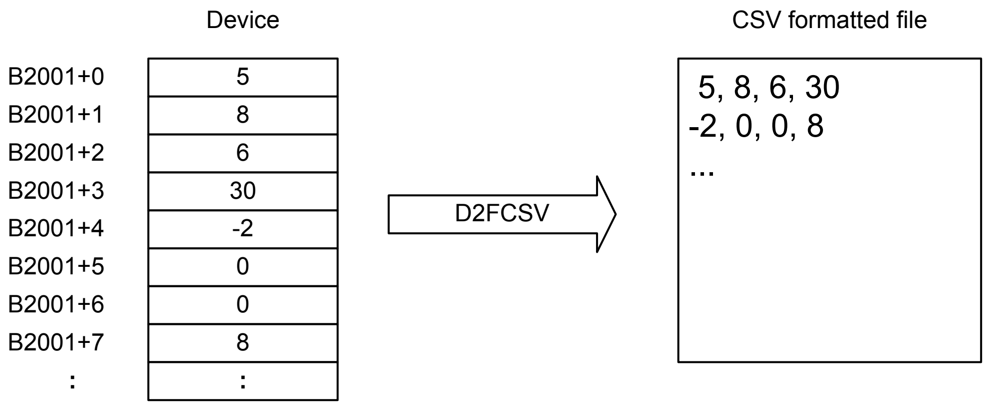
Table 3.16.61 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Converts device data to text and outputs a CSV formatted file.

- Available device unit options for reading are bit, byte, word and long word.
- Device data can be converted to text in decimal, hexadecimal or floating-point representation after reading.
  Decimal ("0" to "1", "-128" to "127", "-32768" to "32767", "-2147483648" to "2147483647")
  Hexadecimal ("0" to "FFFFFFFF")
  Floating-point ([-]d.dddd e[+/-]ddd, [-]dddd.dddd, infinity "-INF" or "+INF", non-numeric "NaN")
- You can specify the field length in characters for text conversion.
- You can specify whether to pad with space characters or pad with zeros when the converted text is shorter than the specified field length.
- Available field delimiter options are the comma (,) and Tab characters.
- Newline can be specified as CRLF (standard for Windows) or LF.
- The number of fields in one record (from line beginning to line end) can be specified.



Note: Device numbers and conversion method shown are examples.

FB0213.VSD

Figure 3.16.12 Device to CSV Formatted File Conversion

TIP

Reading of Device Data

- Reading of data from devices spans multiple scan cycles.
- If you specify the field representation type as floating-point representation, you must specify the device unit as long word for devices storing IEEE single-precision floating-point numbers.

Conversion Error and Interpretation Error

- If a conversion error occurs, "ERR" is written to the field. If a conversion error is detected for a field during conversion, an error is generated but processing continues.
- If the converted text string is longer than the specified field length, a conversion error is generated.

Data Conversion

- If the field representation type is specified as decimal, the sign is included in the output digit count.
- In floating-point representation B, there are always 6 digits after the decimal point. Numeric values smaller than 0.000001 are rounded to 0.

File Pointer Movement

- Start position for writing = current file pointer position
- File pointer position after instruction execution = the byte following the last written byte

■ Programming Example

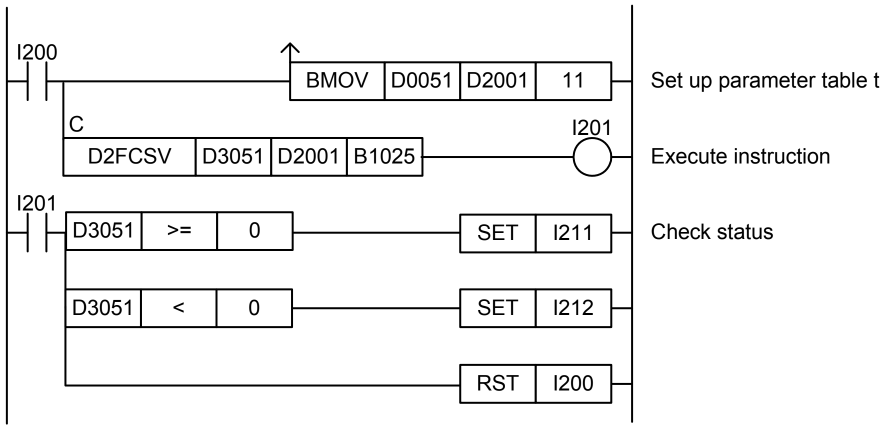


Figure 3.16.13 Example of a Convert Device to CSV File Program

This sample code writes device data starting from B1025 to a file opened as file ID 7 according to the conditions set up in parameter table t.

It specifies ret(=D3051), t(=D2001) and s(=B1025), with t set up as follows.

Device	Value	Table Parameter
t=D2001	100	Timeout interval (= 10 s)
D2002	7	File ID (= 7)
D2003	2	Device unit (= word)
D2004	300	No. of data units to be read (= 300)
D2005		
D2006	1	Field representation type (= hexadecimal)
D2007	4	Field length (= 4 characters)
D2008	0	Field space handling (= pad with spaces)
D2009	0	Delimiter option (= comma)
D2010	0	Newline option (= CRLF)
D2011	30	Newline insertion position (= every 30 fields)

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	300	No. of data units written to file
D3053		
D3054	1510	No. of bytes written to file
D3055		

3.16.12 Convert Binary File to Device (F2DBIN)

F3SP66 F3SP71
F3SP67 F3SP76

Converts data in binary file and writes the data to contiguous devices using the specified data unit.

Table 3.16.62 Convert Binary File to Device

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Convert Binary File to Device	F2DBIN	C ┤ F2DBIN │ │ │ ├	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

C
Convert Device to Binary File ┤ D2FBIN │ ret │ t │ s ├

Table 3.16.63 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2, t+3	No. of data units to be read (L) [-1 = Until file end 0 - 4194304 (if device unit = bit) 0 - 524288 (if device unit = byte) 0 - 262144 (if device unit = word) 0 - 131072 (if device unit = long word)]
	t+4	Data unit (W) [1 = byte 2 = word 3 = long word]
	t+5	Device unit (W) [0 = bit 1 = byte 2 = word 3 = long word]
	t+6	Sign extension[*2] (W) [0 = Pad with zeros 1 = Extend sign]
	t+7, t+8	Write limit in words (L) [1 - 262144 (words)]
d		First device for writing (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".
*2: If the device unit is larger than the data unit, the sign extension setting should be specified.

■ Status (Return Value)

Table 3.16.64 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	No. of data units read (low word) (L) [0-2147483647 (data count)]	
	ret+2	(No. of data units read high word)	
	ret+3	No. of written words low word (L) [0-2147483647 (words)]	
	ret+4	(No. of written words high word)	

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.16.65 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes
d		✓	✓	✓	✓				✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

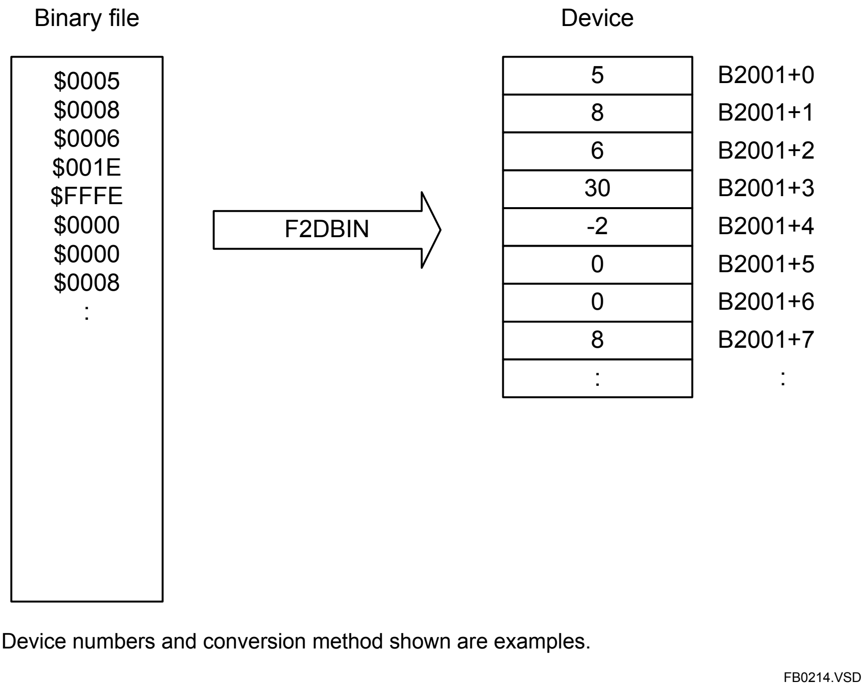
■ Resource Relays

Table 3.16.66 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Converts data in binary file and writes the data to contiguous devices using the specified data unit.



Note: Device numbers and conversion method shown are examples.

Figure 3.16.14 Binary File to Device Conversion

TIP

- If a conversion error occurs, 0 is written to the device. If a conversion error is detected during conversion, an error is generated but processing continues.
- If the specified data unit is larger than the specified device unit, a conversion error is generated.
- If the device unit is specified as bit, 0 is stored for a zero value while 1 is stored for any other value.
- If the specified data unit is smaller than the specified device unit, you can specify to pad with '0's or extend the sign.
- If end-of-file is encountered before the required number of data units are read, execution ends without error.
- If the device unit is specified as byte, the highest-order 8 bits of word data are processed first.
- Writing to device spans multiple scan cycles.

TIP

The file pointer movement is as follows:

- Start position for reading = current file pointer position
- File pointer position after instruction execution = the byte following the last byte that was read

CAUTION

Pay attention to the size of data read when specifying "until file end" as the number of data units to be read.

■ Programming Example

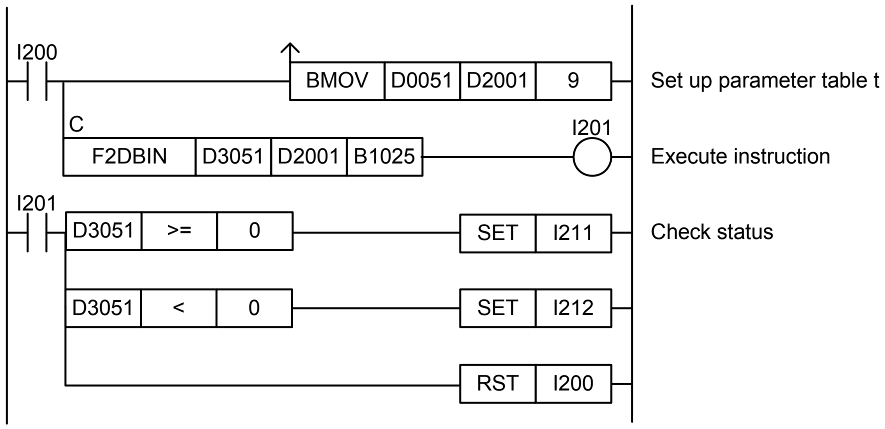


Figure 3.16.15 Example of a Convert Binary File to Device Program

This sample code reads the data contained in a binary file opened as file ID 4 according to the conditions set up in parameter table t.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as shown in the table below.

Device	Value	Table Parameter Name
t=D2001	100	Timeout interval (= 10 s)
D2002	4	File ID (= 4)
D2003	-1	No. of data units to be read (= until file end)
D2004		
D2005	2	Data unit for reading (= word)
D2006	2	Device unit (= word)
D2007	0	Sign extension (irrelevant for this sample program)
D2008	10000	Write limit in words (= 10000 words)
D2009		

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	4066	No. of data units read
D3053		
D3054	4066	No. of written words
D3055		

3.16.13 Convert Device to Binary File (D2FBIN)

F3SP66 F3SP71
F3SP67 F3SP76

Converts device data to a binary file.

Table 3.16.67 Convert Device to Binary File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Convert Device to Binary File	D2FBIN	C D2FBIN	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

Convert Device to Binary File C D2FBIN ret t s

Table 3.16.68 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2, t+3	No. of data units to be read (L) [0 - 4194304 (if device data unit = bit) 0 - 524288 (if device data unit = byte) 0 - 262144 (if device data unit = word) 0 - 131072 (if device data unit = long word)]
	t+4	Device data unit (W) [0 = bit 1 = byte 2 = word 3 = long word]
	t+5	File data unit (W) [1 = byte 2 = word 3 = long word]
	t+6	Sign extension[*2] (W) [0 = Pad with zeros 1 = Extend sign]
s		First device for reading (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

*2: If the file data unit is larger than the device data unit, the sign extension setting should be specified.

■ Status (Return Value)

Table 3.16.69 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	No. of data units written to file (low word) (L) [0-4194304 (data count)]	
	ret+2	(No. of data units written to file (high word))	
	ret+3	No. of bytes written to file (low word) (L) [0-2147483647 (bytes)]	
	ret+4	(No. of bytes written to file (high word))	

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.16.70 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes
s	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

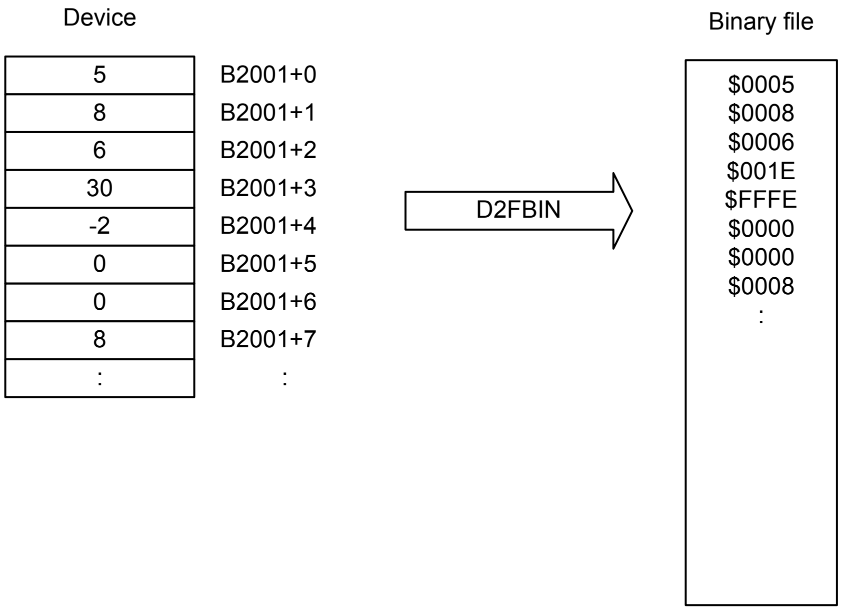
■ Resource Relays

Table 3.16.71 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Converts device data to a binary file.



Note: Device numbers and conversion method shown are examples.

FB0215.VSD

Figure 3.16.16 Device Data to Binary File Conversion

TIP

- If a conversion error occurs, 0 is written to the file. If a conversion error is detected during conversion, an error is generated but processing continues.
- If the specified device unit is larger than the specified file unit, a conversion error is generated.
- If the specified device unit is smaller than the specified file unit, you can specify to pad with '0's or extend the sign.
- If the device data unit is specified as byte, after data are read in word units, the highest-order 8 bits of word data are processed first.
- Reading of data from devices spans multiple scan cycles.

TIP

The file pointer movement is as follows:

- Start position for writing = current file pointer position
- File pointer position after instruction execution = the byte following the last written byte

■ Programming Example

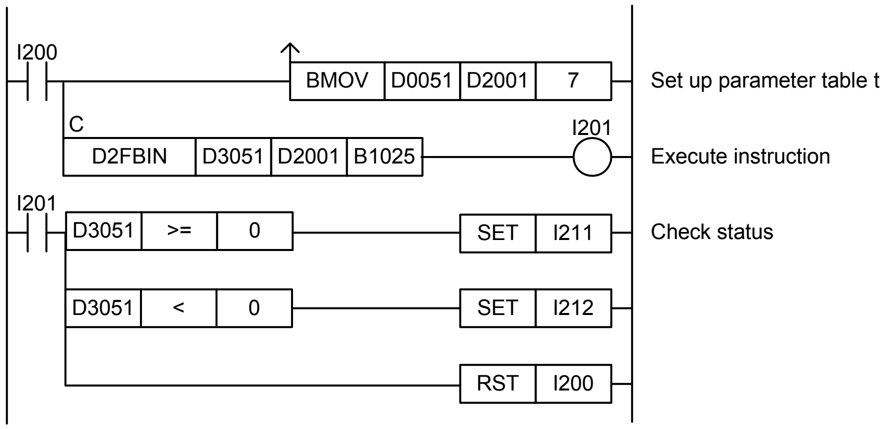


Figure 3.16.17 An Example of a Convert Device to Binary File Program

This sample code writes device data starting from B1025 to a file opened as file ID 7 according to the conditions set up in parameter table t.

It specifies ret(=D3051), t(=D2001) and s(=B1025), with t set up as follows.

Device	Value	Table Parameter
t=D2001	100	Timeout interval (= 10 s)
D2002	7	File ID (= 7)
D2003	300	No. of data units to be read (= 300)
D2004		
D2005	2	Device data unit (= word)
D2006	3	File data unit (= long word)
D2007	1	Sign extension (= Extend sign)

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	300	No. of data units written to file
D3053		
D3054	1200	No. of bytes written to file
D3055		

3.17 File Operation Instructions

File operation instructions performs file based processing such as copying or moving a file.
When using a file operation instruction, you directly specify a file using its filename. Unlike file access instructions, there is no need to open a file using the FOPEN instruction. The system automatically gets the required file ID internally.
Of the file operation instruction group and disk operation instruction group, only instructions from one group can be executed at any one time. If you attempt to execute multiple instructions, the instruction that is executed later will be terminated with a redundant use of function error (error code -3001). Before executing a file operation instruction, check to ensure that the File/Disk Operation Group Busy relay (M1025) is OFF.

TIP

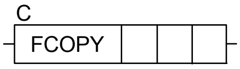
File operation instructions automatically get the required file IDs, sharing a common pool of available file IDs with other file system processes. If many files are opened by file access instructions or other file system processes so that no more unused file IDs are available, file operations instructions cannot be executed successfully.

F3SP66 F3SP71
F3SP67 F3SP76

3.17.1 Copy File (FCOPY)

Copies one or more files.

Table 3.17.1 Copy File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Copy File	FCOPY		✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

Copy File

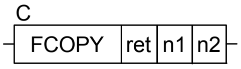


Table 3.17.2 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
n2	Overwrite option (W) [0 = Exit with error without overwriting 1 = Overwrite file of the same name 2 = Overwrite read-only file of the same name]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.17.3 Text Parameter

Parameter		Description
1	s	Source pathname
2	d	Destination pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Subsection 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.17.4 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.17.5 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relay

Table 3.17.6 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1026	No Unused File ID	Execute instruction only if both "No Unused File ID" and "File/Disk Operation Group Busy" relays are OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

Copies one or more files.

Multiple files can be specified using wildcard characters.

If a wildcard character is used to copy multiple files and an error is detected during copying, execution terminates with error.

If the specified source pathname (s) is a directory, the effect is equivalent to specifying a wildcard character ('*') to include all files stored immediately below the directory.

If a file having the same name or a file having the same name and a read-only file attribute already exists at the specified destination pathname, whether the file will be overwritten depends on the specified overwrite option.

■ Programming Example

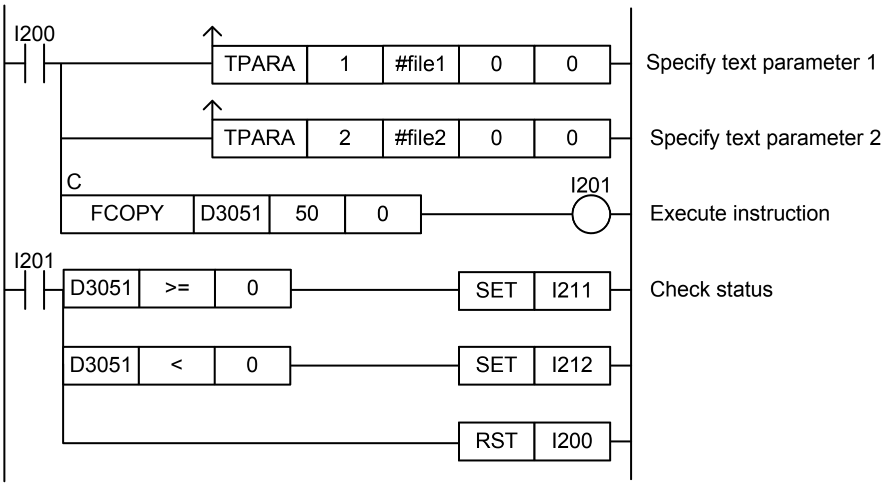


Figure 3.17.1 Example of a Copy File Program

This sample code copies files from the pathname defined by constant name #file1 to the pathname defined by constant name #file2.

It specifies timeout interval as 50 (5 s) and overwrite option as 0 (Exit with error without overwriting).

The table below shows an example of the returned status data (ret), assuming normal exit.

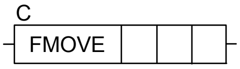
Device	Value	Table Parameter
ret=D3051	0	Status

3.17.2 Move File (FMOVE)

F3SP66 F3SP71
F3SP67 F3SP76

Moves one or more files.

Table 3.17.7 Move File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Move File	FMOVE		✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

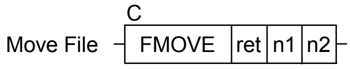


Table 3.17.8 Parameter

Parameter	Description
ret*1	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
n2	Overwrite option (W) [0 = Exit with error without overwriting 1 = Overwrite file of the same name 2 = Overwrite read-only file of the same name]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.17.9 Text Parameter

Parameter		Description
1	s	Source pathname
2	d	Destination pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Subsection 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.17.10 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.17.11 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Devices Available As Instruction Parameters."

■ Resource Relay

Table 3.17.12 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1026	No Unused File ID	Execute instruction only if both "No Unused File ID" and "File/Disk Operation Group Busy" relays are OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

Moves one or more files.

Multiple files can be specified using wildcard characters.

If a wildcard character is used to move multiple files and an error is detected during moving, execution terminates with error.

If the specified source pathname (s) is a directory, the effect is equivalent to specifying a wildcard character ('*') to include all files stored immediately below the directory.

If a file having the same name or a read-only file having the same name already exists at the specified destination pathname, whether the file will be overwritten depends on the specified overwrite option.

■ Programming Example

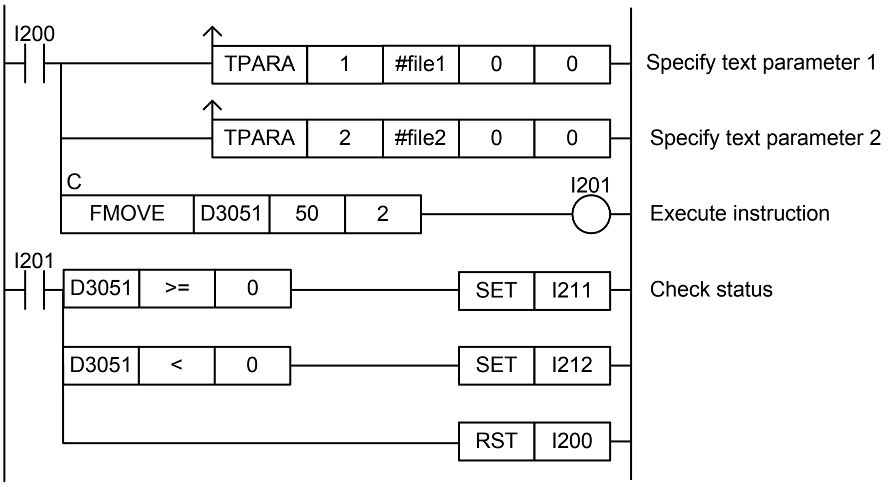


Figure 3.17.2 Example of a Move File Program

This sample code moves copies files from the pathname defined by constant name #file1 to the pathname defined by constant name #file2.

It specifies timeout interval as 50 (5s) and overwrite option as 2 (Overwrite read-only file of the same name).

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

3.17.3 Delete File (FDEL)

F3SP66 F3SP71
F3SP67 F3SP76

Deletes one or more files.

Table 3.17.13 Delete File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Delete File	FDEL	C FDEL	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

Delete File C FDEL ret n1 n2

Table 3.17.14 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
n2	Forced delete option (W) [0 = Exit with error without deleting read-only files 1 = Delete read-only files]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.17.15 Text Parameter

Parameter		Description
1	d	Target pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Subsection 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.17.16 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.17.17 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relay

Table 3.17.18 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

Deletes one or more files.

Multiple files can be specified using wildcard characters.

If a wildcard character is used to delete multiple files and an error is detected during deletion, execution terminates with error.

If the specified target pathname (d) is a directory, the effect is equivalent to specifying a wildcard character ('*') to include all files stored immediately below the directory.

You can specify whether to delete files with read-only file attribute using the forced delete option.

CAUTION

The wildcard character can be used to delete many files in one go. Beware of inadvertently deleting required files when using the wildcard character.

■ Programming Example

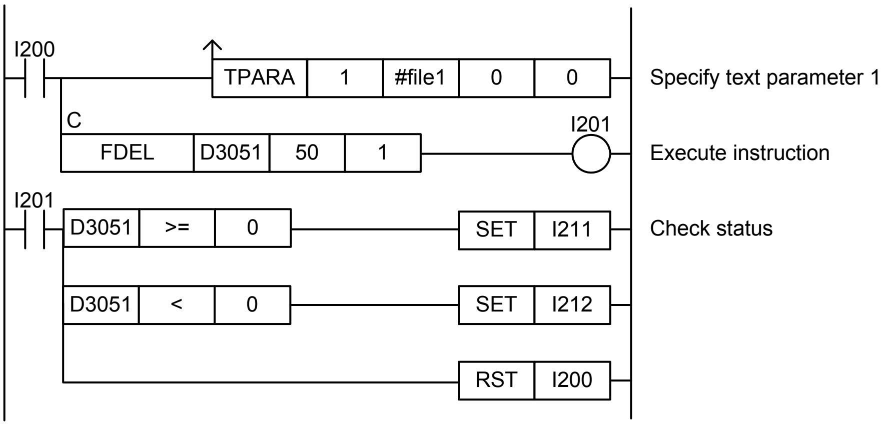


Figure 3.17.3 Example of a Delete File Program

This sample code deletes the file designated by the pathname defined by constant name #file1.

It specifies timeout interval as 50 (5 s) and forced delete option as 1 (Delete read-only files).

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

3.17.4 Make Directory (FMKDIR)

F3SP66 F3SP71
F3SP67 F3SP76

Creates a directory.

Table 3.17.19 Make Directory

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Make Directory	FMKDIR	C FMKDIR	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

Make Directory C FMKDIR ret n1

Table 3.17.20 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.17.21 Text Parameter

Parameter		Description
1	d	Pathname of directory to be created

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Subsection 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.17.22 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.17.23 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relay

Table 3.17.24 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

Creates a directory.

■ Programming Example

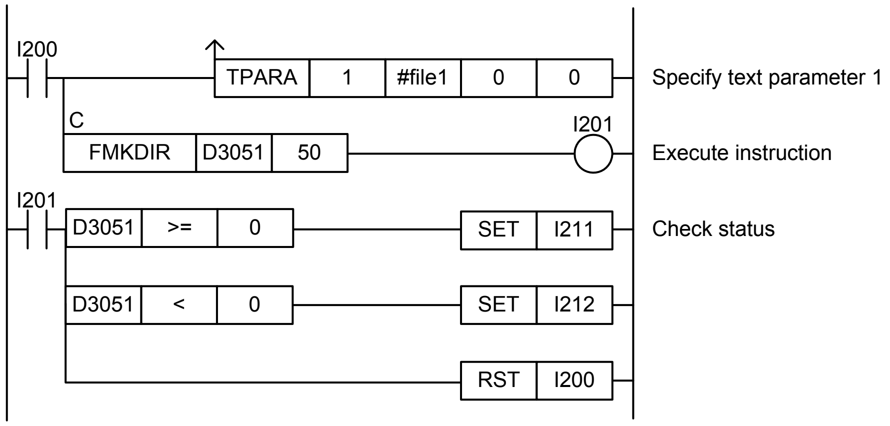


Figure 3.17.4 Example of a Make Directory Program

This sample code creates a directory designated by the directory pathname defined by constant name #file1. It specifies timeout interval as 50 (5 s).

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

3.17.5 Remove Directory (FRMDIR)

F3SP66 F3SP71
F3SP67 F3SP76

Deletes a directory.

Table 3.17.25 Remove Directory

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Remove Directory	FRMDIR	C –[FRMDIR | | |]–	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

C
Remove Directory –[FRMDIR | ret | n1 | n2]–

Table 3.17.26 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
n2	Delete all option (W) [0 = No 1 = Yes]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.17.27 Text Parameter

Parameter		Description
1	d	Pathname of directory to be deleted

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Subsection 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.17.28 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.17.29 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.17.30 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

Deletes a directory.

If the directory to be deleted is not empty, whether the files and subdirectories contained in the directory are deleted depends on the specified "Delete all" option. If the specified "Delete all" option is 1, all contents of the directory are deleted. If a read-only file is encountered during deletion, execution terminates with error. If the specified "Delete all option" is 0 and the directory to be deleted is not empty, execution terminates with error without deleting the directory.

■ Programming Example

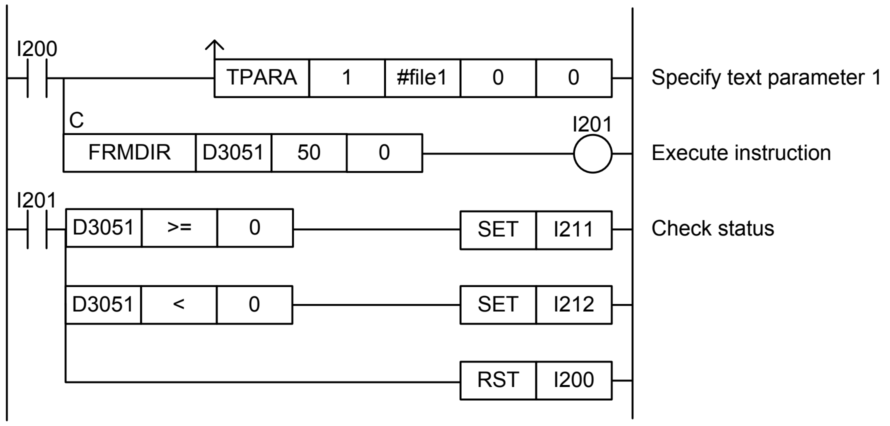


Figure 3.17.5 Remove Directory Programming Example

This sample code deletes the directory designated by the directory pathname defined by constant name #file1. It specifies timeout interval as 50 (5 s).

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

3.17.6 Rename File (FREN)

F3SP66 F3SP71
F3SP67 F3SP76

Renames a file or directory.

Table 3.17.31 Rename File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Rename File	FREN	C FREN	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

C
Rename File FREN ret n1

Table 3.17.32 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.17.33 Text Parameter

Parameter		Description
1	s	Old file pathname or directory pathname
2	d	New file name or directory name

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Subsection 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.17.34 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.17.35 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.17.36 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

Renames a file or directory.

For the "New filename" parameter, you should specify only a filename or directory name, without its file path. If you specify a file pathname including a directory for the parameter, the instruction returns an Invalid Pathname error (status code -12200).

■ Programming Example

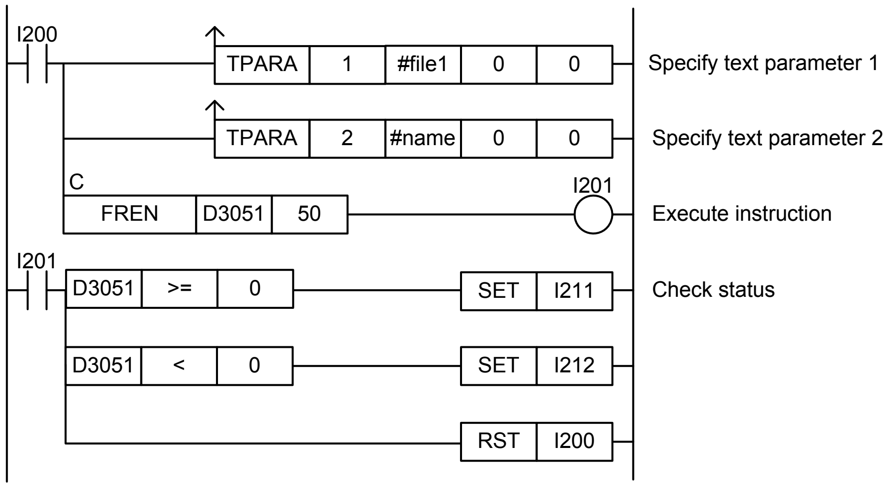


Figure 3.17.6 Example of a Rename File Program

This sample code renames the file designated by the pathname defined by constant name #file1 to the filename defined by constant name #name. It specifies timeout interval as 50 (5 s).

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

3.17.7 File Status (FSTAT)

F3SP66 F3SP71
F3SP67 F3SP76

Returns file status information for a specified file or specified directory.

Table 3.17.37 File Status

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	File Status	FSTAT	C FSTAT	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

C
File Status ┤ FSTAT │ ret │ t │ d ├

Table 3.17.38 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	Output option (W) [0 = File presence only Non-zero = All information]
d		File status output device (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.17.39 Text Parameter

Parameter		Description
1	s	Target file/directory pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Subsection 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.17.40 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit (specified file or directory exists)
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.17.41 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes
d									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.17.42 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

Returns file status information for a specified file or specified directory.

File status information that can be returned include whether file/directory exists, file/directory attribute, file size and last modified time.

Output Option:

Output option controls the file status information to be returned.

If 0 (=file presence only) is specified, nothing is stored to the file status output device. However, whether the file/directory is present is indicated by the returned status (ret).

If 1 (=all information) is specified, the file attribute, file size and last modified time are stored to the specified file status output device.

Table 3.17.43 Information Stored to File Status Output Device

Appended Information		Offset (word)	Size (word)	Description
File attribute	-	+0	1	*1
File size*3	Low word	+1	2	The file size in bytes is stored. For a directory, 0 is stored.
	High word	+2		
Last modified time*2	Year	+3	1	Data is stored in BCD representation. Example: 1999 as $0099, 2000 as $0000
	Month	+4	1	Data is stored in BCD representation. Example: January as $0001
	Date	+5	1	Data is stored in BCD representation. Example: 28th as $0028
	Hour	+6	1	Data is stored in BCD representation. Example: 18:00 hours as $0018
	Minute	+7	1	Data is stored in BCD representation. Example: 15 minutes as $0015
	Second	+8	1	Data is stored in BCD representation. Example: 30 seconds as $0030

Note: Nothing is stored to File Status Output Device if the specified output option is 0 (=File presence only).

*1: For details on the format of the stored file attribute data, see the following table.

*2: The stored information has the same format as the date/time special registers (Z049-Z054).

*3: Word data is handled as an unsigned decimal or hexadecimal number.

Table 3.17.44 File (Directory) Attribute Data

Bit Position	Description
0 (LSB)	0 =Read and write 1=Read-only
1	0=Non-system file 1=System file
2	0=Non-hidden file 1=Hidden file
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15 (MSB)	0 =File 1 =Directory

■ Programming Example

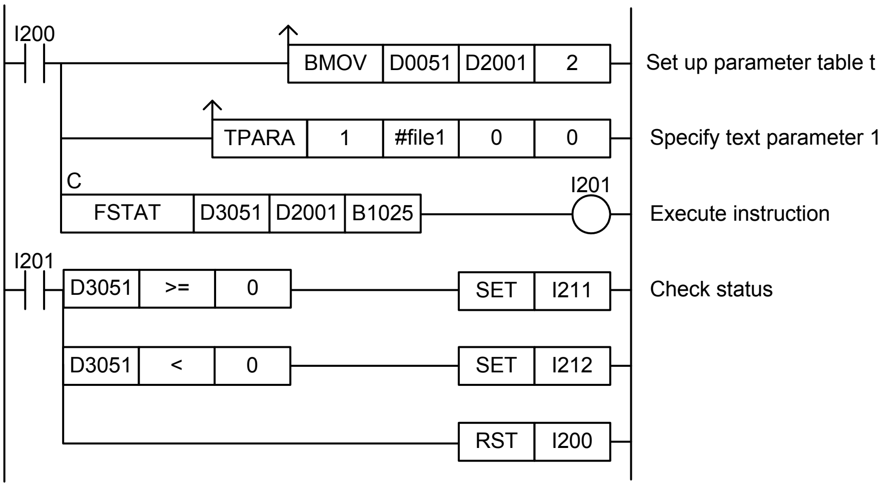


Figure 3.17.7 Example of a File Status Program

This sample code stores status information for the file designated by the pathname defined by constant name #file1 to the device, starting from B1025. It specifies ret(=D3051) and t(=D2001), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	50	Timeout interval (= 5 s)
D2002	1	Output option (=All information)

The table below shows the content of ret, assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

The table below shows the content of d, assuming normal exit.

Device	Value	Table Parameter
B1025	$0000	File attribute
B1026	4620	File size
B1027		
B1028	$0005	Year
B1029	$0012	Month
B1030	$0031	Date
B1031	$0011	Hour
B1032	$0003	Minute
B1033	$0050	Second

3.17.8 File List Start (FLSFIRST)

F3SP66 F3SP71
F3SP67 F3SP76

Declares a file list operation for getting status information of successive files or directories.

Table 3.17.45 File List Start

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	File List Start	FLSFIRST	C ┤ FLSFIRST │ │ ├	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

```
                   C
File List Start ┤ FLSFIRST │ ret │ n ├
```

Table 3.17.46 Parameter

Parameter	Description
ret*1	Device for storing return status (W)
n	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.17.47 Text Parameter

Parameter		Description
1	s	Target pathname (including use of wildcard)

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Subsection 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.17.48 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.17.49 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.17.50 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

TIP

FLSFIRST (File List Start), FLS (File List Next) and FLSFIN (File List End) instructions are used as a group in a file list operation.

Declares a file list operation for getting status information of successive files.

After this declaration, the File List Next (FLS) instruction can be executed repeatedly to get status information for multiple files designated by the use of a wildcard character within the specified directory, one at a time. If the specified source pathname ("s" parameter) is a directory, the effect is equivalent to specifying a wildcard character ('*') to include all files stored immediately below the directory.

The reference pathname declared by this instruction for the list operation remains in force until the File List End (FLSFIN) instruction is executed. Therefore, if the File List End (FLSFIN) instruction is not executed, no new reference pathname can be specified using a File List Start (FLSFIRST) instruction.

The reference pathname can be specified using the wildcard character.

Example: To target all files and directories on the RAM disk, specify:

```
\RAMDISK\*
```

Example: To target all files with file extension ".txt" on the RAM disk, specify:

```
\RAMDISK\*.txt
```

Example: To target all files with filename "abc" and any file extension on the RAM disk, specify:

```
\RAMDISK\abc.*
```

If you execute this instruction while a reference pathname declared by an earlier execution of this instruction is in force, an error will be generated. To specify a new reference pathname for file list operation, you must first execute the File List End (FLSFIN) instruction.
If any reference file is not found, it returns a status of "No match found" (status code -2001).

SEE ALSO

For details on how to get file status information successively, see Subsection 3.17.9, "File List Next (FLS)".

■ Programming Example

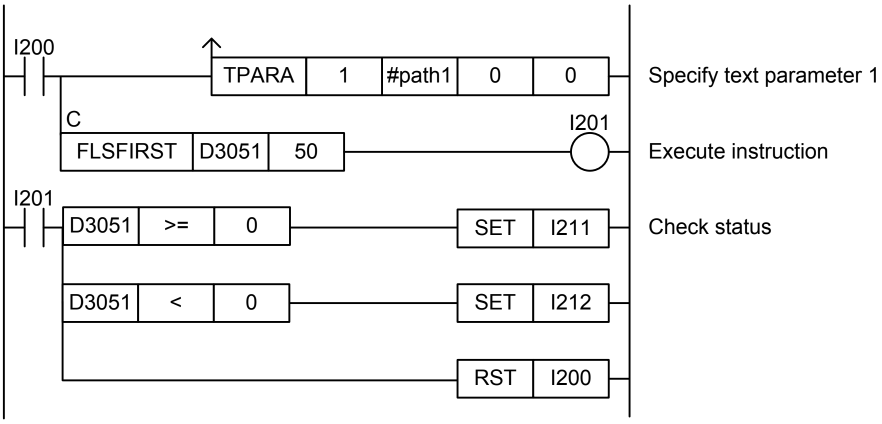


Figure 3.17.8 Example of a File List Start Program

This sample code declares a file list operation for the directory defined by constant name #path1. It specifies timeout interval as 50 (5 s).

The table below shows an example of the returned status data (ret), assuming normal exit.

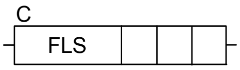
Device	Value	Table Parameter
ret=D3051	0	Status

3.17.9 File List Next (FLS)

F3SP66 F3SP71
F3SP67 F3SP76

Gets status information of the next file in a file list operation pre-declared with a reference pathname.

Table 3.17.51 File List Next

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	File List Next	FLS		✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

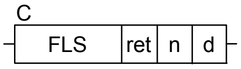
File List Next 

Table 3.17.52 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
d	File Status Output Device (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.17.53 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.17.54 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
d									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.17.55 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

TIP

FLSFIRST (File List Start), FLS (File List Next) and FLSFIN (File List End) instructions are used as a group in a file list operation.

Gets status information of the next file or directory in a file list operation pre-declared with a reference pathname.

When the instruction encounters the last file matching the wildcard character in the reference pathname, it returns a status of "No match found" (status code -2001).

The table below shows the file status information stored to the specified file status output device by the instruction.

Table 3.17.56 Information Stored to File Status Output Device

Appended Information		Offset (word)	Size (word)	Description
File attribute	–	+0	1	*1
File size*3	Low word	+1	2	The file size in bytes is stored. For a directory, 0 is stored.
	High word	+2		
Last modified time*2	Year	+3	1	Data is stored in BCD representation. Example: 1999 as $0099, 2000 as $0000
	Month	+4	1	Data is stored in BCD representation. Example: January as $0001
	Date	+5	1	Data is stored in BCD representation. Example: 28th as $0028
	Hour	+6	1	Data is stored in BCD representation. Example: 18:00 hours as $0018
	Minute	+7	1	Data is stored in BCD representation. Example: 15 minutes as $0015
	Second	+8	1	Data is stored in BCD representation. Example: 30 seconds as $0030
Filename	–	+9 to +136	128	The filename is stored with a variable length.

*1: For details on the format of the stored file attribute data, see the following table.
*2: The stored information has the same format as the date/time special registers (Z049-Z054).
*3: Word data is handled as an unsigned decimal or hexadecimal number.

Table 3.17.57 File (Directory) Attribute Data

Bit Position	Description
0 (LSB)	0 =Read and write 1=Read-only
1	0=Non-system file 1=System file
2	0=Non-hidden file 1=Hidden file
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15 (MSB)	0 =File 1 =Directory

How to Perform a File List Operation:

1. Declare a file list operation by executing a File List Start (FLSFIRST) instruction, specifying a target pathname including a wildcard character.
2. Execute the File List Next (FLS) instruction to get the status information of the next file. The order of the retrieval will be according to the order of files on the file system.
3. Repeat step 2 until "No match found" (-2001) is returned in the instruction status.
4. Execute the File List End (FLSFIN) instruction to end the declared file list operation.

■ Programming Example

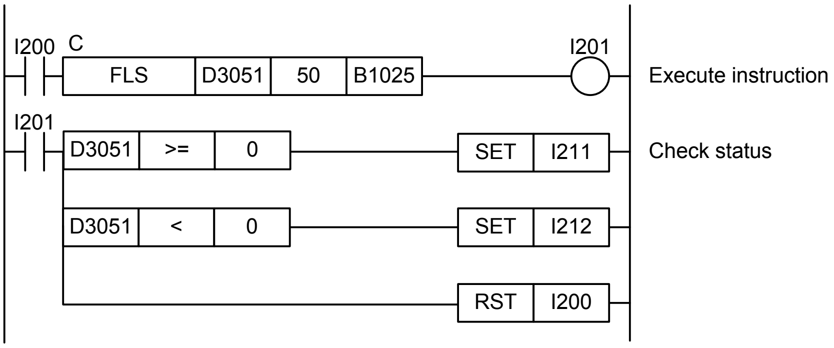


Figure 3.17.9 Example of a File List Next Program

This sample code stores status information for files within the directory previously declared by a File List Start (FLSFIRST) instruction to device, starting from B1025. It specifies timeout interval as 50 (5 s).

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

The table below shows an example of the data stored to file status output device d, assuming normal exit.

Device	Value	Table Parameter
B1025	$0000	File attribute
B1026	4620	File size
B1027		
B1028	$0005	Year
B1029	$0012	Month
B1030	$0031	Date
B1031	$0011	Hour
B1032	$0003	Minute
B1033	$0050	Second
B1034	“AB”	Filename
B1035	“C.”	
B1036	“CS”	
B1037	“V”+NULL	

3.17.10 File List End (FLSFIN)

F3SP66 F3SP71
F3SP67 F3SP76

Declares the end of a file list operation.

Table 3.17.58 File List End

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	File List End	FLSFIN	C FLSFIN	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

C
File List End FLSFIN | ret | n

Table 3.17.59 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.17.60 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.17.61 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.17.62 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

TIP

FLSFIRST (File List Start), FLS (File List Next) and FLSFIN (File List End) instructions are used as a group in a file list operation.

Declares the end of a file list operation.

After instruction execution, the reference pathname specified in the File List Start (FLSFIRST) instruction executed previously no longer holds and a new reference pathname can be specified by executing another File List Start (FLSFIRST) instruction.

■ Programming Example

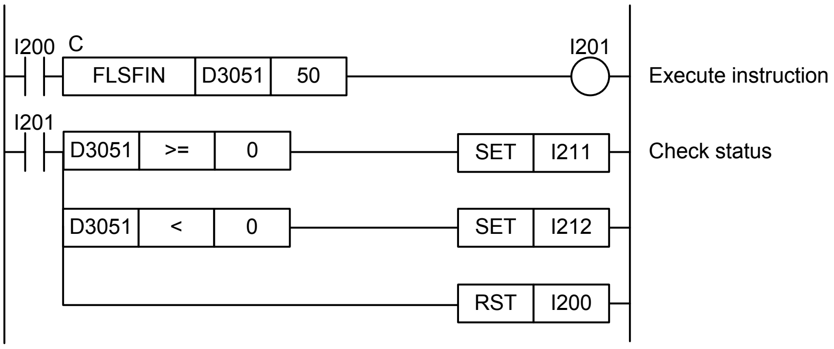


Figure 3.17.10 Example of a File List End Program

This sample code ends a file list operation. It specifies timeout interval as 50 (5 s).

The table below shows an example of the returned status data (ret), assuming normal exit.

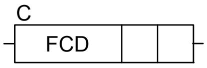
Device	Value	Table Parameter
ret=D3051	0	Status

3.17.11 Change Directory (FCD)

F3SP66 F3SP71
F3SP67 F3SP76

Changes the current directory.

Table 3.17.63 Change Directory

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Change Directory	FCD		✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

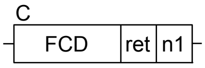
Change Directory 

Table 3.17.64 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.17.65 Text Parameter

Parameter		Description
1	n2	New current directory pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Subsection 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.17.66 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.17.67 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.17.68 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

Changes the current directory.

This instruction changes the current directory, which is common to file access instructions and file operation instructions, but unrelated to the current directory of the FTP client.

The default initial current directory is "\RAMDISK".

A parameter error is generated if the destination directory pathname is omitted.

■ Programming Example

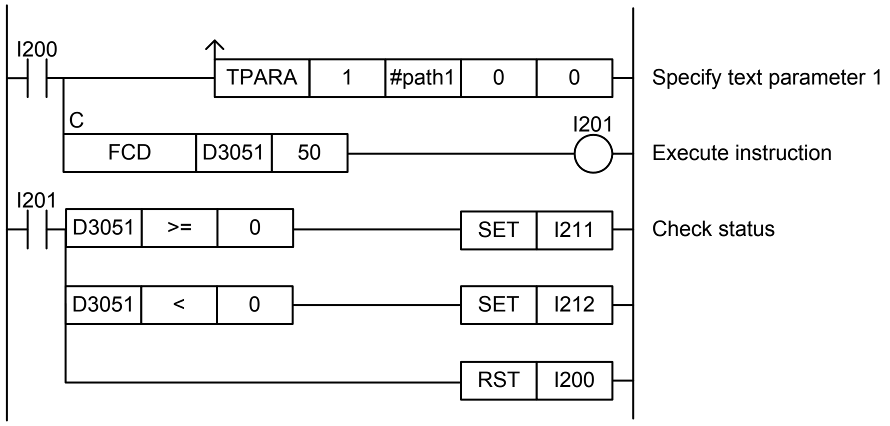


Figure 3.17.11 Example of a Change Directory Program

This sample code changes the current directory to the directory defined by constant name #path1. It specifies timeout interval as 50 (5 s).

The table below shows an example of the returned status data (ret), assuming normal exit.

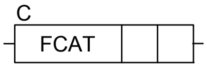
Device	Value	Table Parameter
ret=D3051	0	Status

3.17.12 Concatenate File (FCAT)

F3SP66 F3SP71
F3SP67 F3SP76

Concatenates two files.

Table 3.17.69 Concatenate File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Concatenate File	FCAT		✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

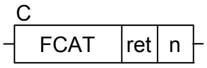
Concatenate File 

Table 3.17.70 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.17.71 Text Parameter

Parameter		Description
1	s1	File pathname 1 (source file 1 and destination)
2	s2	File pathname 2 (source file 2)

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Subsection 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.17.72 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.17.73 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.17.74 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1026	No Unused File ID	Execute instruction only if both "No Unused File ID" and "File/Disk Operation Group Busy" relays are OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

Concatenates two files.

This instruction appends the file designated by file pathname 2 to the file designated by file pathname 1, and stores the result as a file designated by file pathname 1.

Wildcard characters may not be used with this command.

■ Programming Example

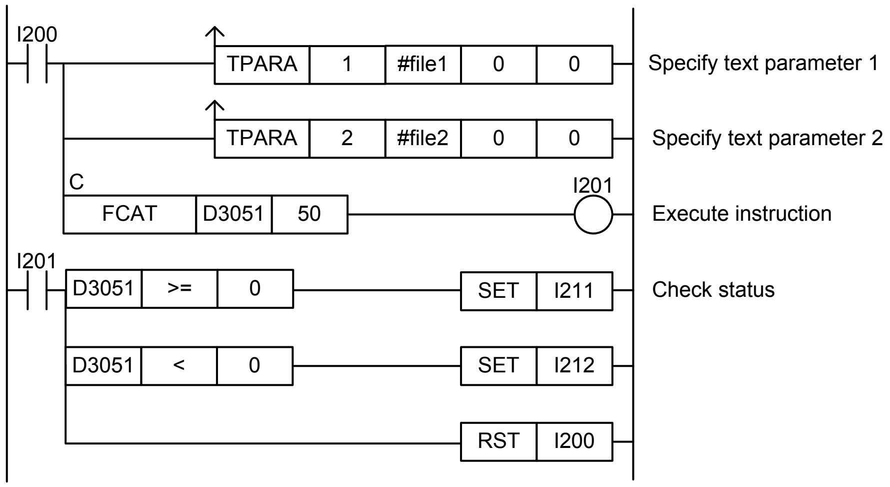


Figure 3.17.12 Example of a Concatenate File Program

This sample code appends the file designated by the file pathname defined by constant name #file2 to the file designated by the file pathname defined by constant name #file1. It specifies timeout interval as 50 (5 s).

The table below shows an example of the returned status data (ret), assuming normal exit.

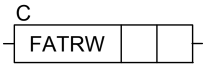
Device	Value	Table Parameter
ret=D3051	0	Status

3.17.13 Change File Attribute (FATRW)

F3SP66 F3SP71
F3SP67 F3SP76

Changes the attribute of a specified file or directory.

Table 3.17.75 Change File Attribute

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Change File Attribute	FATRW		✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

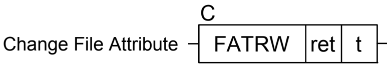


Table 3.17.76 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File attribute (W) [*2]
	t+2	File attribute mask (W) [bit mask] [*2] Specifies which bits of the file attribute are to be changed. [0=hold, 1=overwrite]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

*2: For details on the bitmap of the file attribute, see Table 3.17. 81, "File (Directory) Attribute Data".

Table 3.17.77 Text Parameter

Parameter		Description
1	d	Target file/directory pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Subsection 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.17.78 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.17.79 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.17.80 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

Changes the attribute of a specified file or directory.

Specify the attribute value as one word in binary representation as shown in the table below.

Multiple files can be specified using wildcard characters. If a wildcard character is used to change the attribute of multiple files and an error is detected during execution, execution terminates with error.

File attributes can be changed even for files that are open in write mode.

The file attribute mask can be used to specify the attributes to be changed. A '1' in a bit position of the mask indicates to change the corresponding file attribute while a '0' in a bit position indicates to leave the corresponding file attribute unchanged.

Table 3.17.81 File (Directory) Attribute Data

Bit Position	Description
0 (LSB)	0 = Read and write 1 = Read-only
1	0 = Non-system file 1 = System file
2	0 = Non-hidden file 1 = Hidden file
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15 (MSB)	0 = File 1 = Directory

■ Programming Example

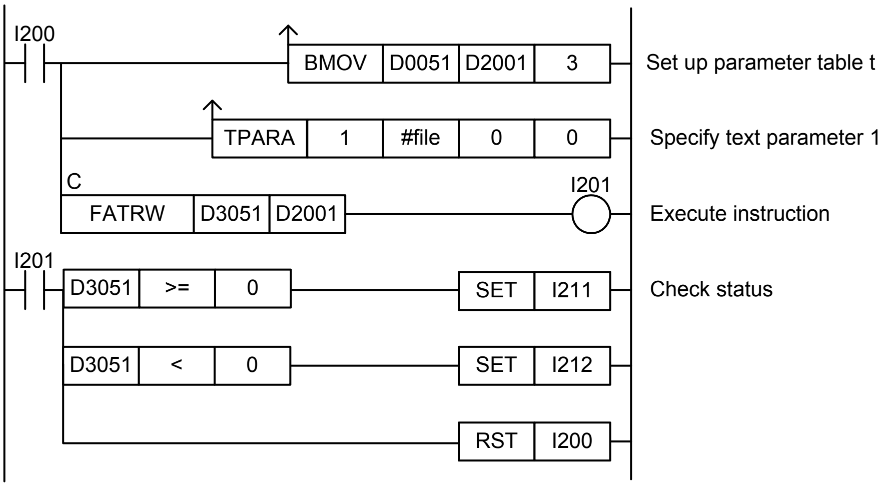


Figure 3.17.13 Change File Attribute Programming Example

This sample code sets the read-only attribute of all CSV formatted files in the directory designated by the directory pathname defined by constant name #file.

```
#file = "\RAMDISK\NEWDATA\*.CSV"
```

It specifies ret(=D3051) and t(=D2001), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	50	Timeout interval (= 5 s)
D2002	$0001	File attribute (= set read-only attribute)
D2003	$0001	File attribute mask (= change read-only attribute and nothing else)

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

3.18 UDP/IP Socket Communications Instructions

3.18.1 UDP/IP Open (UDPOPEN)

F3SP66 F3SP71
F3SP67 F3SP76

Opens a UDP/IP socket to enable communications.

Table 3.18.1 UDP/IP Open

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	UDP/IP Open	UDPOPEN	C UDPOPEN	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

UDP/IP Open — C UDPOPEN | ret | n1 | n2

Table 3.18.2 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
n2	My port number (W) [1-65535][*2 *3]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".
*2: Do not specify my port number as 12289, 12290, 12291, 12305 or 12307 as these numbers are used by the higher-level link service and remote programming service.
*3: Word data is handled as an unsigned decimal or hexadecimal number.

■ Status (Return Value)

Table 3.18.3 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0, > 0	SOCKET ID (W) [0-7] (socket is opened successfully)
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.18.4 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.18.5 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1028	No Unused UDP Socket	Execute UDOPEN instruction only if the No Unused UDP Socket relay is OFF.
	M1029	No Unused TCP Socket	
	M1105 to M1120	Socket Open	
	M1121 to M1136	Socket Busy	
	M1073 to M1088	Socket Sending	
	M1089 to M1104	Socket Receiving	

■ Function

Opens a UDP/IP socket. Opening a socket secures system resources required for communications.

Up to 8 UDP/IP sockets can be open concurrently at any one time.

If execution is successful, this instruction returns a socket ID in status, which is to be used in subsequent UDP/IP send and receive instructions. The socket ID is automatically allocated a value from 0 to 7, but not necessarily sequentially from 0.

■ Programming Example

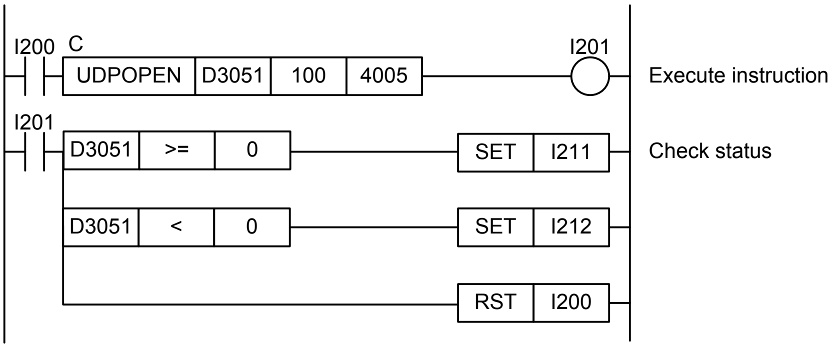


Figure 3.18.1 UDP/IP Open Sample Program

This sample code opens a UDP/IP socket for port number 4005. It specifies the timeout interval as 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	1	Status (socket ID)

3.18.2 UDP/IP Close (UDPCLOSE)

F3SP66 F3SP71
F3SP67 F3SP76

Closes a UDP/IP socket. Once a socket is closed, no more sending or receiving is allowed via the socket unless and until the socket is reopened.

Table 3.18.6 UDP/IP Close

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	UDP/IP Close	UDPCLOSE	C –[UDPCLOSE | | |]–	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

C
UDP/IP Close –[UDPCLOSE | ret | n1 | n2]–

Table 3.18.7 Parameters

Parameter	Description
ret*1	Device for storing return status (W)
n1	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
n2	Socket ID (W) [0-7]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.18.8 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.18.9 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.18.10 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	Execute UDPCLOSE instruction for a socket ID only if its corresponding Socket Busy, Socket Sending and Socket Receiving relays are all off.
	M1029	No Unused TCP Socket	
	M1105 to M1120	Socket Open	
✓	M1121 to M1136	Socket Busy	
✓	M1073 to M1088	Socket Sending	
✓	M1089 to M1104	Socket Receiving	

■ Function

Closes a UDP/IP socket. Once a socket is closed, no more sending or receiving is allowed via the socket.

CAUTION

Issuing multiple close requests for the same socket is not allowed.

■ Programming Example

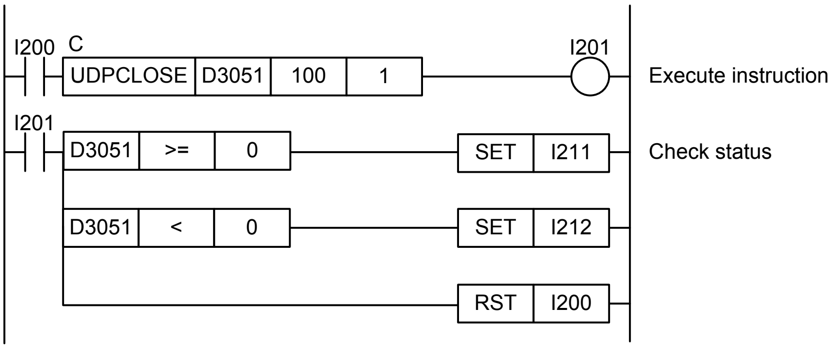


Figure 3.18.2 UDP/IP Close Sample Program

This sample code closes a UDP/IP socket associated with socket ID 1. It specifies the timeout interval as 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.18.3 UDP/IP Send Request (UDPSND)

F3SP66 F3SP71
F3SP67 F3SP76

Sends data stored in a specified device using UDP/IP communications.

Table 3.18.11 UDP/IP Send Request

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro–cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	UDP/IP Send Request	UDPSND	C UDPSND	✓	–	6	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

UDP/IP Send Request C UDPSND ret t s

Table 3.18.12 Parameters

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
	t+1	Socket ID (W) [0-7]
	t+2	Size of send data (W) [0-2048 (bytes)] [*2]
	t+3	Socket destination (W) [[*3] -1 = IP address and port no. (designated by t+4 to t+6) 1-16 = Socket address setting no. in CPU properties]
	t+4	Destination IP address low (W) [$0000-$FFFF] [*3]
	t+5	Destination IP address high (W) [$0000-$FFFF] [*3]
	t+6	Destination port no. (W) [1-65535] [*4]
s		First device of send data (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".
*2: 1472 bytes max. for broadcast communications.
*3: Do not specify destination IP address as 0.0.0.0. Otherwise, the operation or error status will be indefinite.
*4: Word data is handled as an unsigned decimal or hexadecimal number.

■ Status (Return Value)

Table 3.18.13 Status (Return Value)

Offset (word)		Description	
ret	ret+0	> 0	Sent data size [1-2048 (bytes)]
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.18.14 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes
s									✓	✓	✓	✓		✓	✓	#	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."
#: You can specify the constant name of a constant definition but not a normal constant.

■ Resource Relays

Table 3.18.15 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	Execute UDPSND instruction for a socket ID only if both its corresponding Socket Busy and Socket Sending relays are OFF.
	M1029	No Unused TCP Socket	
	M1105 to M1120	Socket Open	
✓	M1121 to M1136	Socket Busy	
✓	M1073 to M1088	Socket Sending	
	M1089 to M1104	Socket Receiving	

■ Function

Sends data stored in a specified device using UDP/IP communications.
You can either specify the destination using a socket address setting number defined in the socket address setup of CPU properties, or specify an IP address and port number directly as instruction parameters. In the latter case, set the socket destination parameter as -1.
For the socket ID parameter, specify the socket ID returned by the UDP/IP Open (UDPOPEN) instruction executed earlier.

Broadcast Transmission:
If UDP broadcast is enabled in the socket setup of CPU properties, you can perform broadcast transmission by setting the lowest byte of the destination IP address to 255. For example, if the network address is 192.168.0.xxx, specify 192.168.0.255 to perform broadcast transmission. Any attempt at broadcast transmission when UDP broadcast is disabled in CPU properties will generate an unknown destination error (error code: -5001).
A target network node may fail to receive a broadcast transmission if it is configured to ignore broadcast transmission or a different IP address is defined as the broadcast address. For details, check with the network administrator.
A maximum send data size of 1472 bytes is allowed for broadcast transmission.

CAUTION

- Concurrent send requests for the same socket are not allowed but concurrent execution of a send request and a receive request is allowed.
- If you specify a socket address setting number with a defined hostname in CPU properties in the instruction, performance will be affected by the time required for DNS resolution.
- The nature of the protocol is such that the instruction will exit normally even if the link is down (e.g. the cable is not connected or the hub is switched off).

■ Programming Example

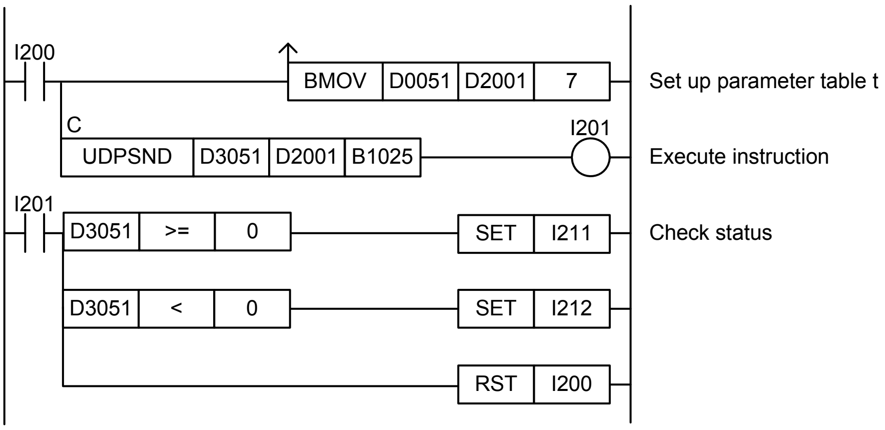


Figure 3.18.3 UDP/IP Send Request Sample Program

This sample code sends to the node associated with socket destination number 4, 200 bytes of data stored in device starting from device B1025.

It specifies ret(=D3051), t(=D2001) and s(=B1025) with t set up as shown in the table below.

Device	Value	Table Parameter
t = D2001	600	Timeout interval (=60 s)
D2002	1	Socket ID (= 1)
D2003	200	Size of send data (= 200 bytes)
D2004	4	Socket destination no. (= 4)
D2005	0	Destination IP address (not required for this example)
D2006		
D2007	0	Destination port no. (not required for this example)

The table below shows the returned status data (ret), assuming normal exit.

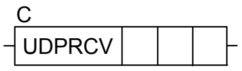
Device	Value	Table Parameter
ret = D3051	200	Status (sent data size)

3.18.4 UDP/IP Receive Request (UDPRCV)

F3SP66 F3SP71
F3SP67 F3SP76

Stores data received from a UDP/IP socket to a specified device.

Table 3.18.16 UDP/IP Receive Request

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro–cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	UDP/IP Receive Request	UDPRCV	 C UDPRCV	✓	–	6	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

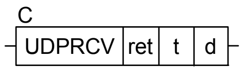
UDP/IP Receive Request 

Table 3.18.17 Parameters

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
	t+1	Socket ID (W) [0-7]
	t+2	Size of receive area (W) [0-4096 (bytes)]
	t+3	Append NULL option (W) [0=no; 1=yes]
	t+4	Buffer option (W) [[*2] 0=Delete packet in receive buffer after retrieval 1=Keep packet in receive buffer after retrieval 2=Check packet size of receive buffer (without receive processing)]
d		First device for storing received data (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".
*2: Buffer option 0 is recommended to avoid buffer overflow.

■ Status (Return Value)

Table 3.18.18 Status (Return Value)

Offset (word)		Description	
ret	ret+0	> 0	Received data size [1-4096 (bytes)] [*5]
		< 0	Error status
	ret+1	CPU properties socket address setting search result (W) [1-16 = Match for both IP address and port no.[*1] 101-116 = Match for IP address only[*2] -1 = No match[*3]]	
	ret+2	Sender IP address low (W) [$0000-$FFFF]	
	ret+3	Sender IP address high (W) [$0000-$FFFF]	
	ret+4	Sender port number (W) [0-65535] [*4]	

*1: If a match is found for the IP address and port number of the sender in the socket address setup of CPU properties, the corresponding setting number is returned.
*2: If a match is found for only the IP address of the sender in the socket address setup of CPU properties, the corresponding setting number + 100 is returned.
*3: If no match is found for the IP address of the sender in the socket address setup of CPU properties, -1 is returned.
*4: Word data is handled as an unsigned decimal or hexadecimal number.
*5: Received data size includes any NULL byte appended according to the Append NULL option.

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.18.19 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con–stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes
d									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.18.20 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	Execute UDPRCV instruction for a socket ID only if both its corresponding Socket Busy and Socket Receiving relays are OFF.
	M1029	No Unused TCP Socket	
	M1105 to M1120	Socket Open	
✓	M1121 to M1136	Socket Busy	
	M1073 to M1088	Socket Sending	
✓	M1089 to M1104	Socket Receiving	

■ Function

Stores data received from a UDP/IP socket to a specified device. For the socket ID parameter, specify the socket ID returned by the UDP/IP Open (UDPOPEN) instruction executed earlier. The size of the data stored to device is returned in status.

You can specify whether to append a NULL byte behind received data using the Append Null Option.

If no data is received in the buffer when the instruction is executed, the instruction waits for data to arrive. However, if the buffer option parameter is set to 2 (= check packet size of receive buffer), the instruction completes execution without entering wait state even if no data has been received. If a timeout interval is specified and no data is received within the specified time, the instruction exits from wait state, holds the result signal to ON and returns a timeout error in status.

If the buffer option parameter is 0, the data packet in the receive buffer is discarded after retrieval regardless of the size of receive area.

Broadcast transmission:
If broadcast transmission is enabled in the socket setup of CPU properties, the instruction receives broadcast packets.

CAUTION

- Concurrent receive requests for the same socket are not allowed but concurrent execution of a send request and a receive request is allowed.
- You should specify the buffer option as 0 (= delete packet after retrieval) unless there is a special reason not to do so. Specifying a non-zero buffer option means that the receive buffer is not emptied and this may result in buffer overflow.
- If you specify a socket address setting number with a defined hostname in CPU properties in the instruction, performance will be affected by the time required for DNS resolution.

■ Programming Example

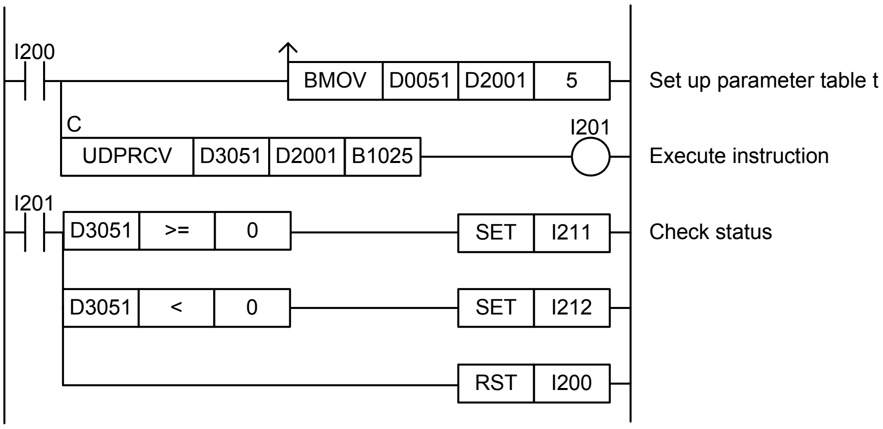


Figure 3.18.4 UDP/IP Receive Request Sample Program

This sample code receives data from the UDP/IP socket associated with socket ID 4, and stores the received data to device, starting from device B1025.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as shown in the table below.

Device	Value	Table Parameter
t = D2001	6000	Timeout interval (= 600 s)
D2002	4	Socket ID (=4)
D2003	2048	Size of receive area (= 2048 bytes)
D2004	0	Append NULL option (= No)
D2005	0	Buffer option (= Delete packet after retrieval (recommended))

The table below shows an example of the returned status data (ret), assuming that a 520-byte packet is received from an IP address (192.168.0.67:10456), which is not registered in the socket address setup of CPU properties.

Device	Value	Table Parameter
ret = D3051	520	Status
D3052	-1	Sender socket address setting number
D3053	$0043	Sender IP address
D3054	$C0A8	
D3055	10456	Sender port number

3.19 TCP/IP Socket Communications Instructions

3.19.1 TCP/IP Open (TCPOPEN)

F3SP66	F3SP71
F3SP67	F3SP76

Opens a TCP/IP socket.

Table 3.19.1 TCP/IP Open

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro–cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	TCP/IP Open	TCPOPEN	C TCPOPEN	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

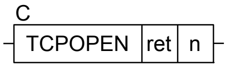
TCP/IP Open 

Table 3.19.2 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.19.3 Status (Return Value)

Offset (word)	Description	
ret	> 0	Socket ID (W) [8-15] (socket is successfully opened)
	< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.19.4 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.19.5 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	Execute TCPOPEN instruction only if the No Unused TCP Socket relay is OFF.
✓	M1029	No Unused TCP Socket	
	M1105 to M1120	Socket Open	
	M1121 to M1136	Socket Busy	
	M1073 to M1088	Socket Sending	
	M1089 to M1104	Socket Receiving	

■ Function

Opens a TCP/IP socket. Opening a socket secures system resources required for communications to enable execution of the TCP/IP Connect Request (TCPCNCT) instruction or the TCP/IP Listen Request (TCPLISN) instruction.

Up to 8 TCP/IP sockets can be open concurrently at any one time. The socket ID is automatically allocated a value from 8 to 15, but not necessarily in any order.

If execution is successful, this instruction returns a socket ID in status, which is to be used in subsequent TCP/IP Connect Request (TCPCNCT) instructions and TCP/IP Listen Request (TCPLISN) instructions. When connected as a client (after executing a TCPCNCT instruction), the same socket ID can also be used in TCP/IP send and receive instructions.

CAUTION

The allocated socket ID can be any value between 8 and 15. Note that it does not start from 0.

■ Programming Example

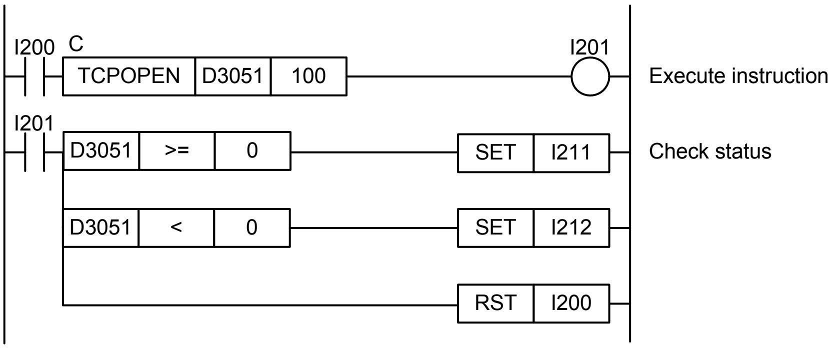


Figure 3.19.1 TCP/IP Open Sample Program

This sample code opens a TCP/IP socket. It specifies the timeout interval as 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

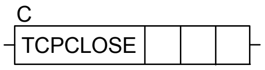
Device	Value	Table Parameter
ret = D3051	8	Status (socket ID)

3.19.2 TCP/IP Close (TCPCLOSE)

F3SP66 F3SP71
F3SP67 F3SP76

Closes a TCP/IP socket. Once a socket is closed, the socket ID can no longer be used.

Table 3.19.6 TCP/IP Close

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro–cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	TCP/IP Close	TCPCLOSE	 C TCPCLOSE	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

TCP/IP Close

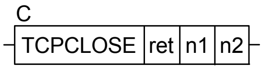


Table 3.19.7 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
n2	Socket ID (W) [8-15]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.19.8 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.19.9 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.19.10 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	Execute TCPCLOSE instruction for a socket ID only if its corresponding Socket Busy, Socket Sending and Socket Receiving relays are all OFF.
	M1029	No Unused TCP Socket	
	M1105 to M1120	Socket Open	
✓	M1121 to M1136	Socket Busy	
✓	M1073 to M1088	Socket Sending	
✓	M1089 to M1104	Socket Receiving	

■ Function

Closes a TCP/IP socket. Once a socket is closed, the socket ID can no longer be used.

CAUTION

- Issuing multiple close requests for the same socket is not allowed.
- Depending on the state of the destination, cancellation may sometimes take a long time to complete.

■ Programming Example

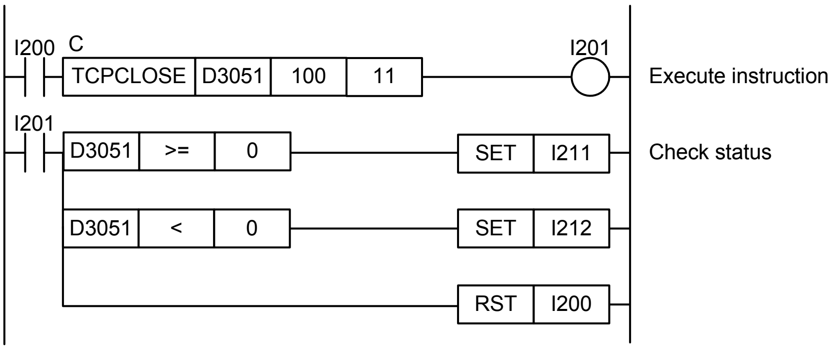


Figure 3.19.2 TCP/IP Close Sample Program

This sample code closes a TCP/IP socket associated with socket ID 11. It specifies the timeout interval as 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

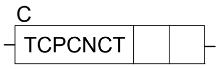
Device	Value	Table Parameter
ret = D3051	0	Status

3.19.3 TCP/IP Connect Request (TCPCNCT)

F3SP66 F3SP71
F3SP67 F3SP76

Issues a connection request to a TCP/IP server (a node which is waiting for connection or, in other words, listening), and establishes a connection if permitted to do so.

Table 3.19.11 TCP/IP Connect Request

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro–cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	TCP/IP Connect Request	TCPCNCT		✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

TCP/IP Connect Request

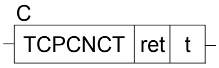


Table 3.19.12 Parameters

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
	t+1	Socket ID (W) [8-15]
	t+2	Socket destination (W) [[*2] -1 = IP address and port no. (designated by t+3 to t+5) 1-16 = Socket address setting no. in CPU properties]
	t+3	Destination IP address low (W) [$0000-$FFFF] [*2]
	t+4	Destination IP address high (W) [$0000-$FFFF] [*2]
	t+5	Destination port no. (W) [1-65535] [*3]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".
*2: Do not specify destination IP address as 0.0.0.0. Otherwise, a parameter error status will be returned.
*3: Word data is handled as an unsigned decimal or hexadecimal number.

■ Status (Return Value)

Table 3.19.13 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.19.14 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.19.15 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	Execute TCPCNCT instruction for a socket ID only if its corresponding Socket Busy relay is OFF.
	M1029	No Unused TCP Socket	
	M1105 to M1120	Socket Open	
✓	M1121 to M1136	Socket Busy	
	M1073 to M1088	Socket Sending	
	M1089 to M1104	Socket Receiving	

■ Function

Prepares for communications as a client by issuing a connection request to a TCP/IP server (a node which is waiting for connection or, in other words, listening), and establishing a connection if permitted to do so.

You can either specify the destination using a socket address setting number defined in the socket address setup of CPU properties, or specify an IP address and port number directly as instruction parameters.

CAUTION

- Issuing multiple connection requests for the same socket is not allowed.
- Issuing a connection request to the address of a node itself is not allowed. Doing so will generate an unknown destination error (error code -5001).
- If a connection error code (-5000) or unknown destination error code (-5001) is returned in status, you must execute the TCP/IP Close (TCPCLOSE) instruction. By nature of a general protocol stack, the socket ID used transits to an invalid state.
- Sometimes communications may fail even after a TCP/IP Connect Request (TCPCNCT) exits normally. This may happen if the server side (listening side) returns a successful reply to a connection request but subsequently encounters an error such as depletion of available sockets and eventually fails to establish a transmission channel.

■ Programming Example

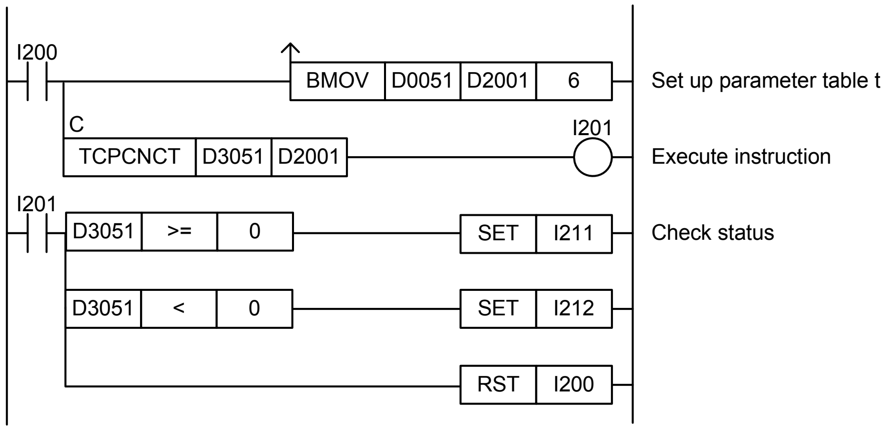


Figure 3.19.3 TCP/IP Connect Request Sample Program

This sample code issues a connection request by directly specifying 192.168.0.6:20677 as the destination address in the instruction.

It specifies ret(=D3051) and t(=D2001) with t set up as shown in the table below.

Device	Value	Table Parameter
t = D2001	600	Timeout interval (=60 s)
D2002	12	Socket ID (= 12)
D2003	-1	Socket destination no.(=direct designation)
D2004	$0006	Destination IP address (= 192.168.0.6)
D2005	$C0A8	
D2006	20677	Destination port no. (= 20677)

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret =D3051	0	Status

3.19.4 TCP/IP Listen Request (TCPLISN)

F3SP66 F3SP71
F3SP67 F3SP76

Waits for connection request from any TCP/IP client, and establishes connection if a request is received.

Table 3.19.16 TCP/IP Listen Request

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro–cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	TCP/IP Listen Request	TCPLISN	C TCPLISN	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

C

TCP/IP Listen Request TCPLISN ret t

Table 3.19.17 Parameters

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
	t+1	Socket ID (W) [8-15]
	t+2	My port number (W) [1-65535][*2 *3]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

*2: Do not specify my port number as 12289, 12290, 12291, 12305 or 12307 as these numbers are used by the higher-level link service and remote programming service.

*3: Word data is handled as an unsigned decimal or hexadecimal number.

■ Status (Return Value)

Table 3.19.18 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	New socket ID for sending and receiving (W) [8-15]	
	ret+2	CPU properties socket address setting search result (W) [1-16 = Match for both IP address and port no.[*1] 101-116 = Match for IP address only[*2] -1 = No match[*3]]	
	ret+3	Source IP address low (W) [$0000-$FFFF]	
	ret+4	Source IP address high (W) [$0000-$FFFF]	
	ret+5	Source port number (W) [0-65535] [*4]	

*1: If a match is found for the IP address and port number of the source in the socket address setup of CPU properties, the corresponding setting number is returned.

*2: If a match is found for only the IP address of the source in the socket address setup of CPU properties, the corresponding setting number + 100 is returned.

*3: If no match is found for the IP address of the source in the socket address setup of CPU properties, -1 is returned.

*4: Word data is handled as an unsigned decimal or hexadecimal number.

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.19.19 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.19.20 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	Execute TCPLISN instruction for a socket ID only if both the No Unused TCP Socket relay and its corresponding Socket Busy relay are OFF.
✓	M1029	No Unused TCP Socket	
	M1105 to M1120	Socket Open	
✓	M1121 to M1136	Socket Busy	
	M1073 to M1088	Socket Sending	
	M1089 to M1104	Socket Receiving	

■ Function

Prepares for communications as a server. Waits for connection request from any TCP/IP client, and establishes connection if a request is received.

When the instruction successfully establishes a connection with a client, it returns a new socket ID in status. The new socket ID is to be used for subsequent sending and receiving. The socket ID specified as a parameter of this instruction is not used for sending and receiving and therefore can be reused to listen for a connection request from a different client by re-executing this instruction. In other words, the same socket (=same port number) can be used to listen for connection requests from multiple clients. After connection, data can be sent to and received from multiple clients.

CAUTION

- Issuing multiple connection requests for the same socket is not allowed.
- When sending data to and receiving data from TCP/IP clients, use the socket ID returned in status by this instruction, but not the socket ID that is specified as a parameter of this instruction.

■ Programming Example

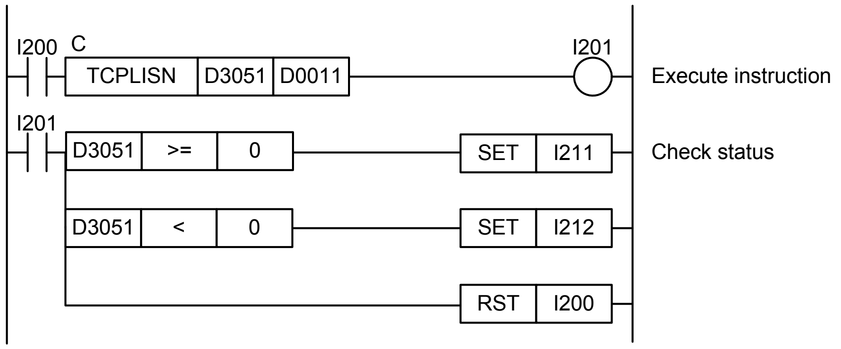


Figure 3.19.4 TCP/IP Connect Request Sample Program

This sample code listens for a connection request from any client. It assumes that the required parameter values (timeout interval, the socket ID to be used by the TCPLISN instruction, my port number) are already stored in the device area starting from device D0011. It specifies D3051 as the first device for storing the returned status and new socket ID for sending and receiving.

The table below shows an example of the returned status data (ret), assuming that that the instruction exited normally after processing a connection request from a peer (192.168.0.9: 10456), which is registered as socket address setting 3 in CPU properties.

Device	Value	Table Parameter
ret = D3051	0	Status
D3052	12	New socket ID for sending and receiving
D3053	3	Socket address setting no. of source
D3054	$0009	Source IP address
D3055	$C0A8	
D3056	10456	Source port number

3.19.5 TCP/IP Send Request (TCPSND)

F3SP66 F3SP71
F3SP67 F3SP76

Sends data stored in a specified device using TCP/IP communications.

Table 3.19.21 TCP/IP Send Request

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro–cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	TCP/IP Send Request	TCPSND	C TCPSND	✓	–	6	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

TCP/IP Send Request
C
TCPSND | ret | t | s

Table 3.19.22 Parameters

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
	t+1	Socket ID (W) [8-15]
	t+2	Size of send data (W) [0-2048 (bytes)]
s		First device of send data (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.19.23 Status (Return Value)

Offset (word)		Description	
ret	ret+0	> 0	Sent data size [1-2048 (bytes)]
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.19.24 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes
s									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.19.25 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	Execute TCPSND instruction for a socket ID only if both its corresponding Socket Busy relay and Socket Sending relay are OFF.
	M1029	No Unused TCP Socket	
	M1105 to M1120	Socket Open	
✓	M1121 to M1136	Socket Busy	
✓	M1073 to M1088	Socket Sending	
	M1089 to M1104	Socket Receiving	

■ Function

Sends data stored in a specified device using TCP/IP communications.

Specify the destination as a socket ID. For a client, specify the socket ID returned by the TCP/IP Open (TCPOPEN) instruction. For a server, specify the socket ID returned by the TCP/IP Listen Request (TCPLISN) instruction.

CAUTION

Concurrent send requests for the same socket are not allowed but concurrent execution of a send request and a receive request is allowed.

■ Programming Example

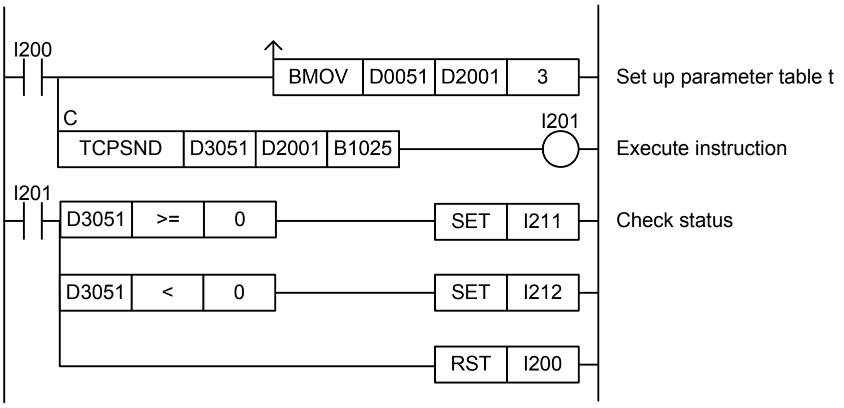


Figure 3.19.5 TCP/IP Send Request Sample Program

This sample code sends to the node connected to TCP/IP socket ID 12, 212 bytes of data stored in device starting from device B1025.

It specifies ret(=D3051), t(=D2001) and s(=B1025) with t set up as shown in the table below.

Device	Value	Table Parameter
t = D2001	600	Timeout interval (=60 s)
D2002	12	Socket ID (= 12)
D2003	212	Size of send data (= 212 bytes)

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	212	Status (sent data size)

3.19.6 TCP/IP Receive Request (TCPRCV)

F3SP66 F3SP71
F3SP67 F3SP76

Stores data received from a TCP/IP socket to a specified device.

Table 3.19.26 TCP/IP Receive Request

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro–cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	TCP/IP Receive Request	TCPRCV	C –[TCPRCV | | |]–	✓	–	6	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

```
                              C
TCP/IP Receive Request  –[TCPRCV|ret| t | d ]–
```

Table 3.19.27 Parameters

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
	t+1	Socket ID (W) [8-15]
	t+2	Size of receive area (W) [0-2048 (bytes)]
	t+3	Append NULL option (W) [*2] [0=no; 1=yes]
	t+4	Buffer option (W) [*3 0 = Delete read data in receive buffer after retrieval (normal mode) 1 = Keep read data in receive buffer after retrieval 2 = Check data size receivable from receive buffer 3 = Delete data in receive buffer without retrieval 4 = Delete read data in receive buffer after retrieval (Auto increment mode) 5 = Delete data in receive buffer after retrieval of the specified size of data (Fixed-size mode)[*4]]
d		First register for receive area (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".
*2: Any NULL byte appended according to the Append NULL option should be included in the size of receive area.
*3: Buffer options 0, 4 and 5 are recommended to avoid buffer overflow.
*4: Not available for F3SP7□-□N and F3SP6□-□S.

■ Status (Return Value)

Table 3.19.28 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0, > 0	Received or receivable data size [0-2048 (bytes)[*1]
		< 0	Error status

*1: Received data size includes any NULL byte appended according to Append NULL option.

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.19.29 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes
d									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.19.30 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	Execute TCPRCV instruction for a socket ID only if both its corresponding Socket Busy relay and Socket Receiving relay are OFF.
	M1029	No Unused TCP Socket	
	M1105 to M1120	Socket Open	
✓	M1121 to M1136	Socket Busy	
	M1073 to M1088	Socket Sending	
✓	M1089 to M1104	Socket Receiving	

■ Function

Stores data received from a TCP/IP socket to a specified device. After execution, the size of data stored to device is returned in status. You can specify whether to append a NULL byte behind received data using the Append Null Option.

If no data is received in the buffer when the instruction is executed, the instruction waits for data to arrive. However, if the buffer option parameter is set to 2 (= check data size receivable from receive buffer), the instruction completes execution without entering wait state even if no data has been received. If a timeout interval is specified and no data is received within the specified time, the instruction exits from wait state, holds the result signal to ON and returns a timeout error in status.

Auto increment mode

Specifying 4 for the buffer option parameter selects auto increment mode in which the instruction automatically increments parameter d (first device for receive area) by the received data size so that the entire data received through multiple instruction executions can be stored contiguously to devices.

Using auto increment mode

Auto increment mode must be used together with normal mode. Specify normal mode for the first execution of TCPRCV and specify auto increment mode for the second and subsequent executions. This way, the received data will be stored contiguously to device.

Do not modify the value of parameter d (first device of receive area) for the second and subsequent executions as the byte offset is computed automatically by the instruction.

Canceling auto increment mode

To reset the byte offset to zero, you can either:

- Execute the instruction by specifying any value other than 2 and 4 for the buffer option;
- Execute the instruction with a modified value of parameter d (first address of receive area)

Fixed-size mode

Specifying 5 for the buffer option parameter selects fixed-size mode in which the instruction is finished after read data in the receive buffer is removed due to the amount of received data reaching the specified receive area size.

If a timeout occurs or the instruction is canceled while the size of received data has not reached the specified receive area size, the data in the receive buffer remains in the buffer, resulting in an insufficient receive data area.

CAUTION

- Concurrent receive requests for the same socket are not allowed but concurrent execution of a send request and a receive request is allowed.
- Selecting a buffer option that does not delete data in the receive buffer may lead to buffer overflow.
- If auto increment mode is specified and the byte offset (= accumulated received size) is equal to or exceeds the specified size of receive area parameter, a "Specified Size/Times Processed" (-2003) status is returned.
- If fixed-size mode is specified and the connection to the destination is closed while the receive area size has not been reached, the contents of the last received data cannot be guaranteed.

■ Programming Example

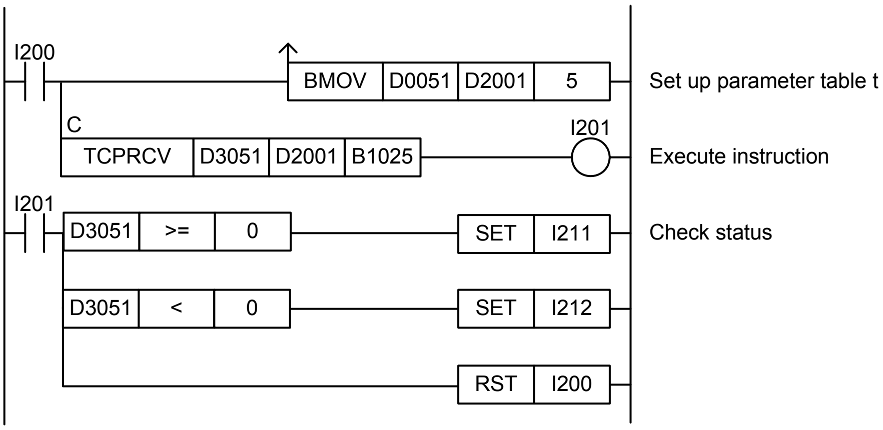


Figure 3.19.6 TCP/IP Receive Request Sample Program

This sample code waits for data from the node connected to TCP/IP socket ID 12, and stores the received data to device, starting from device B1025.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as shown in the table below.

Device	Value	Table Parameter
t = D2001	6000	Timeout interval (= 600 s)
D2002	12	Socket ID (=12)
D2003	2048	Size of receive area (= 2048 bytes)
D2004	0	Append NULL option (= No)
D2005	0	Buffer option (= Normal mode (recommended))

The table below shows the returned status data (ret), assuming normal exit after receiving 608 bytes of data.

Device	Value	Table Parameter
ret = D3051	608	Status (received data size)

3.19.7 Socket Option (SOCKOPT)

F3SP71-4S
F3SP76-7S

Specifies the socket option.

Table 3.19.31 Socket Option

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	—	Socket Option	SOCKOPT	C ─[SOCKOPT │ │ │]─	✓	—	6	—	—

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

```
                C
Socket Option─[SOCKOPT│ret│ t │ s ]─
```

Table 3.19.32 Parameters

Parameter		Description
ret [*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
	t+1	SOCKET ID(W) [8-15]
	t+2	Setting No. (W) [1 = Disable Nagle algorithm 2 = Disable delayed ACK]
s		Setup data (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.19.33 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.19.34 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes
s									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.19.35 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	Execute SOCKOPT instruction for a socket ID only if its corresponding Socket Open is ON, and Socket Busy, Socket Sending, and Socket Receiving are OFF.
	M1029	No Unused TCP Socket	
✓	M1105～M1120	Socket Open	
✓	M1121～M1136	Socket Busy	
✓	M1073～M1088	Socket Sending	
✓	M1089～M1104	Socket Receiving	

■ Function

Sets the socket option for an opened socket and changes how the socket works. It uses the setting number to specify the function and the setup data to specify how the socket functions.

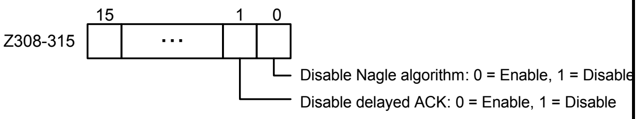
You can configure the setting for disabling the Nagle algorithm by specifying 0, which means "enabled" and is the default value, or 1, which means "disabled", in word for the setup data.

You can configure the setting for disabling the delayed ACK by specifying 0, which means "enabled" and is the default value, or 1, which means "disabled", in word for the setup data.

If incorrect setup data is specified, the socket option is not updated, causing a data processing error (-9015).

You can check the states of the Nagle algorithm and the delayed ACK with the special registers for socket status (Z308 - Z315).

Table 3.19.36 Special Registers for Socket Status

SOCKET ID	Special Registers for Socket Status	Description
8	Z308	 Z308-315 15 1 0 … Disable Nagle algorithm: 0 = Enable, 1 = Disable Disable delayed ACK: 0 = Enable, 1 = Disable
9	Z309	
10	Z310	
11	Z311	
12	Z312	
13	Z313	
14	Z314	
15	Z315	

The socket option is returned to the default value by closing the socket that corresponds to the socket ID.

SEE ALSO

The Nagle algorithm, which is defined in RFC896, improves the efficiency of data transfer when small-sized data is transferred multiple times in TCP/IP communication. The algorithm can achieve this by delaying data transmission until acknowledgments are received, or a data packet that is equivalent to MSS (Max Segment Size) can be sent when the send buffer has some data for which acknowledgments have not yet been received.

The delayed ACK technique is defined in RFC1122. It reduces the number of acknowledgements in TCP/IP communication in order to improve the efficiency of data transfer by setting the ACK flag on a data packet to be sent or by waiting for a certain period of time before returning an ACK, instead of returning the ACK immediately after data is received from a sender.

Both functions are enabled by default and should be used in normal conditions. Disable them only if required.

■ Programming Example

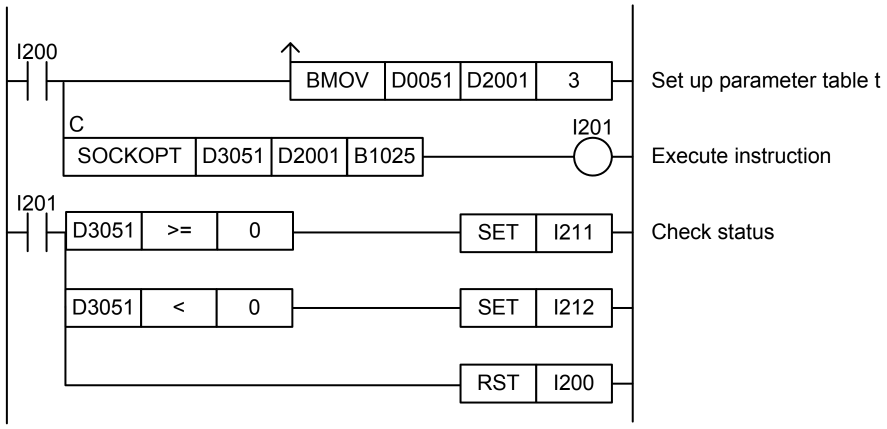


Figure 3.19.7 Socket Option Sample Program

This sample code disables the Nagle algorithm for SOCKET ID12.

It specifies ret(=D3051), t(=D2001) and s(=B1025) with t and s specified as shown in the table below.

Device	Value	Table Parameter
t = D2001	6000	Timeout interval (=60 s)
D2002	12	Socket ID (= 12)
D2003	1	Disable Nagle algorithm
s = B1025	1	Disable

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.20 FTP Client Instruction Specifications

3.20.1 FTP Client Open (FTPOPEN)

F3SP66 F3SP71
F3SP67 F3SP76

Runs FTP client and connects to an FTP server.

Table 3.20.1 FTP Client Open

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Open	FTPOPEN	C FTPOPEN	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

FTP Client Open C FTPOPEN ret n1 n2

Table 3.20.2 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]
n2	FTP client address setting no. (w)[1-4][*2]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".
*2: Do not specify 0.0.0.0 for the Destination IP address in FTP client address setup.

■ Status (Return Value)

Table 3.20.3 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.20.4 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available as Instruction Parameters."

Resource Relays

Table 3.20.5 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

Function

Runs FTP client and connects to an FTP server. If connection is successful, the FTP client is ready to send and receive files. The FTP server must also be running at the destination.

You can select the destination FTP server by specifying a setting number (1-4) of FTP client address setup for instruction parameter n2. In the FTP client address setup, specify one or more FTP server destinations (IP address or hostname), along with port number, account name and password.

SEE ALSO

For details on FTP client address setup, see "FTP Client Address Setup" of Subsection A9.5.5, "FTP Client Setup" of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E) or "FTP Client Address Setup" of Subsection A9.5.5, "FTP Client Setup" of "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

To change the destination when an FTP client is running, terminate the FTP client using the FTP Client Quit (FTPQUIT) instruction and re-execute the FTPOPEN instruction.

The port number used by the FTP client itself is automatically assigned by the system.

CAUTION

Only one FTP client service can be running on a CPU module at any one time.

■ Programming Example

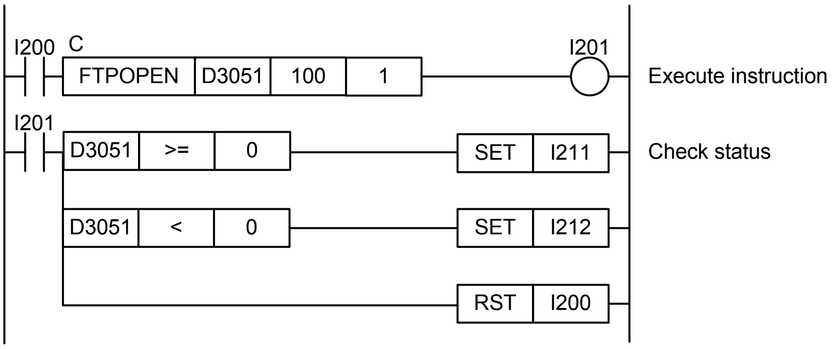


Figure 3.20.1 Example of an FTP Client Open Program

This sample code connects to the FTP server designated by FTP client address setting number 1. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.20.2 FTP Client Quit (FTPQUIT)

F3SP66 F3SP71
F3SP67 F3SP76

Disconnects an FTP client (CPU module) started by the FTP Client Open (FTPOPEN) instruction from its connected FTP server, and terminates the FTP client service.

Table 3.20.6 FTP Client Quit

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Quit	FTPQUIT	C ┤FTPQUIT│ │ ├	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

C
FTP Client Quit ┤FTPQUIT│ret│n├

Table 3.20.7 Parameters

Parameter	Description
ret*1	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.20.8 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.20.9 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available as Instruction Parameters."

Resource Relays

Table 3.20.10 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

Function

Disconnects an FTP client (CPU module) started by the FTP Client Open (FTPOPEN) instruction from its connected FTP server, and terminates the FTP client service.

CAUTION

Depending on the status of the remote node, termination may take a long time.

Programming Example

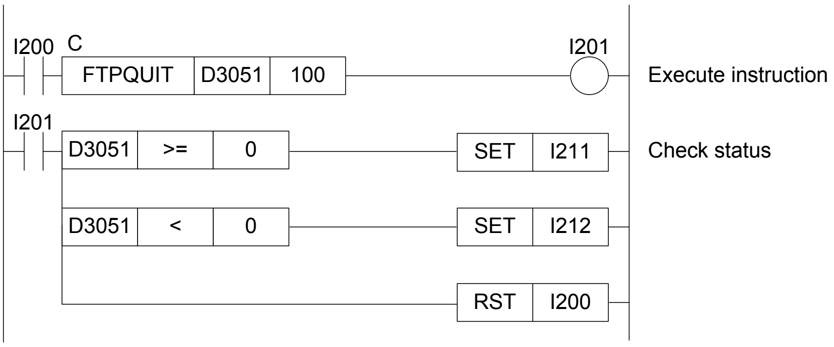


Figure 3.20.2 Example of an FTP Client Quit Program

This sample program terminates an FTP client. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

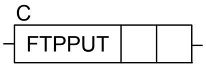
Device	Value	Table Parameter
ret = D3051	0	Status

3.20.3 FTP Client Put File (FTPPUT)

F3SP66 F3SP71
F3SP67 F3SP76

Sends a file stored on the disk of the CPU module to an FTP server.

Table 3.20.11 FTP Client Put File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Put File	FTPPUT		✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

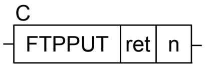
FTP Client Put File 

Table 3.20.12 Parameters

Parameter	Description
ret*1	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.20.13 Text Parameters

Parameter		Description
1	s	Source file pathname *2
2	d	Destination file pathname*1*2

*1: If the value is NULL, the sent data will be stored in the current directory of the FTP server with the same filename as the source filename.

*2: If a wildcard pattern is specified for the source file pathname s, the destination file pathname d must be a directory.

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Section 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.20.14 Status (Return Value)

Offset (word)		Description	
ret	ret+0	> 0	Number of files sent (W)[1-32767]
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.20.15 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available as Instruction Parameters."

■ Resource Relays

Table 3.20.16 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Sends a file stored on the disk of the CPU module to the FTP server.

Multiple files can be sent by including wildcard characters ('*', '?') in the file name. In such situations, even if an error occurs at the FTP server end during file transfer, processing of un-transferred files continues.

At the end of transfer, the number of transferred files is returned and stored in Status.

CAUTION

This instruction cannot be executed concurrently with other FTP client instructions.

■ Programming Example

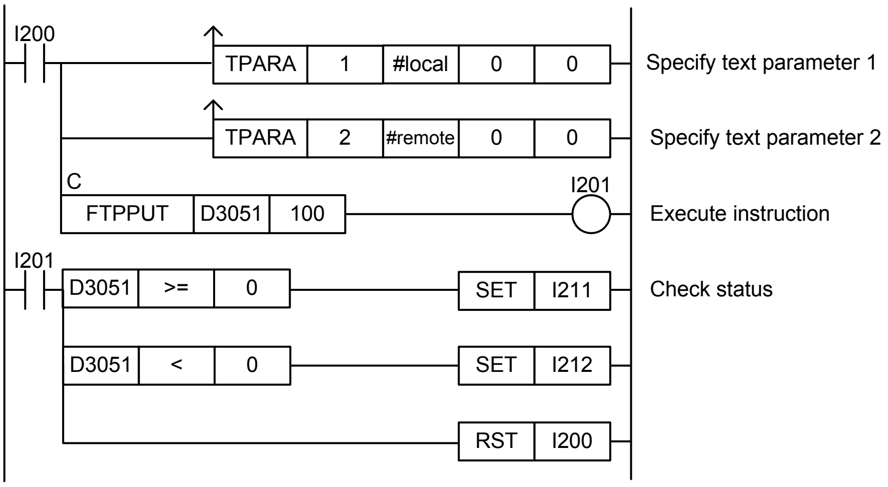


Figure 3.20.3 Example of an FTP Client Put File Program

This sample code transfers a file on the FTP client with file pathname defined by constant name "#local" to the FTP server file pathname defined by constant name "#remote". The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

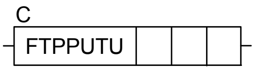
Device	Value	Table Parameter
ret = D3051	1	Status

3.20.4 FTP Client Put Unique File (FTPPUTU)

F3SP66 F3SP71
F3SP67 F3SP76

Sends a file on the module disk to the FTP server to be stored with a unique filename.

Table 3.20.17 FTP Client Put Unique File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Put Unique File	FTPPUTU		✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

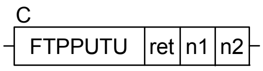
FTP Client Put Unique File 

Table 3.20.18 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]
n2	Filename return option (W) [0 = Filename is not returned. 1 = Filename is returned.]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.20.19 Text Parameters

Parameter		Description
1	s1	Source file pathname
2	s2	(Reserved)[*1]

*1: Always specify NULL for this system-reserved parameter.

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Section 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.20.20 Status (Return Value)

Offset (word)		Description	
ret	ret+0	> 0	Number of files sent (W)[1]
		< 0	Error status
	ret+1 -17	Destination file name determined by FTP server (0-32 characters) Appended with a trailing NULL character.	

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.20.21 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available as Instruction Parameters."

■ Resource Relays

Table 3.20.22 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Sends from the module to the FTP server a file, which is to be stored in the current directory of the FTP server with a unique filename automatically determined by the FTP server according to its specifications. The original filename of the sent file on the FTP client is ignored during file naming.

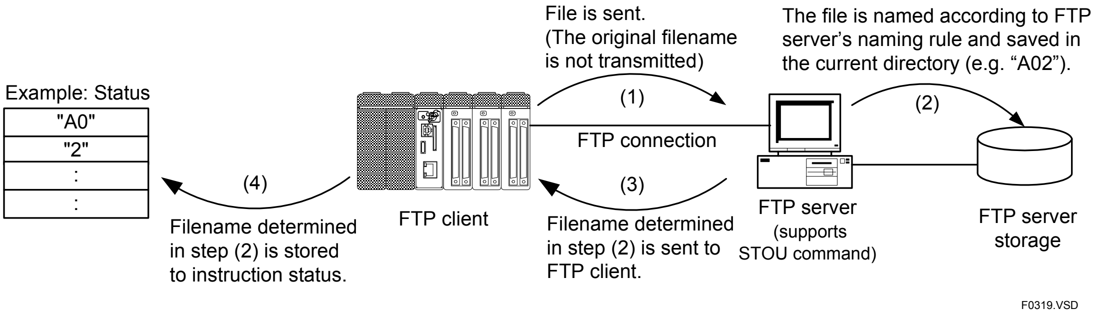


Figure 3.20.4 Sending a File to FTP Server

The destination filename on the FTP server can be returned as status data using the Filename Return Option parameter. The filename is returned without its pathname. The maximum filename length that can be returned is 32 characters and any characters exceeding the limit will be discarded.

Always specify NULL for the system-reserved s2 text parameter.

Wildcard characters must not be used with this instruction.

TIP

Filename extraction processing of the module follows the RFC1123 specifications. It will work correctly even if the reply from the FTP server does not contain the "FILE:" string. In this case, the module extracts and outputs the last word from the reply text, which therefore may include other characters preceding the filename. (The reply from the IIS of Microsoft Windows does not contain the "FILE:" string so the last word will be extracted as the filename.)

CAUTION

- This instruction cannot be executed concurrently with other FTP client instructions.
- Wildcard characters must not be used with this instruction.

■ Programming Example

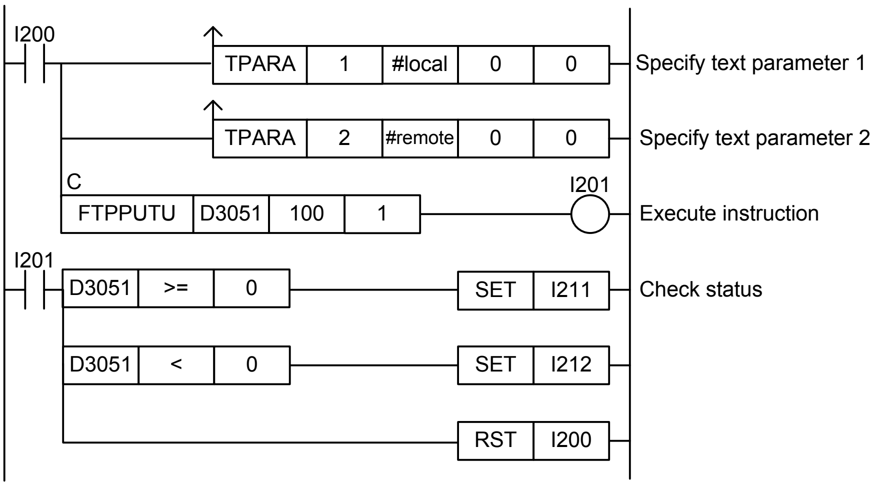


Figure 3.20.5 Example of an FTP Client Put Unique File Program

This sample code uses the FTPPUTU instruction to send a file on the FTP client with file pathname defined by constant name "#local" to be stored in the current directory of the FTP server with a unique name. The timeout interval is set to 100 (10 s); the Filename Return Option is set to 1.

```
    #local = "\ramdisk\mydir\myfile.csv"
```

The table below shows the returned status, assuming normal exit and a destination filename on the FTP server of "A000004.tmp".

Device	Value	Table Parameter
ret = D3051	1	Status
D3052	“A0”	File name determined by the FTP server (A000004.tmp)
D3053	“00”	
D3054	“00”	
D3055	“4.”	
D3056	“tm”	
D3057	“p”+NULL	

3.20.5 FTP Client Append File (FTPAPEND)

F3SP66 F3SP71
F3SP67 F3SP76

Sends a file on the module disk to be appended to a specified file on the FTP server.

Table 3.20.23 FTP Client Append File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Append File	FTPAPEND	C FTPAPEND	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

FTP Client Append File C FTPAPEND | ret | n

Table 3.20.24 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.20.25 Text Parameters

Parameter		Description
1	s	Source file pathname
2	d	Destination file pathname[*1]

*1: If the value is NULL, the sent data will be stored in the current directory of the FTP server with the same filename as the source filename.

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Section 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.20.26 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.20.27 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available as Instruction Parameters."

■ Resource Relays

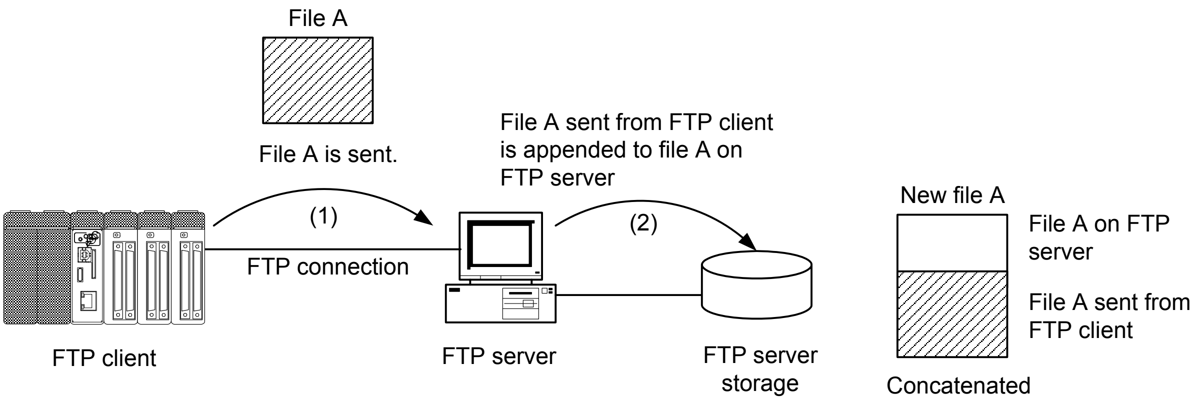
Table 3.20.28 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Sends a file on the module disk to be appended to a specified file on the FTP server.

If the specified destination filename exists on the FTP server, the sent file is appended to the existing file. Otherwise, this instruction behaves the same way as the FTP Client Put File (FTPPUT) instruction.



F0320.VSD

Figure 3.20.6 Appending a File to a Specified File on the FTP Server

CAUTION

- This instruction cannot be executed concurrently with other FTP client instructions.
- Wildcard characters must not be used with this instruction.

■ Programming Example

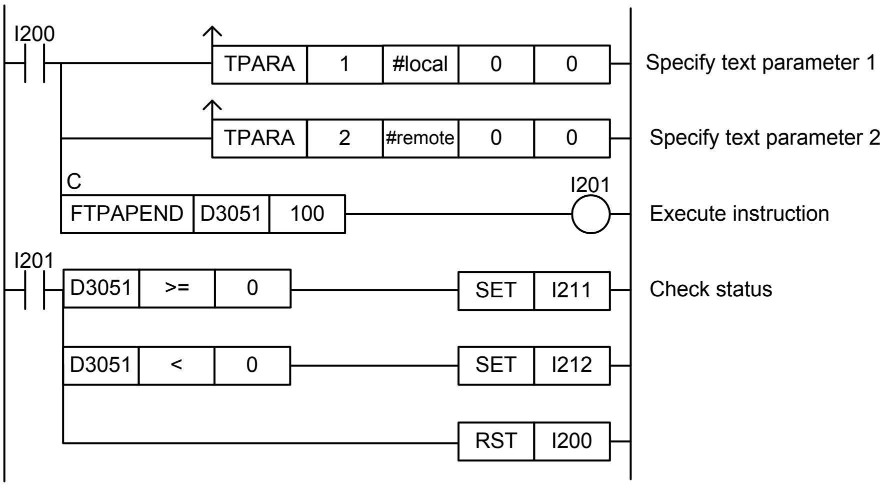


Figure 3.20.7 Example of an FTP Client Append File Program

This sample code uses the FTPAPEND instruction to send a file on the FTP client designated by constant name "#local" to be appended to a file on the FTP server designated by constant name "#remote". The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

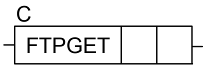
Device	Value	Table Parameter
ret = D3051	0	Status

3.20.6 FTP Client Get File (FTPGET)

F3SP66 F3SP71
F3SP67 F3SP76

Gets a file from the FTP server and saves it to the disk of the module.

Table 3.20.29 FTP Client Get File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Get File	FTPGET		✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

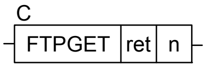
FTP Client Get File 

Table 3.20.30 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.20.31 Text Parameters

Parameter		Description
1	s	Source file pathname [*2]
2	d	Destination file pathname [*1*2]

*1: If the value is NULL, the received data will be stored in the current directory of the FTP client with the same filename as the source filename.

*2: If a wildcard pattern is specified for the source file pathname s, the destination file pathname d must be a directory.

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Section 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.20.32 Status (Return Value)

Offset (word)		Description	
ret	ret+0	> 0	Number of files received (W) [1-32767]
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.20.33 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available as Instruction Parameters."

■ Resource Relays

Table 3.20.34 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Gets a file from the FTP server and saves it to the disk of the module.

Multiple files can be retrieved by including wildcard characters ('*', '?') in the file name. In such situations, even if an error occurs at the FTP server end during file transfer, processing of un-transferred files continues.

At the end of transfer, the number of transferred files is returned and stored in Status.

CAUTION

This instruction cannot be executed concurrently with other FTP client instructions.

■ Programming Example

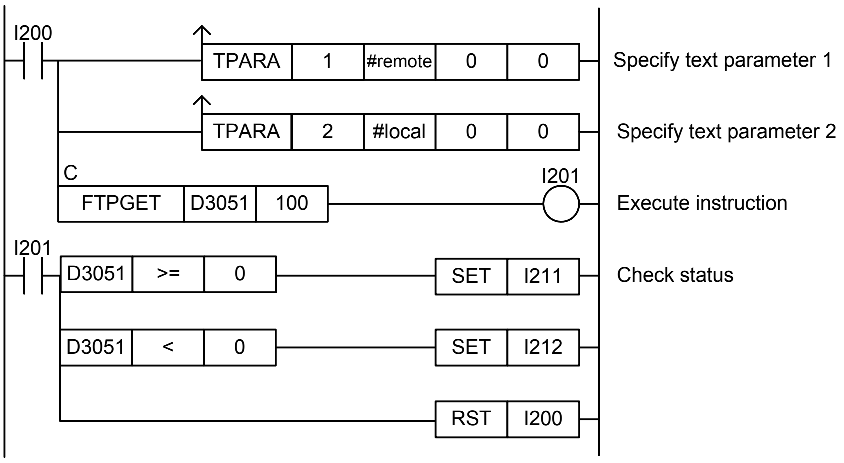


Figure 3.20.8 Example of an FTP Client Get File Program

This sample code gets a file on the FTP server with file pathname defined by constant name #remote and saves it to the FTP client file pathname defined by constant name "#local". The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	1	Status

3.20.7 FTP Client Change Directory (FTPCD)

F3SP66 F3SP71
F3SP67 F3SP76

Changes the remote current directory on the FTP server.

Table 3.20.35 FTP Client Change Directory

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Change Directory	FTPCD	C FTPCD	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

FTP Client Change Directory C FTPCD ret n1

Table 3.20.36 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.20.37 Text Parameters

Parameter		Description
1	n2	New current directory pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Section 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.20.38 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.20.39 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available as Instruction Parameters."

■ Resource Relays

Table 3.20.40 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Changes the remote current directory on the FTP server.

CAUTION

This instruction cannot be executed concurrently with other FTP client instructions.

■ Programming Example

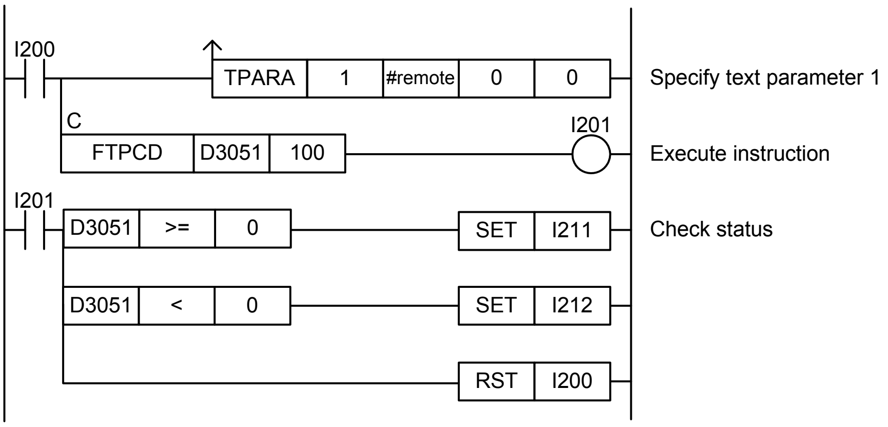


Figure 3.20.9 Example of an FTP Client Change Directory Program

This sample code changes the current directory on the FTP server to the directory defined by constant name #remote. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.20.8 FTP Client Change Local Directory (FTPLCD)

F3SP66 F3SP71
F3SP67 F3SP76

Changes the local current directory on the FTP client.

Table 3.20.41 FTP Client Change Local Directory

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Change Local Directory	FTPLCD	C FTPLCD	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

C
FTP Client Change Local Directory FTPLCD ret n1

Table 3.20.42 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.20.43 Text Parameters

Parameter		Description
1	n2	New local current directory pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Section 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.20.44 Status (Return Value)

Offset (word)	Description	
ret	0	Normal exit
	< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.20.45 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available as Instruction Parameters."

■ Resource Relays

Table 3.20.46 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Changes the local current directory on the FTP client. The local current directory defaults to "\RAMDISK" when FTP client is started.

Changing the local current directory of the FTP client does not affect the current directories of other processing systems (e.g. current directory of the file system instruction group) as the current directories are independent of each other.

■ Programming Example

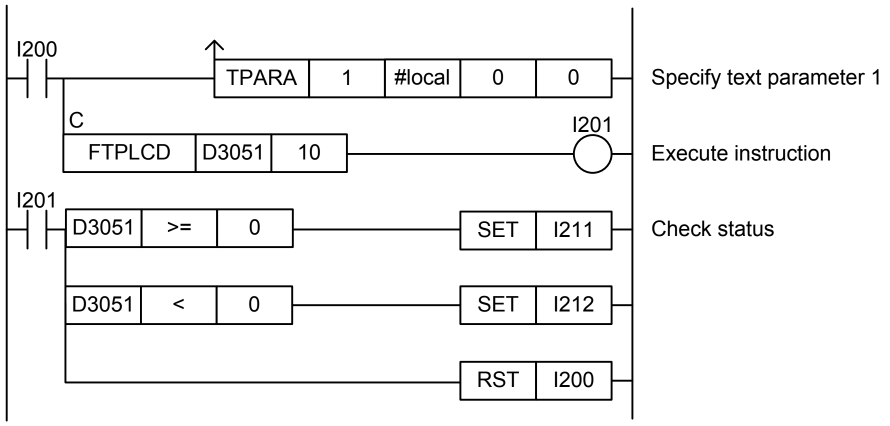


Figure 3.20.10 Example of an FTP Client Change Local Directory Program

This sample code changes the current directory on the FTP client to the directory defined by constant name #local. The timeout interval is set to 10 (1 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.20.9 FTP Client Current Directory Info (FTPPWD)

F3SP66 F3SP71
F3SP67 F3SP76

Gets information about the current directory of the FTP server.

Table 3.20.47 FTP Client Current Directory Info

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Current Directory Info	FTPPWD	C FTPPWD | | |	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

FTP Client Current Directory Info C FTPPWD | ret | t | d

Table 3.20.48 Parameters

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]
	t+1	Max. returned words (W) [1-65]
d		Destination device (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.20.49 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.20.50 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓	✓		✓	✓		Yes	Yes
d									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available as Instruction Parameters."

■ Resource Relays

Table 3.20.51 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Gets information about the current directory of the FTP server. The returned file pathname is a text string of maximum 127 bytes, with a NULL byte appended at the end. If the returned file pathname exceeds the specified maximum number of returned words, the excess bytes are discarded, and a data processing error code (-9015) is stored in status.

■ Programming Example

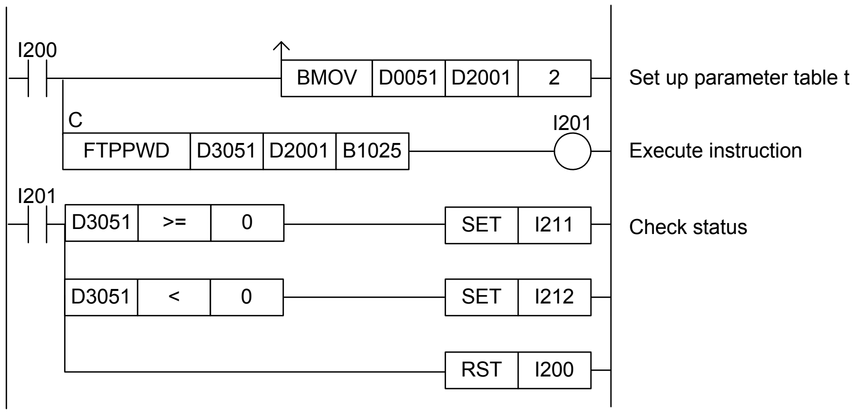


Figure 3.20.11 Example of an FTP Client Current Directory Info Program

This sample code gets information about the current directory of the connected FTP server, and stores the current directory pathname to device B1025.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as follows.

Device	Value	Table Parameter
t = D2001	100	Timeout interval (= 10 s)
D2002	65	Maximum returned words (= 65 words)

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

The table below shows a sample output of current directory information.

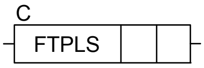
Device	Value	Table Parameter
d = B1025	"C: "	Current directory information (C: \MYDATA)
B1026	"\M"	
B1027	"YD"	
B1028	"AT"	
B1029	"A"+ NULL	

3.20.10 FTP Client Get File List (FTPLS)

F3SP66 F3SP71
F3SP67 F3SP76

Gets detailed information about a specified directory or file on the FTP server.

Table 3.20.52 FTP Client Get File List

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Get File List	FTPLS		✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

FTP Client Get File List

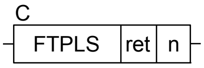


Table 3.20.53 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.20.54 Text Parameters

Parameter		Description
1	s	Target directory pathname [*1]
2	d	Output file pathname
3	n2	"ls" command option [*2]

*1: Specify a NULL value to get information about the current directory.

*2: Prefix the command option parameter value with a hyphen ('-'). For more details about the "ls" command options, see the specifications of the FTP server. If an option is specified, the target directory pathname 's' parameter is ignored and information about the current directory is returned.

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Section 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.20.55 Status (Return Value)

Offset (word)		Description	
Ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.20.56 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available as Instruction Parameters."

■ Resource Relays

Table 3.20.57 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Gets a list of the names of files and directories contained in a FTP server directory designated by the target pathname ('s') parameter. The returned information is output in text format to a file designated by the output file pathname ('d') parameter. Internally, this instruction executes the "NLST" FTP command.

You can specify options for the "NLST" command as a parameter of this FTPLS instruction. For instance, specifying a command option of "-l" returns the file attribute, creation date and other information in addition to the file name. Beware, however, that if you specify an option, the source directory pathname ('s') parameter is ignored, and information of the current directory is always returned.

The table below lists common "NLST" options used. The FTP server function of this module supports only the "-l" option when the module is running as an FTP server.

Table 3.20.58 Examples of "NLST" command options

Option	Description
-l	Returns list output containing file size, creation date and other additional information.
-t	Returns list output sorted in descending order of date.
-tr	Returns list output sorted in ascending order of date.
-F	Appends a '/' identifier behind directory names.
-tF	Returns list output sorted in descending order of date, and appends a '/' identifier behind directory names.

Note: Supported "ls" command options vary with individual FTP server implementations so some of the options described above may be unavailable.

CAUTION

- The operation and implementation of the "NLST" options is according to the specifications of an individual FTP server.
- This instruction cannot be executed concurrently with other FTP client instructions.

■ Programming Example

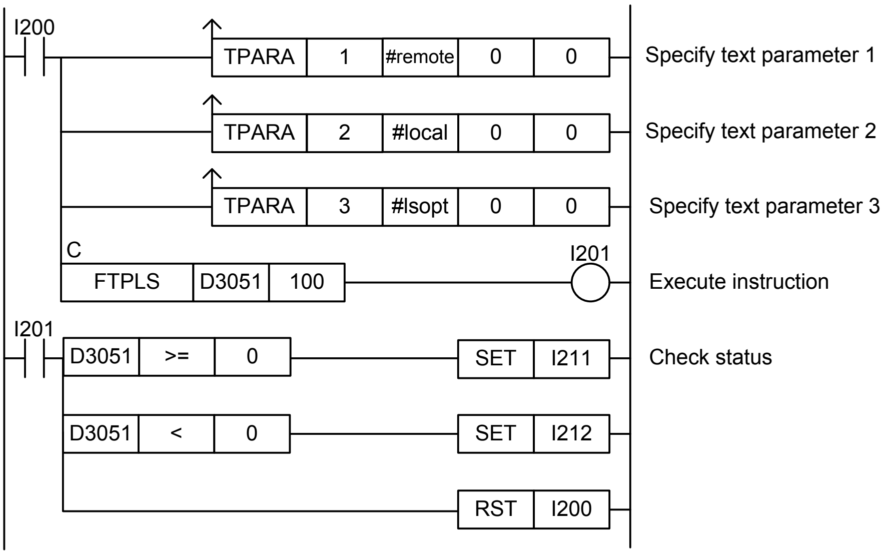


Figure 3.20.12 Example of an FTP Client Get File List Program

This sample code gets file information for the current directory of the FTP server as constant name #remote is assigned the null string. The returned information is output to the file pathname defined by constant name #local. "NLST" option string defined by constant name #lsopt is included as an instruction parameter. The timeout interval is set to 100 (10 s).

```
#remote = ""
#local = "\ramdisk\filestat.txt"
#lsopt = "-l"
```

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.20.11 FTP Client Delete File (FTPDEL)

F3SP66 F3SP71
F3SP67 F3SP76

Deletes one or more specified files on the FTP server.

Table 3.20.59 FTP Client Delete File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Delete File	FTPDEL	C FTPDEL	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

FTP Client Delete File

C
FTPDEL | ret | n

Table 3.20.60 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.20.61 Text Parameters

Parameter		Description
1	d	Target file pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Section 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.20.62 Status (Return Value)

Offset (word)		Description	
ret	ret+0	> 0	Number of deleted files (W) [1-32767]
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

Available Devices

Table 3.20.63 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available as Instruction Parameters."

Resource Relays

Table 3.20.64 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

Function

Deletes one or more specified files on the FTP server.

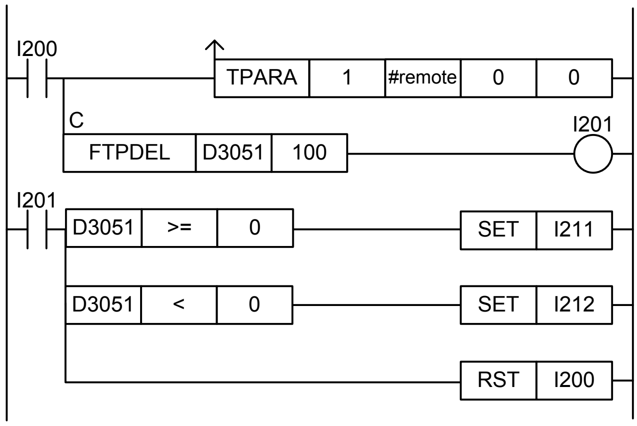
Multiple files can be deleted by including wildcard characters ('*', '?') in the file name. In such situations, even if an error occurs at the FTP server end during file deletion, processing of undeleted files continues.

The number of deleted files is returned and stored in Status.

CAUTION

This instruction cannot be executed concurrently with other FTP client instructions.

Programming Example



Specify text parameter 1

Execute instruction

Check status

Figure 3.20.13 Example of an FTP Client Delete File Program

This sample code deletes the file on the FTP server with pathname defined by constant name #remote. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

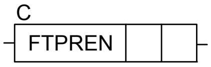
Device	Value	Table Parameter
ret = D3051	1	Status

3.20.12 FTP Client Rename File (FTPREN)

F3SP66 F3SP71
F3SP67 F3SP76

Renames a file on the FTP server.

Table 3.20.65 FTP Client Rename File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Rename File	FTPREN		✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

FTP Client Rename File

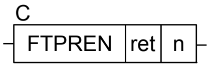


Table 3.20.66 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.20.67 Text Parameters

Parameter		Description
1	s	Old file pathname
2	d	New file pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Section 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.20.68 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.20.69 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available as Instruction Parameters."

■ Resource Relays

Table 3.20.70 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Renames a file on the FTP server.

CAUTION

This instruction cannot be executed concurrently with other FTP client instructions.

■ Programming Example

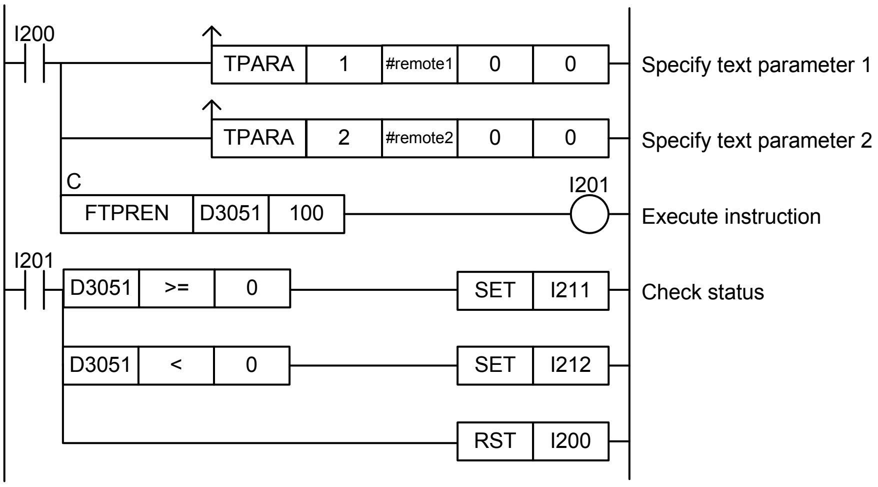


Figure 3.20.14 Example of an FTP Client Rename File Program

This sample code renames an FTP server file designated by constant name #remote1 to the new name defined by constant name #remote2. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.20.13 FTP Client Make Directory (FTPMKDIR)

F3SP66 F3SP71
F3SP67 F3SP76

Creates a directory on the FTP server.

Table 3.20.71 FTP Client Make Directory

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Make Directory	FTPMKDIR	C FTPMKDIR	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

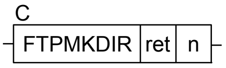
FTP Client Make Directory 

Table 3.20.72 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.20.73 Text Parameters

Parameter		Description
1	d	Pathname of directory to be created

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Section 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.20.74 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.20.75 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available as Instruction Parameters."

■ Resource Relays

Table 3.20.76 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Creates a directory on the FTP server.

CAUTION

This instruction cannot be executed concurrently with other FTP client instructions.

■ Programming Example

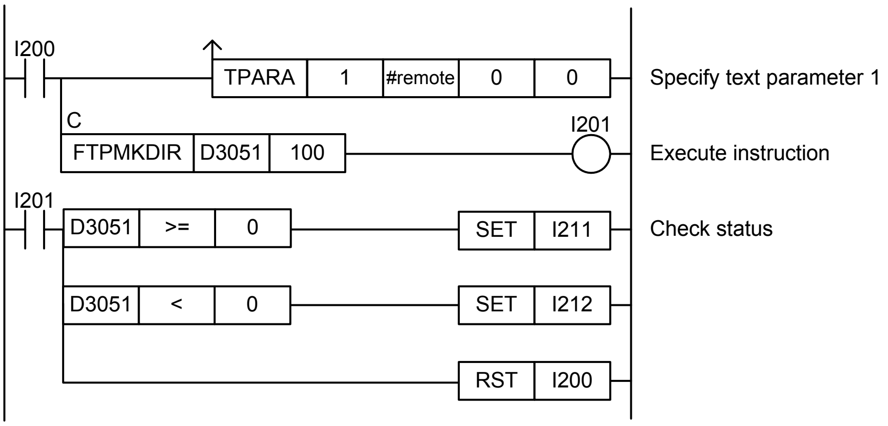


Figure 3.20.15 Example of an FTP Client Make Directory Program

This sample code creates a new directory on the FTP server according to the directory pathname defined by constant name #remote. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.20.14 FTP Client Remove Directory (FTPRMDIR)

F3SP66 F3SP71
F3SP67 F3SP76

Deletes a specified directory on the FTP server.

Table 3.20.77 FTP Client Remove Directory

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Remove Directory	FTPRMDIR	C FTPRMDIR	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

FTP Client Remove Directory C FTPRMDIR | ret | n

Table 3.20.78 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.20.79 Text Parameters

Parameter		Description
1	d	Pathname of directory to be deleted

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Section 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.20.80 Status (Return Value)

Offset (word)	Description	
ret	0	Normal exit
	< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.20.81 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available as Instruction Parameters."

■ Resource Relays

Table 3.20.82 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Deletes a specified directory on the FTP server.

CAUTION

This instruction cannot be executed concurrently with other FTP client instructions.

■ Programming Example

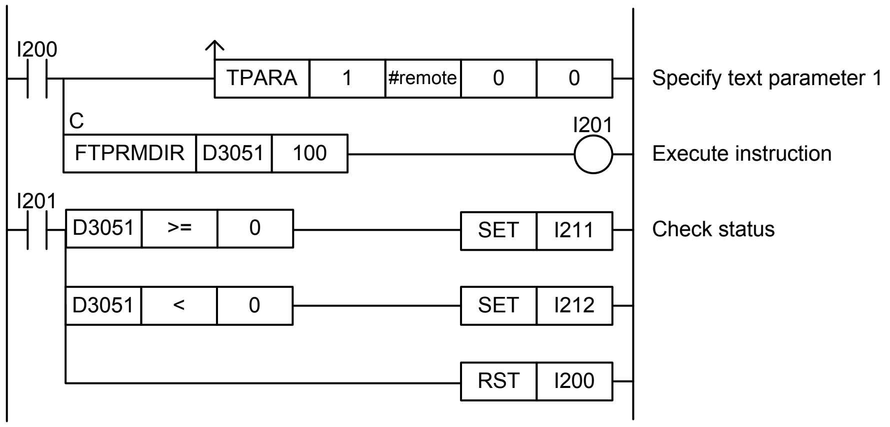


Figure 3.20.16 Example of an FTP Client Remove Directory Program

This sample code deletes from the FTP server the directory designated by the directory pathname defined by constant name #remote. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

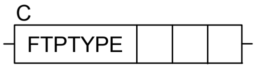
Device	Value	Table Parameter
ret = D3051	0	Status

3.20.15 FTP Client Representation Type (FTPTYPE)

F3SP66 F3SP71
F3SP67 F3SP76

Selects ASCII or binary representation for FTP data transfer.

Table 3.20.83 FTP Client Representation Type

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Representation Type	FTPTYPE		✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

FTP Client Representation Type

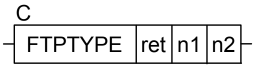


Table 3.20.84 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]
n2	Representation type (W)[0 = ASCII, 1 = binary]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.20.85 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.20.86 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available as Instruction Parameters."

■ Resource Relays

Table 3.20.87 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Selects ASCII or binary representation for FTP data transfer. The representation type defaults to binary when FTP client is started.

In binary representation data transfer, data in files are transferred as is. In general, binary representation can be used for any file format. Both text files (e.g. files with filename extensions ".txt", ".ypjc" and ".yprp") and binary files (e.g. files with filename extensions ".bin", ".pdf", ".doc" and ".jpg") can be sent using binary representation.

ASCII representation is used for sending text files when newline code conversion is required. At transmission, CRLF and CR characters are transmitted without change but LF characters are converted to CRLF. Beware that specifying ASCII representation for transferring a binary file will result in invalid data due to conversion processing.

CAUTION

- This instruction cannot be executed concurrently with other FTP client instructions.
- If the contents of the source file and destination file are unexpectedly different, check whether the problem arose because ASCII representation was specified. If no newline conversion is required, specify binary representation for FTP transfer of all file formats.

■ Programming Example

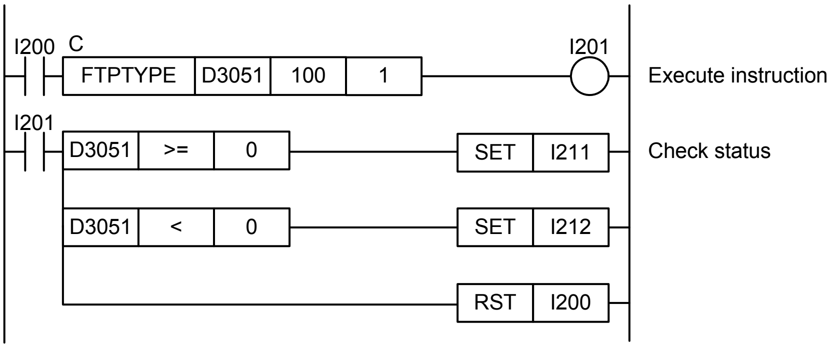


Figure 3.20.17 Example of an FTP Client Representation Type Program

This sample code switches to binary representation for data transfer to the connected FTP server. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.21 FTP Server Instructions

FTP server instructions can be executed to suspend or resume the FTP server request service, which accepts requests from remote FTP clients.

3.21.1 FTP Server Run Request Service (FTPSRUN)

F3SP66 F3SP71
F3SP67 F3SP76

This FTP server instruction resumes the FTP server request service, which accepts requests from FTP clients, if the service had been suspended by a FTP Server Stop Request Service (FTPSSTOP) instruction or by Function Removal of configuration.

Table 3.21.1 FTP Server Run Request Service

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Server Run Request Service	FTPSRUN	C FTPSRUN	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

FTP Server Run Request Service C FTPSRUN | ret | n

Table 3.21.2 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.21.3 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.21.4 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.21.5 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Execution of this instruction is normally not required as FTP server is automatically started at power on or module reset. This FTP server instruction, however, can be used to resume the FTP server request service, which accepts requests from FTP clients, if the service had been suspended by a FTP Server Stop Request Service (FTPSSTOP) instruction or by Function Removal of configuration.

CAUTION

Processing of this instruction is always completed even in the presence of a timeout or cancellation.

■ Programming Example

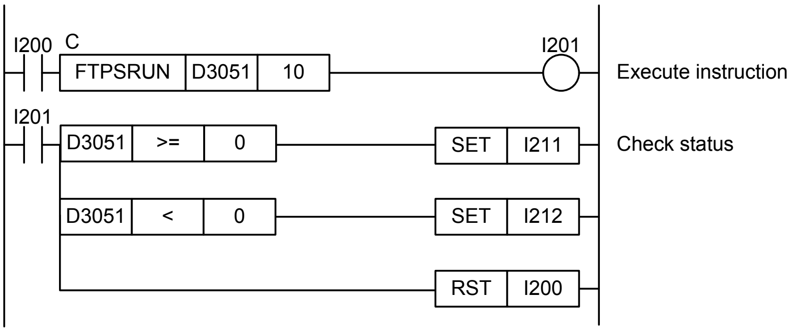


Figure 3.21.1 Example of an FTP Server Run Request Service Program

This sample code resumes the FTP server request service, which had been suspended by a FTP Server Stop Request Service (FTPSSTOP) instruction. The timeout interval is set to 10 (1 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.21.2 FTP Server Stop Request Service (FTPSSTOP)

F3SP66 F3SP71 F3SP67 F3SP76

This FTP server instruction suspends the FTP server request service, which accepts requests from FTP clients.

Table 3.21.6 FTP Server Stop Request Service

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Server Stop Request Service	FTPSSTOP	C ┤FTPSSTOP│ │ │├	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

```
                                      C
FTP Server Stop Request Service  ┤FTPSSTOP│ret│n├
```

Table 3.21.7 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.21.8 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.16.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.21.9 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

Table 3.21.10 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

This FTP server instruction suspends the FTP server request service, which accepts requests from FTP clients. If a request from an FTP client is being processed when this instruction is executed, processing of the request will still be carried through to the end.

CAUTION

Processing of this instruction is always completed even in the event of a timeout or cancellation.

■ Programming Example

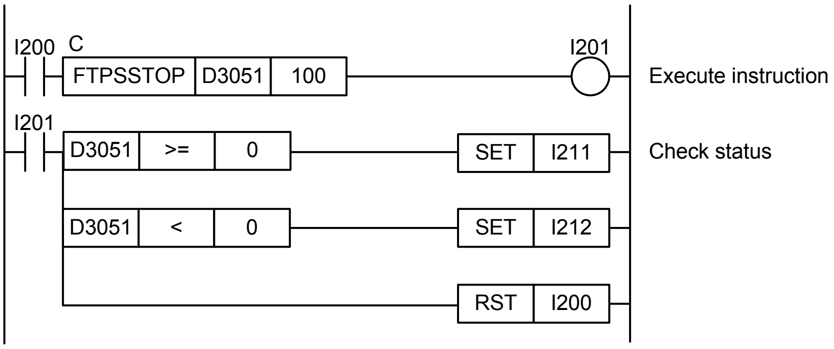


Figure 3.21.2 Example of an FTP Server Stop Request Service Program

This sample program suspends the FTP server request service. The timeout interval is set to 100 (10 s).

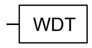
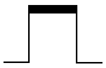
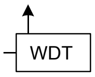
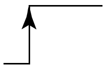
The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.22 Miscellaneous Instructions

3.22.1 Refresh Watchdog Timer (WDT)

Table 3.22.1 Refresh Watchdog Timer

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	85	Refresh Watchdog Timer	WDT		✓	—		1	—	—
	85P		↑WDT					2		

■ Parameter

Refresh Watchdog Timer

F031501.VSD

■ Function

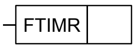
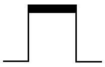
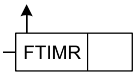
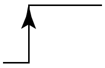
The Refresh Watchdog Timer instruction refreshes the watchdog timer. This instruction is used to perform processing whose scan time is temporarily extended when executing a sequence while monitoring the scan time (e.g., uploading the day's data at 12 o'clock on a daily basis).

This instruction clears the watchdog timer that is monitoring the sequence scan time. If the scan time is found extremely longer than normal, execute the WDT instruction several times during the pertinent process.

3.22.2 Read Free Run Timer (FTIMR)

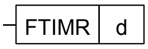
F3SP22
F3SP28
F3SP38 | F3SP53
F3SP58
F3SP59 | F3SP66
F3SP67 | F3SP71
F3SP76

Table 3.22.2 Read Free Run Timer

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	186	Read Free Run Timer	FTIMR	 FTIMR	✓	—		2	16 bits	—
	186P		↑FTIMR	 FTIMR				3		

■ Parameter

Read Free Run Timer



F031502.VSD

d : Device for storing the value of the free-run timer (0 to 65,535 in 10 μs increments). For example, the timer value 1 equals 10μs.

■ Available Devices

Table 3.22.3 Devices Available for Read Free Run Timer Instruction

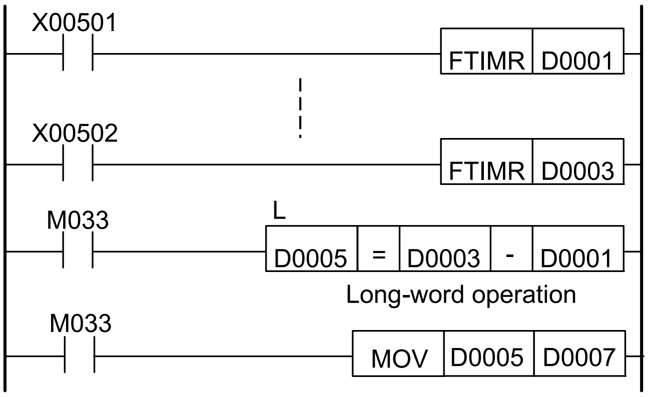
Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d		✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓	✓*1	✓*1	✓*1	✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value
*3: Counter current value

■ Function

The FTIMR instruction reads and stores the free-run timer of the sequence CPU module to a specified device. The timer counts from 0 to 65,535 and then returns to 0. The unit of the timer value is 10 µs.

Be careful when you use the free-run timer as FA-M3 Range-free Multi-controller does not support unsigned arithmetic operations. To perform an unsigned arithmetic operation, execute the operation by treating the timer value as long-word data, and then read the result as unsigned words, as shown below:

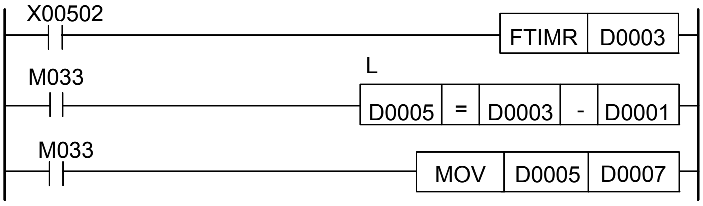


F031503.VSD

Figure 3.22.1 Example of Handling Timer Value in Unsigned Arithmetic Operation

■ Programming Example

The program shown below reads and stores the free-run timer to D0003, subtracts D0001 from it, and stores the result in D0007 if X00502 is on. D0001 holds the old timer value.



Line No	Instruction	Operands				
0001	LD	X00502				
0002	FTIMR	D0003				
0003	LD	M033				
0004	CAL L	D0005	=	D0003	-	D0001
0005	MOV	D0005	D0007			

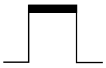
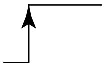
F031504.VSD

Figure 3.22.2 Example of a Program with Read Free Run Timer Instruction

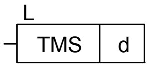
3.22.3 Start Elapsed Time Measurement (TMS)

F3SP71
F3SP76

Table 3.22.4 Start Elapsed Time Measurement

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	965	Start Elapsed Time Measurement	TMS L	L TMS	✓	—		2	32 bit	—
	965P		↑TMS L	↑L TMS				3		

■ Parameter

Start Elapsed Time Measurement 

F3223001.VSD

d : Device number of the first device for storing the free-run timer value.

■ Available Devices

Table 3.22.5 Devices Available for the TMS Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
d									✓	✓*1	✓*1	✓*1		✓*1			Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The TMS instruction stores the free-run timer value, which is the reference value for time measurement, as a long-word in a specified device.

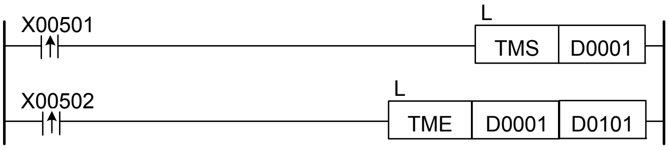
The TMS instruction can be used to measure the elapsed time from the start of measurement by using it with the TME instruction.

The resolution of time measurement is 1μs. Up to 2,147,483,647μs (approx. 35 minutes) can be measured. Unlike the timer (TIM) instruction, measurement is performed when the instruction is executed, thus time can be measured even in subroutines or macro instructions. It is also possible to measure multiple elapsed times by providing multiple areas for storing free-run timer values. If time measurement exceeds 2,147,483,647μs (approx. 35 minutes), the result of the elapsed time measurement is indefinite.

■ Programming Example

The sample code shown below is a program that measures time from X00501 ON to X00502 ON.

When X00501 turns on, a TMS instruction is executed and the free-run timer value is stored at D0001 to D0002 as a long-word. When X00502 turns on, a TME instruction is executed and the measured elapsed time is stored at D0101 to D0102 as a long-word.



Line No.	Instruction	Operands				
0001	LDU	X00501				
0002	TMS L	D0001				
0003	LDU	X00502				
0004	TME L	D0001	D0101			

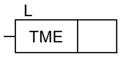
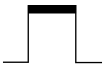
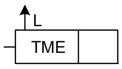
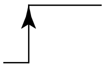
F3223001.VSD

Figure 3.22.3 Example of a TMS Program

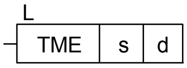
3.22.4 Elapsed Time Measurement (TME)

F3SP71
F3SP76

Table 3.22.6 Elapsed Time Measurement

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	966	Elapsed Time Measurement	TME L		✓	—		3	32 bit	—
	966P		↑TME L					4		

■ Parameter

Elapsed Time Measurement 

F3224001.VSD

s : Device number of the first device storing the free-run timer value obtained by the TMS instruction
d : Device number of the first device for storing the measured elapsed time.

■ Available Devices

Table 3.22.7 Devices Available for the TME Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s									✓	✓*1	✓*1	✓		✓			Yes	Yes
d									✓	✓*1	✓*1	✓*1		✓*1			Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

The TME instruction stores the elapsed time, which is measured from the execution of the TMS instruction, in a specified device as a long-word in 1μs increments.

The TMS instruction can be used to measure the elapsed time from the start of measurement by using it with the TME instruction.

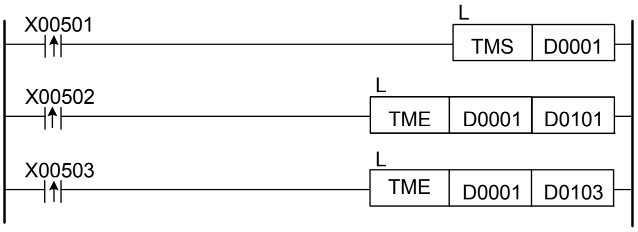
The resolution of time measurement is 1μs. Up to 2,147,483,647μs (approx. 35 minutes) can be measured. Unlike the timer (TIM) instruction, measurement is performed when the instruction is executed, thus time can be measured even in subroutines or macro instructions. Time measurement can be performed any number of times within 2,147,483,647μs (approx. 35 minutes) based on the free-run timer value stored by the TMS instruction. If time measurement exceeds 2,147,483,647μs (approx. 35 minutes), the result of the elapsed time measurement is indefinite.

■ Programming Example

The sample code shown below is a program that measures time from X00501 ON to X00502 ON and time from X00501 ON to X00503 ON.

When X00501 turns on, a TMS instruction is executed and the free-run timer value is stored at D0001 to D0002 as a long-word. When X00502 turns on, a TME instruction is executed and the measured elapsed time is stored at D0101 to D0102 as a long-word.

When X00503 turns on, a TME instruction is executed and the measured elapsed time is stored at D0103 to D0104 as a long-word.



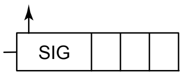
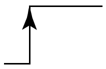
Line No.	Instruction	Operands				
0001	LDU	X00501				
0002	TMS L	D0001				
0003	LDU	X00502				
0004	TME L	D0001	D0101			
0005	LDU	X00503				
0006	TME L	D0001	D0103			

F3224001.VSD

Figure 3.22.4 Example of a TME Program

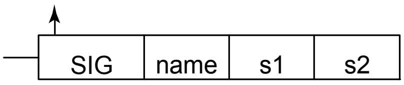
3.22.5 Interrupt to BASIC (SIG)

Table 3.22.8 Interrupt to BASIC

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	280P	Interrupt to BASIC	↑SIG		✓	—		5	—	—

■ Parameter

Interrupt to BASIC



F031505.EPS

name : Signal name
s1 : Device number of the first device storing the additional data
s2 : Destination

■ Available Devices

Table 3.22.9 Devices Available for the Interrupt to BASIC Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
name																	No	No
s1	✓	✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓	✓	Yes	Yes
s2	✓	✓	✓	✓*1	✓*1	✓*1	✓*2	✓*3	✓	✓*1	✓*1	✓*1	✓*1	✓*1	✓	✓	Yes	Yes

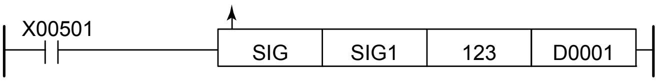
*1: See Section 1.17, "Devices Available As Instruction Parameters."
*2: Timer current value
*3: Counter current value

■ Function

The Interrupt to BASIC instruction sends a signal to a BASIC program, and so on. The additional data s1 is 1 word (2 bytes) long. You cannot send long word data with this instruction. The destination s2 is 1 word (2 bytes) long and its higher-order byte is always set to $00. The lower-order byte must be set to the BASIC CPU number (CPU's slot number). Parameter "name" specifies the name of the signal to be sent (8 bytes or less ASCII codes).

■ Programming Example

The sample code shown below sends a signal named “SIG1” to the BASIC CPU designated by the lower-order byte of D0001 with additional data “123” if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	↑SIG	SIG1	123	D0001		

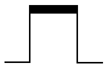
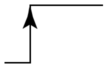
F031506.VSD

Figure 3.22.5 Example of an Interrupt to BASIC Program

3.22.6 Sampling Trace (TRC)

F3SP25 F3SP35	F3SP22 F3SP28 F3SP38	F3SP53 F3SP58 F3SP59	F3SP66 F3SP67	F3SP71 F3SP76

Table 3.22.10 Sampling Trace

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Processing Unit	Carry
					Yes	No				
Application Instruction	181	Sampling Trace	TRC	TRC	✓	—		1	—	—
	181P		↑TRC	↑ TRC				2		

■ Parameter

Sampling Trace

F031511.VSD

■ Function

The Sampling Trace instruction performs a sampling trace (collecting trace data). When this instruction is placed in a program where sampling trace is to start and the start of sampling trace is specified, the system starts storing the run-time value of the specified devices in the sampling trace buffer.

When two or more TRC instructions are encountered during a single scan, up to four occurrences of the instruction are executed in the order in which they appear; the fifth and subsequent occurrences of the TRC instruction are ignored.

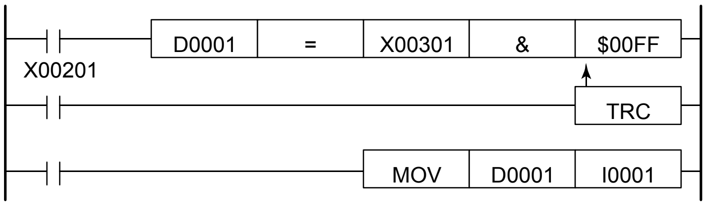
However, there is no limit on the number of TRC instruction occurrences during a single scan for F3P71 and F3SP76.

SEE ALSO

For details on sampling trace, see Section 6.10 of "Sequence CPU Instruction Manual - Functions (for F3SP22-0N, F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S)" (IM 34M06P13-01E), Section A6.10 of "Sequence CPU - Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E), or Section A6.12 of "Sequence CPU Instruction Manual - Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

■ Programming Example

The sample code shown below starts a sampling trace (collecting trace data) when X00201 is set to ON.

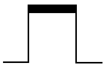
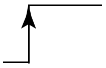
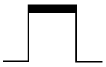
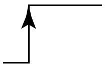
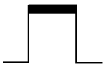
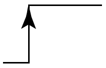


F031512.VSD

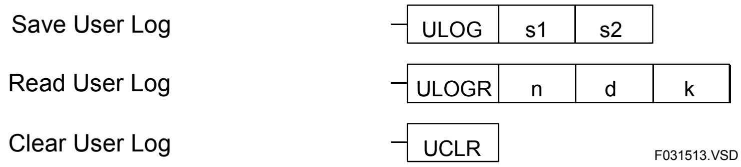
Figure 3.22.6 Example of a Sampling Trace Program

3.22.7 Save User Log (ULOG), Read User Log (ULOGR), Clear User Log (UCLR)

Table 3.22.11 Save User Log, Read User Log, Clear User Log

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	961	Save User Log	ULOG	ULOG	✓	—		4	16 bits	—
	961P		↑ULOG	↑ ULOG				5		
	962	Read User Log	ULOGR	ULOGR	✓	—		5	16 bits	—
	962P		↑ULOGR	↑ ULOGR				6		
	963	Clear User Log	UCLR	UCLR	✓	—		2	16 bits	—
	963P		↑UCLR	↑ UCLR				3		

■ Parameter



s1	:	Main code (Range: -32768 to +32767). Can store messages No. 1 to 64.
s2	:	Subcode (Range: -32768 to +32767)
n	:	Read position (Range: 0 to 63, 0 for the last data)
d	:	Device number of the first device for storing the read data
k	:	Size of the read message area (0 to 32 bytes; 0 for no message)

■ Available Devices

(1) Save User Log

Table 3.22.12 Devices Available for the Save User Log Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
s1	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
s2	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes

*1: Timer current value
*2: Counter current value
*3: See Section 1.17, "Devices Available As Instruction Parameters."

(2) Read User Log

Table 3.22.13 Devices Available for the Read User Log Instruction

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
n	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes
d		✓	✓	✓*3	✓*3	✓*3			✓	✓*3	✓*3	✓*3	✓*3	✓*3	✓		Yes	Yes
k	✓	✓	✓	✓	✓	✓	✓*1	✓*2	✓	✓*3	✓*3	✓	✓	✓	✓	✓	Yes	Yes

*1: Timer current value
*2: Counter current value
*3: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

(1) Save User Log

The Save User Log instruction stores the parameters specified in s1 and s2 in the user log area as user log information together with the date and time. The saved user log information can be read from WideField3, WideField2, WideField, Ladder Diagram Support Program M3, or using the User Log Read instruction. A CPU can store messages that correspond to main codes (s1) of 1 to 64.

(2) Read User Log

The Read User Log instruction reads the user log information that is identified by read position n into the device position d. A read position of 0 corresponds to the latest data. The greater the read position, the older the user log information is.

The date and time read, main code, and subcode are represented in ASCII. If a message corresponding to the main code is stored, as many as k bytes of message text are also read. If the size of the read message text is smaller than k (bytes), the extra read message area is padded with Nulls ($0000). Null ($0000) data is returned if no user log information is stored at read position n.

SEE ALSO

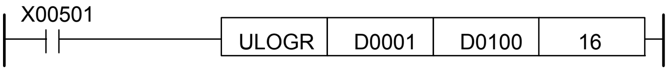
For details on user logs, see Section 6.14 of "Sequence CPU Instruction Manual - Functions (for F3SP22-0N, F3SP28-3N/3S, F3SP38-6N/6S, F3SP53-4H/4S, F3SP58-6H/6S, F3SP59-7S)" (IM 34M06P13-01E), Section A6.14 of "Sequence CPU - Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E), or Section A6.13 of "Sequence CPU Instruction Manual - Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

(3) Clear User Log

The Clear User Log instruction clears the user log information.

■ Programming Example

The sample code shown below reads 16 bytes of user log information from D001 into D0100 if X00501 is on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	ULOGR	D0001	D0100	16		

User Log Information

$$\underbrace{\text{05 / 10 / 19}}_{\text{Date}}\quad\underbrace{\text{11 : 20 : 34}}_{\text{Time}}\quad\underbrace{\text{+1}}_{\text{Main code}}\quad\underbrace{\text{+23}}_{\text{Subcode}}\quad\underbrace{\text{Emergency Stop}}_{\text{Message}}$$

D0100	$3035	'0' '5'
D0101	$3130	'1' '0'
D0102	$3139	'1' '9'
D0103	$3131	'1' '1'
D0104	$3230	'2' '0'
D0105	$3334	'3' '4'
D0106	$2020	' ' ' '
D0107	$2020	' ' ' '
D0108	$2B31	'+' '1'
D0109	$2020	' ' ' '
D0110	$202B	' ' '+'
D0111	$3233	'2' '3'

D0112	$4560	'Em'
D0113	$6572	'er'
D0114	$6765	'ge'
D0115	$6E63	'nc'
D0116	$7900	'y'
D0117	$5374	'St'
D0118	$6F70	'op'
D0119	$0000	

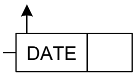
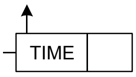
F031514.VSD

Figure 3.22.7 Example of a User Log Read Program

3.22.8 Set Date (DATE), Set Time (TIME)

F3SP22-0S
F3SP28-3S
F3SP38-6S | F3SP53-4S
F3SP58-6S
F3SP59-7S | F3SP66
F3SP67 | F3SP71
F3SP76

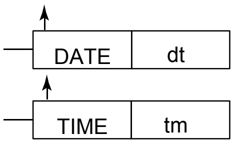
Table 3.22.14 Set Date and Set Time

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	520P	Set Date	↑DATE	 DATE	✓	—	↑	3	3 words	—
	521P	Set Time	↑TIME	 TIME	✓	—	↑	3	3 words	—

■ Parameter

Set Date

Set Time



F031515.VSD

dt : First device for date data
tm : First device for time data

■ Available Devices

Table 3.22.15 Devices Available for Set Date

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
dt			✓	✓*1	✓*1				✓	✓	✓*1	✓*1		✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

Table 3.22.16 Devices Available for Set Time

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
tm			✓	✓*1	✓*1				✓	✓	✓*1	✓*1		✓*1			Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

● Set Date (DATE)

Sets the date (year/month/day) of the sequence CPU module.

To set the date, store the desired date value in binary format in devices, and specify the first address of these devices in a Set Date instruction, as shown below.

dt	yy
dt+1	mm
dt+2	dd

yy : Last two digits of the year (1980-2079)

mm : Month (1 to 12)

dd : Day (1 to 31)

Example: Setting the date to January 3, 2010

dt	10
dt+1	1
dt+2	3

TIP

You can also use a character string to set the system date. To do so, use the Set Date String (SDATE) instruction.

TIP

You can also directly operate on special registers and special relays to change the date. For details, see "Sequence CPU – Functions User's Manual."

CAUTION

- Always enter date and time in binary data format for the Set Date instruction and Set Time instruction. Do not use BCD data format for these instructions.
- If an impossible date or time (e.g., February 31) is specified, the current date or time setting remains unchanged. No error is generated.
- If the specified date value or time value is out of range (e.g., 100 for year, 13 for month, or -1 for minute), an instruction error is generated (error code $2101: invalid parameter range).

CAUTION

Do not use the Set Date instruction in a sensor control block or I/O module interrupt program.

● Set Time (TIME)

Sets the time (hour/minute/second) of a sequence CPU module.

To set the time, store the desired time value in devices using the format shown below, and specify the first address of these devices in a Set Time instruction. Store all time data in binary format.

tm	hh
tm+1	mm
tm+2	ss

hh : Hour in 24-hour format (0 to 23)

mm : Minute (0 to 59)

ss : Second (0 to 59)

Example: Setting the time to 19:55:00

tm	19
tm+1	55
tm+2	0

TIP

You can also use a character string to set the date. To do so, use the Set Time String (STIME) instruction.

TIP

You can also directly operate on special registers and special relays to change the time. For details, see "Sequence CPU – Functions User's Manual."

CAUTION

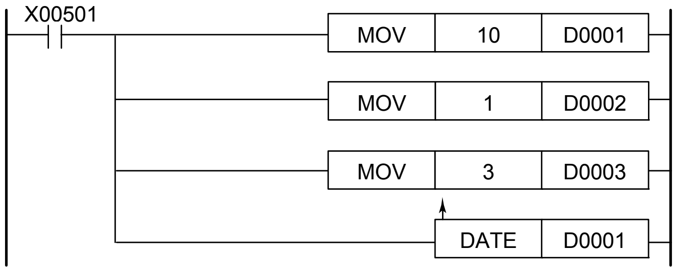
- Always enter time values in binary data format for the Set Time instruction. Do not use BCD data format for the instruction.
- If the specified time value is out of range (e.g., -5 for hour, 70 for minute, or 100 for second), an instruction error is generated (error code $2101: invalid parameter range).

CAUTION

Do not use the Set Time instruction in a sensor control block or I/O module interrupt program.

■ Programming Example

The sample code shown below sets the date to January 3, 2010 if X00501 turns on.

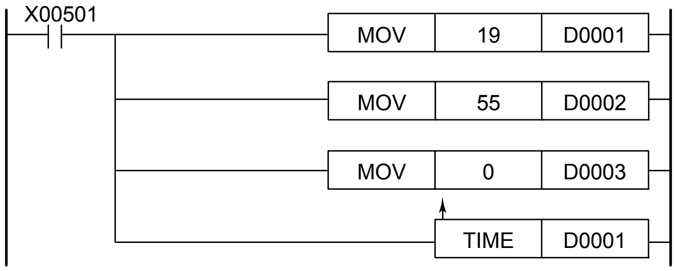


Line No.	Instruction	Operands				
0001	LD	X00501				
0002	PUSH					
0003	MOV	10	D0001			
0004	STCRD					
0005	MOV	1	D0002			
0006	STCRD					
0007	MOV	3	D0003			
0008	POP					
0009	DATE P	D0001				

F031516.VSD

Figure 3.22.8 Example of a Set Date Program

The sample code shown below sets the time to 19:55:00 if X00501 turns on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	PUSH					
0003	MOV	19	D0001			
0004	STCRD					
0005	MOV	55	D0002			
0006	STCRD					
0007	MOV	0	D0003			
0008	POP					
0009	TIME P	D0001				

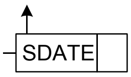
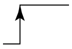
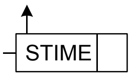
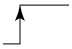
F031517.VSD

Figure 3.22.9 Example of a Set Time Program

3.22.9 Set Date String (SDATE), Set Time String (STIME)

F3SP22-0S F3SP28-3S F3SP38-6S | F3SP53-4S F3SP58-6S F3SP59-7S | F3SP66 F3SP67 | F3SP71 F3SP76

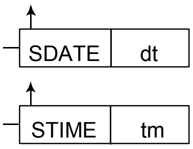
Table 3.22.17 Set Date String and Set Time String

Classi-fication	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Execution Condition	Step Count	Pro-cessing Unit	Carry
					Yes	No				
Appli-cation Instruc-tion	522P	Set Date String	↑SDATE		✓	—		3	8 bits	—
	523P	Set Time String	↑STIME		✓	—		3	8 bits	—

■ Parameter

Set Date String

Set Time String



F031518.VSD

dt : First device for device for date string data
tm : First device for time string data

■ Available Devices

Table 3.22.18 Devices Available for Set Date String

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
dt			✓	✓*1	✓*1				✓	✓	✓*1	✓*1		✓*1	✓		Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

Table 3.22.19 Devices Available for Set Time String

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Constant	Index Modification	Indirect Specification, Pointer P
tm			✓	✓*1	✓*1				✓	✓	✓*1	✓*1		✓*1			Yes	Yes

*1: See Section 1.17, "Devices Available As Instruction Parameters."

■ Function

● Set Date String (SDATE)

Sets the date (year/month/day) of a sequence CPU module.

To set the date, store the date string data in one of the following formats in devices, and specify the beginning address of these devices in a Set Date String instruction. Always append a null byte ($00) at the end of date strings, whichever the format.

format1 : Using no delimiter
“yymmdd”＋ $00
format2 : Using the forward slash “/” ($2F) as delimiter
“yy/mm/dd”＋ $00
format3 : Using the blank space “” ($20) as delimiter
“yy mm dd”＋ $00

yy : Last two digits of the year (19$\boxed{80}$-20$\boxed{79}$)
mm : Month (01 to 12)
dd : Day (01 to 31)

Example: Setting the date to January 3, 2015
format1 : “150103”＋ $00
format2 : “15/01/03”＋ $00
format3 : “15 01 03”＋ $00

TIP

You can also use binary data to set the date. To do so, use the Set Date (DATE) instruction.

TIP

You can also directly operate on special registers and special relays to change the date. For details, see "Sequence CPU – Functions User's Manual."

CAUTION

- If an impossible date or time (e.g., February 31) is specified, the current date remains unchanged. No error is generated.
- If the specified date value or time value is out of range (e.g., 100 for year, 13 for month, or -1 for minute), an instruction error is generated (error code $2101: invalid parameter range).
- If the format of the specified date string is invalid so that the string cannot be recognized as a date string, an instruction error is generated (error code $2101: invalid parameter range).

CAUTION

Do not use the Set Date String instruction in a sensor control block or I/O module interrupt program.

● Set Time String (STIME)

Sets the time (hour/minute/second) of the sequence CPU module.

To set the time, store the time string data in any of the following formats in devices, and specify the first address of these devices in a Set Time String instruction. Always append a null byte ($00) at the end of time strings, whichever the format

format1 : Using no delimiter
"hhmmss"＋ $00
format2 : Using the colon ":" ($2F) as delimiter
"hh:mm:ss"＋ $00
format3 : Using the blank space " " ($20) as delimiter
"hh mm ss"＋ $00

hh : Hour in 24-hour format (00-23)
mm : Minute (00-59)
ss : Second (00-59)

Example: Setting the time to 19:55:00
format1 : "195500"＋ $00
format2 : "19:55:00"＋ $00
format3 : "19 55 00"＋ $00

TIP

You can also use binary data to set the time. To do so, use the Set Time (TIME) instruction.

TIP

You can also directly operate on special registers and special relays to change the time.

For details, see "Sequence CPU – Functions User's Manual."

CAUTION

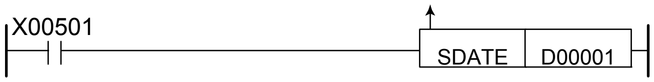
- If the specified time value is out of range (e.g., -5 for hour, 70 for minute, or 100 for second), an instruction error is generated (error code $2101: invalid parameter range).
- If a time string for time is not in correct format so that it cannot be recognized as a time string, an instruction error will be generated. (error code $2101: invalid parameter range).

CAUTION

Do not use the Set Time String instruction in a sensor control block or I/O module interrupt program.

■ Programming Example

The sample code shown below reads the date string stored in the devices starting with D0001 and sets the date if X00501 turns on.

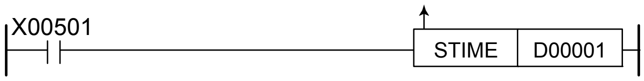


Line No.	Instruction	Operands				
0001	LD	X00501				
0002	SDATE	D0001				

F031519.VSD

Figure 3.22.10 Example Set Date String Program

The sample code shown below reads the time string stored in devices starting with D0001 and sets the time if X00501 turns on.



Line No.	Instruction	Operands				
0001	LD	X00501				
0002	STIME	D0001				

F031520.VSD

Figure 3.22.11 Example of a Set Time String Program

3.22.10 Write CPU Properties (PWRITE)

F3SP66 F3SP71
F3SP67 F3SP76

Writes CPU property values stored in device starting from a specified device to the module.

Table 3.22.20 Write CPU Properties

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Write CPU Properties	PWRITE	C PWRITE	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

Write CPU Properties C PWRITE | ret | n1 | s

Table 3.22.21 Parameters

Parameter	Description
ret[*1]	Device for storing status (W)
n1	Setup no. (W) [1= (Reserved) [*2] 2= (Reserved) [*2] 3=Ethernet setup 4=Socket setup 5=Socket address setup 6=Higher-level link service setup 7=FTP client setup 8=FTP client address setup 9=FTP server setup 10=Rotary switch setup 11=Network filter setup 100=RENEW part 900=Write to internal ROM]
s	First device of CPU property data (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)

*2: Do not specify setup no. indicated as "(reserved)" in the above table.

Table 3.22.22 Text Parameter

Parameter		Description
1	n2	Security keyword

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Section 1.19, "Text Parameter."

■ Status (Return Value)

Table 3.22.23 Status (Return Value)

Offset (word)		Description	
ret	ret+0	= 0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

■ Available Devices

Table 3.22.24 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
s									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

■ Resource Relays

None

■ Function

This is an instruction for setting CPU property values from a ladder program.

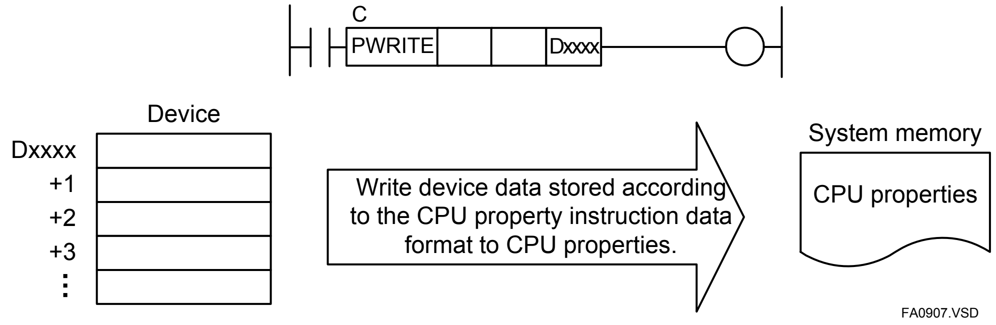

C
PWRITE
Dxxxx
Device
Dxxxx
+1
+2
+3
⋮
Write device data stored according to the CPU property instruction data format to CPU properties.
System memory
CPU properties
FA0907.VSD

Figure 3.22.12 Write CPU Properties

This instruction writes CPU properties on per CPU property setup basis. Before executing the instruction, store the CPU property data in devices according to the required data format. During execution, the data is copied to the system memory. The modified CPU property values in the system memory are then applied to various functions according to the corresponding application timing for each CPU property setup.

SEE ALSO

- For details on the required format for the CPU property data, see Table, "CPU Property Instruction Data Format" for the respective CPU property setup in Section A9.5, "CPU Property Items" of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E) and Section A9.5, "CPU Property Items" of "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E). For details on the RENEW part, see "■ RENEW Part (RENEW PROPERTY SELECTOR PART)" of Section A9.4, "CPU Property File Specifications."
- For details on when CPU Properties are applied, see "■ When are CPU Properties Applied" of Section A9.5, "CPU Property Items" of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E) and "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

Before executing the instruction, you must specify values for the setup items listed in the table below in the CPU property data device area designated by the First Device of CPU Property Data parameter according to the CPU Property Instruction Data Format.

Table 3.22.25 Required Setup in CPU Property Data Device Area

Setup Name [Setup No.]	Required Setup Items	Description
Socket address setup [5]	Socket address setting no.	Specify the target socket address setting no. (n) of CPU properties for writing.
	Socket address type	Select IP address or hostname as the data type for the socket address setting.
FTP client address setup [8]	Destination FTP server setting no.	Specify the target destination FTP server setting no. (n) of CPU properties for writing.
	Destination FTP server address type	Select IP address or hostname as the data type for the destination FTP server address setting.
Network filter setup [11]	Allowed host setting no.	Specify the target allowed host setting no. (n) of CPU properties for writing.
	Allowed host address type	Select IP address or hostname as the data type for the allowed host setting. This item is not used when reading CPU properties.

To ensure that CPU property data copied to the system memory remains valid after power off, you need to write the data to the internal ROM. To do so, execute this instruction, specifying 900 for the setup number parameter. This writes all CPU property data in the system memory to the internal ROM in one go so for faster processing, you can do this after copying CPU property data for multiple setup numbers to the system memory.

If CPU property data is protected with a security keyword, you must specify a valid security keyword as a text parameter.

Multiple concurrent executions of this instruction are not allowed. An error is also generated if this instruction is executed while CPU property data is being written by WideField3 or a smart access function.

CAUTION

- No timeout interval can be specified for this instruction, which always waits indefinitely. Furthermore, instruction execution always completes regardless of any cancellation request.
- Always write CPU property data to the internal ROM eventually. Otherwise, all modifications will be lost after power off and CPU property values will revert to their old values after power on.
- Specifying an out-of-range property value generates a data processing error (error code -9015). When this happens, the invalid property value is not written but all other valid property values are written to system memory. Beware of possible loss of data integrity among property values in this case.

■ Programming Example

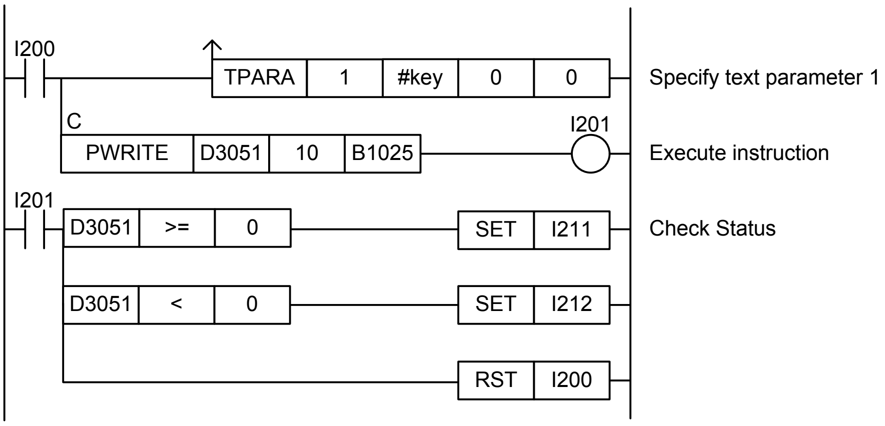


Figure 3.22.13 Write CPU Properties Sample Program

This sample code modifies the rotary switch setup of CPU properties data, which is protected. It assumes that the security keyword is defined by constant name #key, and the new rotary switch setup is stored to device, starting from device B1025.

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter Name
ret=D3051	0	Status

3.22.11 Read CPU Properties (PREAD)

F3SP66 F3SP71
F3SP67 F3SP76

Reads and stores a specified setup of CPU properties to device, starting with a specified first device.

Table 3.22.26 Read CPU Properties

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Read CPU Properties	PREAD	C ┤ PREAD │ │ │ ├	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see Subsection 1.18.1, "Operation of Continuous Type Application Instructions."

■ Parameter

```
                            C
Read CPU Properties  ┤ PREAD │ret│n1│ d ├
```

Table 3.22.27 Parameters

Parameter	Description
ret*1	Device for storing status (W)
n1	Setup no. (W) [1= (Reserved) *2 2= (Reserved) *2 3=Ethernet setup 4=Socket setup 5=Socket address setup 6=Higher-level link service setup 7=FTP client setup 8=FTP client address setup 9=FTP server setup 10=Rotary switch setup 11=Network filter setup]
d	First output device for CPU Properties (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)

*2: Do not specify setup no. indicated as "(reserved)" in the above table.

Table 3.22.28 Text Parameter

Parameter		Description
1	n2	Security keyword string Specify a valid security keyword if CPU properties is protected.

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see Section 1.19, "Text Parameter."

Status (Return Value)

Table 3.22.29 Status (Return Value)

Offset (word)		Description	
ret	ret+0	= 0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see Subsection 1.18.4, "Error Status of Continuous Type Application Instructions."

Available Devices

Table 3.22.30 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	F	W	Z	R	V	Con-stant	Index Modification	Indirect Specification, Pointer P
ret									✓	✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓	✓		✓	✓	✓	Yes	Yes
d									✓	✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.17, "Devices Available As Instruction Parameters."

Resource Relays

None

Function

Reads and stores a specified setup of CPU properties to device, starting from a specified first device.

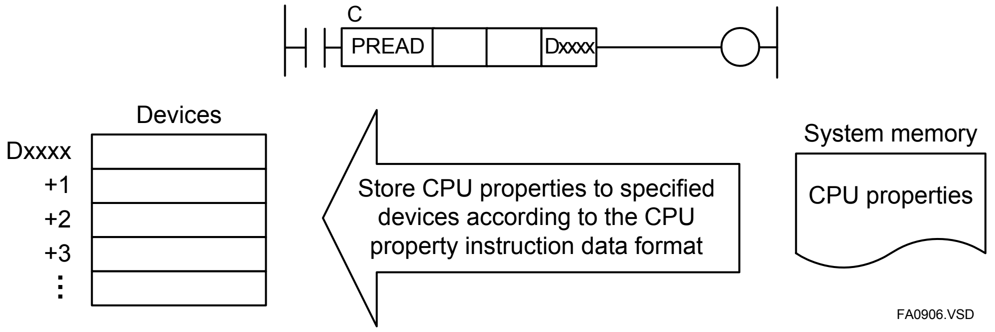

C
PREAD
Dxxxx
Devices
Dxxxx
+1
+2
+3
⋮
Store CPU properties to specified devices according to the CPU property instruction data format
System memory
CPU properties
FA0906.VSD

Figure 3.22.14 Read CPU Properties to Device

The instruction reads current CPU property data in the system memory but not the backup data in the internal ROM. For details on the relationship between CPU property data stored in the system memory and in the internal ROM, see the description for the Write CPU Properties (PWRITE) instruction.

The data read is stored to the output device area designated by the First Output Device for CPU Properties parameter according to the CPU property instruction data format.

SEE ALSO

For details on the format of the returned data, see Section A9.5, "CPU Property Items" of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM 34M06P14-01E) or "Sequence CPU Instruction Manual – Functions (for F3SP71-4N/4S, F3SP76-7N/7S)" (IM 34M06P15-01E).

Before executing the instruction, you must specify values for the setup items listed in the table below in the output device area according to the CPU Property Instruction Data Format. The required setup items are similar to those of the Write CPU Properties (PWRITE) instruction, except that some setup items do not apply to the PREAD instruction. For instance, a destination can be written as an IP address or a hostname but is always read according to the way it was written.

Table 3.22.31 Required Setup in CPU Property Instruction Data Format

Setup Name [Setup No.]	Required Setup Items	Description
Socket address setup [5]	Socket address setting no.	Specify the target socket address setting no. (n) of CPU properties for reading.
	Socket address type	This item is ignored by the PREAD instruction. Data is read according to how it was written.
FTP client address setup [8]	Destination FTP server setting no.	Specify the target destination FTP server setting no. (n) of CPU properties for reading.
	Destination FTP server address type	This item is ignored by the PREAD instruction. Data is read according to how it was written.
Network filter setup [11]	Allowed host setting no.	Specify the target allowed host setting no. (n) of CPU properties for reading.
	Allowed host address type	This item is ignored by the PREAD instruction. Data is read according to how it was written.

If CPU property data is protected with a security keyword, you must specify a valid security keyword as a text parameter.

CAUTION

No timeout interval can be specified for this instruction, which always waits indefinitely. Furthermore, instruction execution always completes regardless of any cancellation request.

■ Programming Example

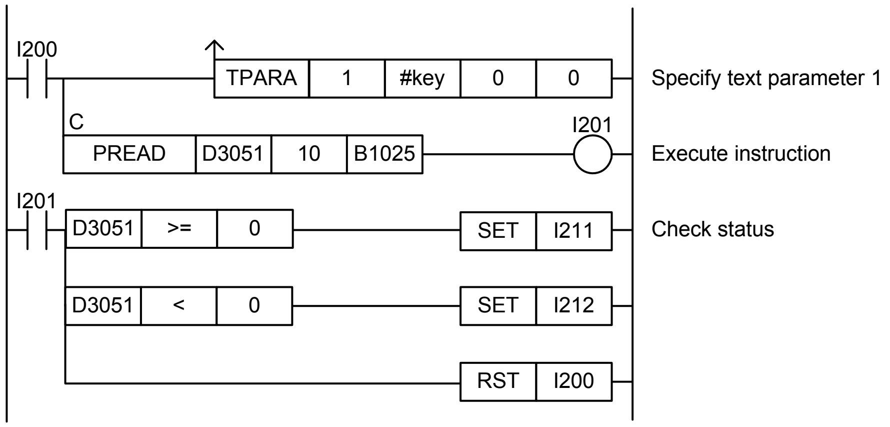


Figure 3.22.15 Read CPU Properties Sample Program

This sample code reads the rotary switch setup of CPU properties data, which is protected. It assumes that the security keyword is defined by constant name #key. The rotary switch setup data is stored to the devices starting from B1025.

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter Name
ret=D3051	0	Status

Appendix 1. Special Relays (M)

Special relays have specific functions, such as indicating the internal state of a sequence CPU module or detecting errors. In programs, these relays are used mainly for contacts A and B.

CAUTION

Do not write to a special relay unless it is marked as "write-enabled". Special relays are used by the sequence CPU module. Writing to these relays incorrectly may lead to system shutdown or other failures. Using forced set/reset instruction in debug mode is also prohibited.

CAUTION

Special relays with index modification cannot be specified as destinations for data output and if specified, will result in instruction processing errors during execution.

CAUTION

Special relays cannot be specified as output destinations in block transfer and table output ladder instructions, and if specified, will cause instruction processing errors during execution.

- Block transfer instructions: BMOV, BSET, SMOV, etc.
- Table output instructions: ULOGR, FIFWR, etc.

Appendix 1.1 Block Start Status Relays

Block Start Status relays indicate which blocks are executed when only specified blocks are executed.
These relays are numbered in ascending order as M001, M002, ... to correlate with block 1, block 2, ...

Table Appendix 1.1 Block Start Status Relays

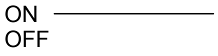
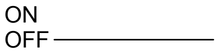
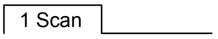
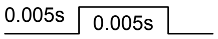
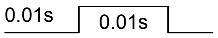
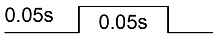
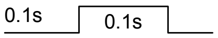
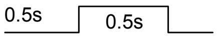
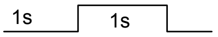
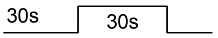
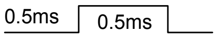
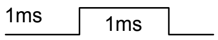
Item	Block Start Status Relays		
No.	Name	Function	Description
M001 to M032	Block n Start Status	ON : Run OFF: Stop	Indicates whether block n is executed when the module is configured to execute specified blocks only.
M2001 to M3024			

Note: The Start Status relays assigned to blocks 1 to 32 are M0001 to M0032 and M2001 to M2032 (M0001 to M0032 have the same values as M2001 to M2032.) Similarly, Start Status relays M2033 to M3024 map to blocks 33 to 1024.

Appendix 1.2 Utility Relays

Utility relays are used to provide timing in a program or issue instructions to the CPU module.

Table Appendix 1.2 Utility Relays

Item	Utility Relays		
No.	Name	Function	Description
M033	Always ON		Used for initialization or as a dummy contact in a program.
M034	Always OFF		
M035	1 Scan ON at Program Start		Turns on for one scan only after a program starts execution
M036 [*1]	0.01 s Clock		Generates a clock pulse of 0.01 s period.
M037 [*1]	0.02 s Clock		Generates a clock pulse of 0.02 s period.
M038 [*1]	0.1 s Clock		Generates a clock pulse of 0.1 s period.
M039 [*1]	0.2 s Clock		Generates a clock pulse of 0.2 s period.
M040 [*1]	1 s Clock		Generates a clock pulse of 1 s period.
M041 [*1]	2 s Clock		Generates a clock pulse of 2 s period.
M042 [*1]	1 min Clock		Generates a clock pulse of 60 s period.
M047 [*1]	1 ms Clock		Generates a clock pulse of 1 ms period.
M048 [*1]	2 ms Clock		Generates a clock pulse of 2 ms period.
M066	Normal Subunit Transmission Line	ON : Normal transmission line or no fiber-optic FA-bus installed OFF: Unspecified or abnormal transmission line	
M097	ON for One Scan at Sensor CB Start	ON : At block start OFF: In all other cases	Turns on for one scan when the sensor control block starts (at the first execution of the sensor control block).

*1: Relays M036 to M048 have their rising and falling clock timing synchronized.

SEE ALSO

For details on the M066 Utility relay (Normal Subunit Transmission Line), see "Fiber-optic FA-bus Module and Fiber-optic FA-bus Type 2 Module, FA-bus Type 2 Module" (IM 34M06H45-01E).

Appendix 1.3 Sequence Operation and Mode Status Relays

Sequence operation and mode status relays indicate the status of sequence operation and various modes.

Table Appendix 1.3 Sequence Operation and Mode Status Relays (1/2)

Item	Sequence Operation and Mode Status Relays		
No.	Name	Function	Description
M079	Press Event	Turns on for 1 scan to report a press event	The User Event Press 1 (or 2) function causes this relay to turn on for one scan cycle when a user presses and releases the SET switch with the MODE switch set to $E (or $F). A program may monitor this relay together with special register Z0117 to detect and process a user-defined event. This is a read-only relay.
M080	Press & Hold Event	Turns on for 1 scan to report a press and hold event	The User Event Press 1 (or 2) function causes this relay to turn on for one scan cycle when a user presses and holds the SET switch with the MODE switch set to $E (or $F). A program may monitor this relay together with special register Z0117 to detect and process a user-defined event. This is a read-only relay.
M113	RDY LED	ON : Lit OFF : Off	Indicates whether the RDY LED is lit or off. Read-only.
M115	RUN LED	ON : Lit OFF : Off	Indicates whether the RUN LED is lit or not lit. Read-only.
M117	ALM LED(1)	ON : Lit OFF : Off	Indicates whether the ALM LED is lit or off. If the LED is blinking, ALM LED (2) is ON. Read-only.
M118	ALM LED(2)	ON : Blinking OFF : Not blinking	
M119	ERR LED(1)	ON : Lit OFF : Off	Indicates whether the ERR LED is lit or off. If the LED is blinking, ERR LED (2) is ON. Read-only.
M120	ERR LED(2)	ON : Blinking OFF : Not blinking	
M121	SD LED(1)	ON : Lit OFF : Off	Indicates whether the SD LED is lit or off. If the LED is blinking, SD LED (2) is ON. Read-only.
M122	SD LED(2)	ON : Blinking OFF : Not blinking	
M123	EXE LED(1)	ON : Lit OFF : Off	Indicates whether the EXE LED is lit or off. If the LED is blinking, EXE LED (2) is ON. Read-only.
M124	EXE LED(2)	ON : Blinking OFF : Not blinking	
M125 (write-enabled)	US1 LED(1)	ON : Lit OFF : Off	Indicates whether the US1 LED is lit or off. If the LED is blinking, US1 LED (2) is ON. You can also manipulate the US1 LED status by writing to this relay.
M126 (write-enabled)	US1 LED(2)	ON : Blinking OFF : Not blinking	
M127 (write-enabled)	US2 LED(1)	ON : Lit OFF : Off	Indicates whether the US2 LED is lit or off. If the LED is blinking, US2 LED (2) is ON. You can also manipulate the US2 LED status by writing to this relay.
M128 (write-enabled)	US2 LED(2)	ON : Blinking OFF : Not blinking	

Table Appendix 1.4 Sequence Operation and Mode Status Relays (2/2)

Item	Sequence Operation and Mode Status Relays		
No.	Name	Function	Description
M129	Run Mode Flag	ON : Run mode OFF: Other modes	Indicates the status of CPU operation.
M130	Debug Mode Flag	ON : Debug mode OFF: Other modes	Indicates the status of CPU operation.
M131	Stop Mode Flag	ON : Stop mode OFF: Other modes	Indicates the status of CPU operation.
M132	Pause Flag	ON : Pause OFF: Run	Indicates the status of program execution during debug mode operation.
M133	Execution Flag	ON : Specified blocks OFF: All blocks	Indicates whether all blocks or specified blocks are executed.
M136	Power-on Operation Flag	ON : Power-on operation OFF: Other modes of operation	Indicates whether operation was initiated by power on or reset
M137	Sensor CB Execution Status	ON : Run OFF: Stop	Indicates the status of sensor control block operation.
M172 (write-enabled)	Set Clock Time	ON : Time being set OFF:	Requests to set clock data.
M173	Input-offline Flag	ON : Offline OFF: Online	Indicates that input refreshing has stopped.
M174	Output-offline Flag	ON : Offline OFF: Online	Indicates that output refreshing has stopped.
M175	Shared-I/O-offline Flag	ON : Offline OFF: Online	Indicates that shared refreshing has stopped.
M176	Link-I/O-offline Flag	ON : Offline OFF: Online	Indicates that link refreshing has stopped.
M188	Carry Flag	ON : Carry enabled OFF: Carry disabled	Carry flag used by shift and rotate operations
M197	Existence of CPU1	ON : Exists. OFF: Does not exist.	Indicates whether or not a CPU exists in slot 1.
M198	Existence of CPU2	ON : Exists. OFF: Does not exist.	Indicates whether or not a CPU exists in slot 2.
M199	Existence of CPU3	ON : Exists. OFF: Does not exist.	Indicates whether or not a CPU exists in slot 3.
M200	Existence of CPU4	ON : Exists. OFF: Does not exist.	Indicates whether or not a CPU exists in slot 4.
M225	CPU1 Sequence Program Execution	ON : Run OFF: Stop	Indicates whether sequence program of CPU in slot 1 is running.
M226	CPU2 Sequence Program Execution	ON : Run OFF: Stop	Indicates whether sequence program of CPU in slot 2 is running.
M227	CPU3 Sequence Program Execution	ON : Run OFF: Stop	Indicates whether sequence program of CPU in slot 3 is running.
M228	CPU4 Sequence Program Execution	ON : Run OFF: Stop	Indicates whether sequence program of CPU in slot 4 is running.
M241	Link Status	ON : Link Up OFF: Link Down	Indicates link status. Works in conjunction with ON/OFF of LNK LED at the front of the module.
M250	CARD1 Mounted	ON : Mounted OFF: Not mounted	Turns on if memory card CARD1 is mounted. Turns off if otherwise.

SEE ALSO

For details on clock data, see Appendix 2, "Special Registers (Z)".

Appendix 1.4 Self-diagnosis Status Relays

Self-diagnosis status relays indicate the results of self-diagnosis by the sequence CPU.

Table Appendix 1.5 Self-diagnosis Status Relays

Item	Self-diagnosis Status Relays		
No.	Name	Function	Description
M193	Self-diagnosis Error	ON : Error OFF: No error	Result of self diagnosis is stored in special registers Z17 to Z19
M194	Battery Error	ON : Error OFF: Normal	Indicates a failure in backup batteries.
M195	Momentary Power Failure	ON : Momentary power failure OFF: No momentary power failure	Indicates that a momentary power failure has occurred.
M196	Inter-CPU Communication Error	ON : Error OFF: Normal	Indicates that a communication failure has occurred in shared relays (E) or shared registers (R).
M201	Instruction Processing Error	ON : Error OFF: No error	Information of instruction processing error is stored in special registers Z22 to Z24.
M202	II/O Comparison Error	ON : Error OFF: Normal	Indicates that the state of module installation is not consistent with the program.
M203	I/O Module Error	ON : Error OFF: Normal	Indicates that no access is possible to I/O modules. The slot number of the error module is stored in special registers Z33 to Z40.
M204	Scan Timeout	ON : Error OFF: Normal	Indicates that scan time has exceeded the scan monitoring time.
M210	Subunit Communication Error	ON : Error OFF: Unspecified or normal line	An error has been detected in the fiber-optic FA-bus module. The slot number of the error module is stored in special registers Z89 to Z96.
M211	Subunit Transmitter Switching Has Occurred	ON : Error OFF: Unspecified or normal line	
M212	Sensor CB Scan Timeout	ON : Error OFF: Normal	Indicates that the execution interval of the sensor control block cannot be maintained.

SEE ALSO

For details on the M210 (Subunit Communication Error) and M211 (Subunit Transmitter Switching Has Occurred) self-diagnosis relays, see "Fiber-optic FA-bus Module and Fiber-optic FA-bus Type 2 Module, FA-bus Type 2 Module" (IM 34M06H45-01E).

Appendix 1.5 FA Link Module Status Relays

FA Link module status relays indicate the status of FA link.

SEE ALSO

For details on FA link module status relays, see the sections on special relays and special registers of “FA Link H Module, Fiber-optic FA Link H Module” (IM 34M06H43-01E),

Table Appendix 1.6 FA Link Module Status Relays

Item	FA Link Module Status Relays		
No.	Name	Function	Description
M257 to M480 M8321 to M8992	FA Link Error	ON : Error OFF: Normal	Indicates the status of FA links.

Appendix 1.6 FL-net Interface Module Status Relays

FL-net interface module status relays indicate the status of FL-net.

Table Appendix 1.7 FL-net Interface Module Status Relays

Item	FL-net Interface Module Status Relays		
No.	Name	Function	Description
M3521 to M3774	Node Participation Status	1: Participating 0: Not participating	FL-net system 1[*1]
M3777 to M4030	Upper Layer Operation Signal Error	1: Error 0: Normal	FL-net system 1[*1]
M4033 to M4286	Operation Status	1: Run 0: Stop	FL-net system 1[*1]
M4289 to M4542	Common Memory Data Valid	1: Valid 0: Invalid	FL-net system 1[*1]
M4561 to M4814	Node Participation Status	1: Participating 0: Not participating	FL-net system 2[*2]
M4817 to M5070	Upper Layer Operation Signal Error	1: Error 0: Normal	FL-net system 2[*2]
M5073 to M5326	Operation Status	1: Run 0: Stop	FL-net system 2[*2]
M5329 to M5582	Common Memory Data Valid	1: Valid 0: Invalid	FL-net system 2[*2]

*1: If both FL-net and FA link are installed, FL-net are allocated smaller system numbers.
*2: If both FL-net and FA ink are installed, FL-net are allocated larger system numbers.

SEE ALSO

For details, see "FL-net (OPCN-2) Interface Module" (IM 34M06H32-02E)

TIP

A system refers to a group of units connected to one FL-net.

Appendix 1.7 Continuous Type Application Instruction Resource Relays

These relays indicate the usage of resources of continuous type application instructions.

SEEL ALSO

For details on continuous type application instruction resource relays, see the description of individual continuous type application instructions.

Table Appendix 1.8 Resource Relays (related to file system instructions)

Category	Continuous Type Application Instruction Resource Relays		
No.	Name	Function	Description
M1026	No Unused File ID	No unused file ID is available.	Turns on when all file IDs are in use. This is a read-only relay. Do not write to it.
M1025	File/Disk Operation Group Busy	File operation instruction group or disk operation instruction group is running.	Turns on during execution of any file operation instruction or disk operation instruction. Execution of any other file operation instruction or disk operation instruction is not allowed while this relay is ON. This relay is not affected by file access instructions. This is a read-only relay. Do not write to it.
M1041 to M1056	File ID Open	File ID is open.	Each file ID is associated with one special relay. The relay for a file ID turns on while the file ID is open. When the relay for a file ID is OFF, no instruction using the file ID can be executed. This is a read-only relay. Do not write to it.
M1057 to M1072	File ID Busy	File ID is busy.	Each file ID is associated with one special relay. The relay for a file ID turns on during execution of any file system instruction using the file ID. When the relay for a file ID is ON, no other file system instruction using the same file ID can be executed. This is a read-only relay. Do not write to it.

Table Appendix 1.9 Resource Relays (related to socket instructions)

Category	Continuous Type Application Instruction Resource Relays		
No.	Name	Function	Description
M1028	No Unused UDP Socket	No unused UDP socket is available.	Turns on when all UDP/IP sockets are in use. This is a read-only relay. Do not write to it.
M1029	No Unused TCP Socket	No unused TCP socket is available.	Turns on when all TCP/IP sockets are in use. This is a read-only relay. Do not write to it.
M1105 to M1120	Socket Open	Socket is open.	Each socket ID is associated with one special relay. The relay for a socket ID turns on while the socket ID is open. When the relay for a socket ID is OFF, the socket ID cannot be used. This is a read-only relay. Do not write to it.
M1121 to M1136	Socket Busy	Socket is busy.	Each socket ID is associated with one special relay. The relay for a socket ID turns on during execution of any socket instruction using the socket ID. When the relay for a socket ID is ON, no other socket communication instruction using the same socket ID can be executed except for concurrent execution of sending and receiving. This is a read-only relay. Do not write to it.
M1073 to M1088	Socket Sending	Socket is performing send processing.	Each socket ID is associated with one special relay. The relay for a socket ID turns on during send processing of the socket. When the relay for a socket ID is ON, no send request is allowed for the same socket ID. This is a read-only relay. Do not write to it.
M1089 to M1104	Socket Receiving	Socket is performing receive processing.	Each socket ID is associated with one special relay. The relay for a socket ID turns on during receive processing of the socket. When the relay for a socket ID is ON, no receive request is allowed for the same socket ID. This is a read-only relay. Do not write to it.

Table Appendix 1.10 Resource Relays (related to FTP Client instructions)

Category	Continuous Type Application Instruction Resource Relays		
No.	Name	Function	Description
M1027	FTP Client Busy	An FTP client instruction is being executed.	This relay turns on during execution of any FTP client instruction. When the relay is ON, no other FTP client instruction can be executed. By inserting this relay in the input condition of a FTP client instruction, you can prevent inadvertent redundant execution. This is a read-only relay. Do not write to it.

Appendix 2. Special Registers (Z)

Special registers have specific functions, such as indicating the internal state of a programmable controller or indicating errors.

CAUTION

- Do not write to a special register (Z), including those not listed in the table above (e.g., Z010 to Z016), unless it is marked as "write-enabled". Special registers are used by the sequence CPU module. Writing to these registers incorrectly may lead to system shutdown or other failures.
- Special registers (Z) with index modification cannot be specified as destinations for data output and if specified, will cause instruction processing errors during execution.
- Special registers (Z) cannot be specified as output destinations in block transfer and table output ladder instructions, and if specified, will cause instruction processing error during execution.

  Block transfer instructions: BMOV, BSET, SMOV, etc.

  Table output instructions: ULOGR, FIFWR, etc.

Appendix 2.1 Sequence Operation Status Registers

Sequence operation status registers indicate the status of sequence operation.

Table Appendix 2.1 Sequence Operation Status Registers (1/2)

Category	Sequence Operation Status Registers		
No.	Name	Function	Description
Z001	Scan Time (Run mode)	Latest scan time	Stores the latest scan time in 100 μs increments.
Z002	Minimum Scan Time (Run mode)	Minimum scan time	Allows the latest scan time to be read in 100 μs increments if it is shorter than the minimum scan time.
Z003	Maximum Scan Time (Run mode)	Maximum scan time.	Allows the latest scan time to be read in 100 μs increments if it is longer than the maximum scan time.
Z004	Scan Time (Debug mode)	Latest scan time	Stores the latest scan time in 100 μs increments.
Z005	Minimum Scan Time (Debug mode)	Minimum scan time	Allows the latest scan time to be read in 100 μs increments if it is shorter than the minimum scan time.
Z006	Maximum Scan Time (Debug mode)	Maximum scan time.	Allows the latest scan time to be read in 100 μs increments if it is longer than the maximum scan time.
Z007	Peripheral-process Scan Time	Latest scan time	Stores the latest scan time in 100 μs increments. (Tolerance: Scan time of one control process)
Z008	Minimum Peripheral-process Scan Time	Minimum scan time	Allows the latest scan time to be read in 100 μs increments if it is shorter than the minimum scan time. (Tolerance: Scan time of one control process)
Z009	Maximum Peripheral-process Scan Time	Maximum scan time.	Allows the latest scan time to be read in 100 μs increments if it is longer than the maximum scan time. (Tolerance: Scan time of one control process)

Table Appendix 2.2 Sequence Operation Status Registers (2/2)

Category	Sequence Operation Status Registers		
No.	Name	Function	Description
Z010	Refresh Peripheral-process Scan Time	Latest scan time.	Stores the latest scan time in 10 μs increments.
Z011	Minimum Refresh Peripheral-process Scan Time	Minimum scan time.	Allows the latest scan time to be read in 10 μs increments if it is shorter than the minimum scan time.
Z012	Maximum Refresh Peripheral-process Scan Time	Maximum scan time.	Allows the latest scan time to be read in 10 μs increments if it is longer than the maximum scan time.

Appendix 2.2 Self-diagnosis Status Registers

Self-diagnosis status registers indicate the results of self-diagnostics by the sequence CPU.

Table Appendix 2.3 Self-diagnosis Status Registers

Category	Self-diagnosis Status Registers		
No.	Name	Function	Description
Z017	Self-diagnosis Error	Self-diagnosis error No.	Stores the results of self-diagnosis.*
Z018		Self-diagnosis error block No.	
Z019		Self-diagnosis error instruction No.	
Z022	Instruction Processing Error	Instruction processing error No.	Stores errors detected during instruction processing.*
Z023		Instruction processing error block No.	
Z024		Instruction processing error instruction No.	
Z027	I/O Comparison Error	I/O comparison error No.	Stores detailed information on I/O comparison error.*
Z028		I/O comparison error block No.	
Z029		I/O comparison error instruction No.	
Z033 to Z040	I/O Error	Slot no. with I/O error 16 2 1 0 ····· 1 0	Stores, as a bit pattern, slot numbers where an I/O error is detected. Z033: Main unit Z034: Subunit 1 Z035: Subunit 2 Z036: Subunit 3 Z037: Subunit 4 Z038: Subunit 5 Z039: Subunit 6 Z040: Subunit 7
Z041	Module Recognition	Main unit	Slot number 16 1 0 ····· 1 0 0 : No modules are recognized. Unable to read/write. 1 : Modules are recognized.
Z042		Subunit 1	
Z043		Subunit 2	
Z044		Subunit 3	
Z045		Subunit 4	
Z046		Subunit 5	
Z047		Subunit 6	
Z048		Subunit 7	
Z089	Subunit Communication Error Slot	Main unit	Slot number 16 1 0 ····· 1 0 Fiber-optic FA-bus module 0: Normal transmission line; Unspecified transmission line; or Loaded with a wrong module 1: Abnormal transmission line (“Subunit communication error” or “Sub unit transmitter switching has occurred)
Z090		Subunit 1	
Z091		Subunit 2	
Z092		Subunit 3	
Z093		Subunit 4	
Z094		Subunit 5	
Z095		Subunit 6	
Z096		Subunit 7	

*: For details on error codes stored in these special registers, see the chapter on "RAS Functions" of "Sequence CPU – Functions User's Manual."

SEE ALSO

For details on the Z089 to Z096 special registers (Communication error slot), see "Fiber-optic FA-bus Module and Fiber-optic FA-bus Type 2 Module, FA-bus Type 2 Module" (IM 34M06H45-01E).

Appendix 2.3 Utility Registers

Table Appendix 2.4 Utility Registers

Category	Utility Registers		
No.	Name	Function	Description
Z049 (write-enabled)	Clock Data	Last two digits of calendar year	Stores "year" as a BCD-coded value. e.g. 1999 as $0099 2000 as $0000
Z050 (write-enabled)		Month	Stores "month" as a BCD-coded value. e.g. January as $0001
Z051 (write-enabled)		Day of month	Stores "day of month" as a BCD-coded value. e.g. 28th as $0028
Z052 (write-enabled)		Hour	Stores "hour" as a BCD-coded value. e.g. 18:00 hours as $0018
Z053 (write-enabled)		Minute	Stores "minute" as a BCD-coded value. e.g. 15 minutes as $0015
Z054 (write-enabled)		Second	Stores "second" as a BCD-coded value. e.g. 30 seconds as $0030
Z055		Day of week ($0000 to $0006)	Stores "day of week" as a BCD-coded value. e.g. Wednesday as $0003
Z056	Constant Scan Time	Value of constant scan time	0.1 ms increments e.g. 10 ms as 100
Z057	Constant Scan Time	Value of constant scan time	1 ms increments e.g. 10 ms as 10
Z058	Scan Monitoring Time	Value of scan monitoring time	1 ms increments e.g. 200 ms as 200

You can set clock data using the Set Date instruction (DATE), Set Time instruction (TIME), Set Date String instruction (SDATE), and Set Time String instruction (STIME).

- **Procedure for Setting Clock Data without Using Ladder Instructions**

  1. Write the clock data to special registers Z049 to Z054
     (use a MOV P instruction. Using BMOV or BSET instructions will generate an instruction error).
  2. Set special relay M172 to ON within the same scan as step (1)
     (use a DIFU instruction).
  3. Set special relay M172 to OFF in the scan subsequent to step (2).
     Stop writing the clock data to special registers Z049 to Z054 in the same scan.

  Note that no change will be made to clock data, which reverts to its original value if the setup value is invalid.

- **Accuracy of Clock Data**

  The accuracy of clock data is specified as:
  Maximum daily error = ±8 s (±2 s, when actually measured)
  The clock accuracy is reset to the maximum daily error of -1.2 s/+2 s, however, when the power is turned off and on again. In addition, you can input a correction value from the programming tool. If you specify an appropriate correction value, the clock data is corrected during the power-off-and-on sequence, thus offsetting the cumulative error.

Appendix 2.4 FA Link Module Status Registers

FA Link module status registers indicate the status of FA links.

SEE ALSO

For details on the FA link module status registers, see Special relays/registers sections in "FA Link H Module, Fiber-optic FA Link H Module" (IM 34M06H43-01E).

Table Appendix 2.5 FA Link Module Status Registers

Category	FA Link Module Status		
No.	Name	Function	Description
Z075	Local Station No.		System 1 (FA link)
Z076	Local Station No.		System 2 (FA link)
Z077	Local Station No.		System 3 (FA link)
Z078	Local Station No.		System 4 (FA link)
Z079	Local Station No.		System 5 (FA link)
Z080	Local Station No.		System 6 (FA link)
Z081	Local Station No.		System 7 (FA link)
Z082	Local Station No.		System 8 (FA link)
Z065	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 1 (FA link)
Z066	Cyclic Transmission Time		System 1 (FA link) 1 ms increments
Z070	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 2 (FA link)
Z071	Cyclic Transmission Time		System 2 (FA link) 1 ms increments
Z257	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 3 (FA link)
Z258	Cyclic Transmission Time		System 3 (FA link) 1 ms increments
Z262	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 4 (FA link)
Z263	Cyclic Transmission Time		System 4 (FA link) 1 ms increments
Z267	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 5 (FA link)
Z268	Cyclic Transmission Time		System 5 (FA link) 1 ms increments
Z272	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 6 (FA link)
Z273	Cyclic Transmission Time		System 6 (FA link) 1 ms increments
Z277	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 7 (FA link)
Z278	Cyclic Transmission Time		System 7 (FA link) 1 ms increments
Z282	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 8 (FA link)
Z283	Cyclic Transmission Time		System 8 (FA link) 1 ms increments

TIP

Units that make up a system are known as stations.

Appendix 2.5 Sequence CPU Module Status Registers

CPU module status registers indicate the status of a CPU.

Table Appendix 2.6 Sequence CPU Module Status Registers

Category	Sequence CPU Module Status Registers		
No.	Name	Function	
Z105	Number of User Log Records	For details on user log, see Section A6, "User Log Management Function" of "Sequence CPU – Functions User's Manual."	
Z109	Sensor CB Execution Time	Time taken from starting of input refreshing for the sensor control block through program execution to completion of output refreshing. (Unit: 10 μs)	
Z111	Maximum Sensor CB Execution Time	The maximum time taken to execute the sensor control block. (Unit: 10 μs)	
Z113 (write-enabled)	Number of Invalid Accesses	Counts the number of connection requests received from IP addresses that are not registered with the network filter function. The counter is incremented from 0 to 65535, and resets to zero when it exceeds 65535. To reset the counter value, write a zero value to the register, or restore the module to factory settings.	
Z114	MAC Address	MAC address (low word)	Low-order 16 bits [$xxxx]
Z115		MAC address (mid word)	Mid-order 16 bits [$64xx]
Z116		MAC address (high word)	High-order 16 bits [$0000]
Z117	MODE Switch	Stores the MODE switch value. Its value is updated when the SET switch is pressed or pressed and held. Its value does not change within the same scan. Read-only.	
Z121～Z128[*1]	Model Information	CPU model name and revision number of firmware.	
Z130[*4]	Sampling Trace Implementation Status	F3SP7□-□N: -1: Trigger not set 0: Waiting for a trigger 1 to 99: Progress of trace (%) 100: End of trace F3SP7□-□S: -1: Trigger not set 0: Waiting for a start trigger 1 to 98[*2]: Progress of trace (%) 99[*3]: Writing to a file 100: End of trace	
Z131[*5]	Number of Implemented Sampling Cycles	0 to 100: Number of cycles	
Z657[*4]	Monitoring Time for Continuous Write to Operation Log	Used to change the setting time for the monitoring of continuous write to an operation log. Continuous write to a same operation log via the same route is not performed for (Monitoring Time for Continuous Write to Operation Log * 10) ms. When the value is set to 0, it defaults to 1 sec. (10-ms increments)	
Z658[*4]	Number of automatic operation log outputs	Increases when the operation log file name is changed from "new_operation.yolg" to "old_operation.yolg" while the automatic operation log output to a SD card is enabled.	

*1: For module "F3SP67-6S" with firmware Rev1,
Z121 "F3"
Z122 "SP"
Z123 "67"
Z124 "6S"
Z125 "/R"
Z126 "01"
Z127 "/ "
Z128 " "

*2: Progress of trace is (Number of implemented sampling cycles/Specified number of sampling cycles (S_MAX) x 100) %.

*3: The value becomes 99 only while writing to file.

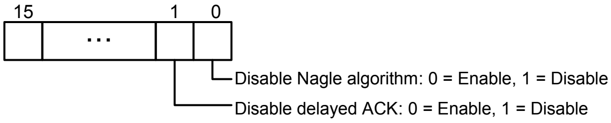
*4: These are supported by only F3SP71-4N, F3SP76-7N, F3SP71-4S and F3SP76-7S.

*5: These are supported by only F3SP71-4S and F3SP76-7S.

Appendix 2.6 Socket Status Registers

Socket status registers are special registers related to the TCP/IP socket ID status.

Table Appendix 2.7 Socket Status Registers

Category	Socket Status	
No.	**Name**	**Description**
Z308	SOCKET ID8 Socket status	 15 1 0 … Disable Nagle algorithm: 0 = Enable, 1 = Disable Disable delayed ACK: 0 = Enable, 1 = Disable
Z309	SOCKET ID9 Socket status	
Z310	SOCKET ID10 Socket status	
Z311	SOCKET ID11 Socket status	
Z312	SOCKET ID12 Socket status	
Z313	SOCKET ID13 Socket status	
Z314	SOCKET ID14 Socket status	
Z315	SOCKET ID15 Socket status	

SEE ALSO

For details on the socket status registers, see “■Socket Option (SOCKOPT)” of “Sequence CPU – Network Functions (for F3SP71-4N/4S, F3SP76-7N/7S)“ (IM 34M06P15-02E).

Blank Page

Appendix 3. List of Ladder Sequence Instructions

- For basic instructions, the execution time values given in the following table apply independent of the input conditions.
- For the Compare, Compare Long-word, Compare Double Long-word, Compare Float, and Compare Double-precision Float instructions, the execution time values given in the following table apply independent of the input conditions.
- For the other application instructions, the execution time values given in the following table apply to the processing time in active state.
- For the other application instructions (excluding those of the differential type), the processing time in inactive state can be calculated as follows:

F3SP05, F3SP08, and F3SP21	0.18 μs × step count
F3SP25	0.12 μs × step count
F3SP35, F3SP22, F3SP28, and F3SP38	0.09 μs × step count
F3SP53, F3SP58, F3SP66, and F3SP67	0.035 μs × step count
F3SP71, F3SP76	0.0075 μs × (step count + 1 + n)

n: Number of index modification parameters

- For differential-type application instructions, the processing time in inactive state can be calculated as follows:

F3SP05, F3SP08, and F3SP21	0.18 μs × (step count +2)
F3SP25	0.12 μs × (step count +2)
F3SP35, F3SP22, F3SP28, and F3SP38	0.09 μs × (step count +2)
F3SP53, F3SP58, F3SP66, and F3SP67	0.035 μs × (step count +2)
F3SP71, F3SP76	0.0075 μs × (step count + 2 + n)

n: Number of index modification parameters

■ Ladder Sequence Basic Instructions

Classifi-cation	Instruction	Mnemonic	Step Count		Condition	Instruction Execution Time (μs)					
			Without index modification	With index modification		F3SP05 F3SP08 F3SP21	F3SP25	F3SP35	F3SP22 F3SP28 F3SP38	F3SP53 F3SP58 F3SP59 F3SP66 F3SP67	F3SP71 F3SP76
Basic Instructions	Load	LD	1	2	*1	—	—	—	0.045	0.0175	—
						0.18	0.12	0.09	0.09	0.035	0.00375
	Load Not	LDN	1	2	*1	—	—	—	0.045	0.0175	—
						0.18	0.12	0.09	0.09	0.035	0.00375
	And	AND	1	2	*1	—	—	—	0.045	0.0175	—
						0.18	0.12	0.09	0.09	0.035	0.00375
	And Not	ANDN	1	2	*1	—	—	—	0.045	0.0175	—
						0.18	0.12	0.09	0.09	0.035	0.00375
	Or	OR	1	2	*1	—	—	—	0.045	0.0175	—
						0.18	0.12	0.09	0.09	0.035	0.00375
	Or Not	ORN	1	2	*1	—	—	—	0.045	0.0175	—
						0.18	0.12	0.09	0.09	0.035	0.00375
	Load Differential Up	LDU	2	3		—	—	—	0.27	0.105	0.015
	Load Differential Down	LDD	2	3		—	—	—	0.27	0.105	0.015
	And Load	ANDLD	1	—		0.18	0.12	0.09	0.09	0.035	0.00375
	Or Load	ORLD	1	—		0.18	0.12	0.09	0.09	0.035	0.00375
	Inverter	INV	1	—		—	—	—	0.09	0.035	0.0075
	Logical Differential Up	UP	1	—		—	—	—	0.27	0.105	0.0075
	Logical Differential Down	DWN	1	—		—	—	—	0.18	0.07	0.0075
	Logical Differential Up Using Specified Device	UPX	1	2		—	—	—	0.27	0.105	0.0075
	Logical Differential Down Using Specified Device	DWNX	1	2		—	—	—	0.18	0.07	0.0075
	Out	OUT	1	2	*2	—	—	—	0.09	0.035	—
						0.36			0.18	0.07	0.00375
	Out Not	OUTN	1	2	*2	—	—	—	0.09	0.035	—
						0.90			0.18	0.07	0.00375
	Flip-Flop	FF	2	3		—	—	—	0.45	0.175	0.19
	Push	PUSH	1	—	*1	—	—	—	0.045	0.0175	—
						0.18	0.12	0.09	0.09	0.035	0.00375
	Stack	STCRD	1	—		0.18	0.12	0.09	0.09	0.035	0.00375
	Pop	POP	1	—		0.18	0.12	0.09	0.09	0.035	0.00375
	Set	SET	1	2	Execute-while-ON	0.36	0.24	0.18	0.18	0.07	0.0075
			2	3	Differential	0.72	0.48	0.36	0.36	0.14	0.015
	Reset	RST	1	2	Execute-while-ON	0.36	0.24	0.18	0.18	0.07	0.0075
			2	3	Differential	0.72	0.48	0.36	0.36	0.14	0.015
	Timer	TIM	2/4[*3]	2/4[*3]		0.72	0.48	0.36	0.36	0.175	0.2
	Counter	CNT	2	3		0.72	0.48	0.36	0.36	0.175	0.4
	Differential Up	DIFU	2	3	*2	—	—	—	0.18	0.07	—
						0.54	0.36	0.27	0.27	0.105	0.015
	Differential Down	DIFD	2	3	*2	—	—	—	0.18	0.07	—
						0.54	0.36	0.27	0.27	0.105	0.015
	Interlock	IL	1	—		0.18	0.12	0.09	0.09	0.035	0.0075
	Interlock Clear	ILC	1	—		0.18	0.12	0.09	0.09	0.035	0.0075
	Load Specified Bit	LDW	1	2		—	—	—	0.18	0.07	0.0075
	Out Specified Bit	OUTW	1	2	Execute-while-ON	—	—	—	0.27	0.105	0.0075
			2	3	Differential	—	—	—	0.54	0.21	0.015
	Set Specified Bit	SETW	1	2	Execute-while-ON	—	—	—	0.27	0.105	0.0075
			2	3	Differential	—	—	—	0.54	0.21	0.015
	Reset Specified Bit	RSTW	1	2	Execute-while-ON	—	—	—	0.27	0.105	0.0075
			2	3	Differential	—	—	—	0.54	0.21	0.015
	Off-Delay	OFDLY	4	4		—	—	—	—	—	3.0
	On-Delay	ONDLY	4	4		—	—	—	—	—	3.0
	Pulse	PULSE	4	4		—	—	—	—	—	3.0
	End	END	1	2		—	—	—	—	—	—
	Nop	NOP	1	2		0.18	0.12	0.09	0.09	0.035	0.0075

*1: If a PUSH (start a branch) instruction follows immediately after a LD, LDN, AND, ANDN, OR, or ORN instruction

*2: If a STCRD (effect a branch) or POP (end a branch) instructions follows immediately after an OUT, OUTN, DIFU, or DIFD instruction.

*3: 4 steps for F3SP71-4N, F3SP76-7N, and F3SP☐☐-☐S

■ Ladder Sequence Application Instructions (1/5)

Classification	Instruction		Mnemonic	Step Count			Condition	Instruction Execution Time (μs)					
				Always-execute	Execute-while-ON	Differential		F3SP05 F3SP08 F3SP21	F3SP25	F3SP35	F3SP22 F3SP28 F3SP38	F3SP53 F3SP58 F3SP59 F3SP66 F3SP67	F3SP71 F3SP76
Comparison Instructions	Compare		CMP	3	—	—	High speed proc.	0.36	0.24	0.18	0.18	0.07	0.015
								3.8	2.5	1.9	1.35	0.525	1.4
	Compare Long-word		CMP L	3	—	—	High speed proc.	—	—	—	0.45	0.175	0.015
								5.6	3.7	2.8	1.89	0.735	1.4
	Compare Double Long-word	=, < >	CMP D	5	—	—	High speed proc.	—	—	—	—	—	0.0525
								—	—	—	—	—	0.4
		>, > =, <, < =					High speed proc.	—	—	—	—	—	0.0825
								—	—	—	—	—	0.5
	Compare Float		FCMP	4	—	—	High speed proc.	—	—	—	—	—	0.015
								—	8.7	6.5	6.5	2.5	1.5
	Compare Double-precision Float		FCMP E	5	—	—	High speed proc.	—	—	—	—	—	0.045
								—	—	—	—	—	0.85
	Table Compare		BCMP	—	5	6	n=0	13.6	9.1	6.8	6.8	2.6	2.5
							n=999	4ms	2.7ms	2ms	2ms	0.8ms	338.7
	Table Compare Long-word		BCMP L	—	5	6	n=0	15.2	10.1	7.6	7.6	3.0	2.7
							n=999	6ms	4ms	3ms	3ms	1.2ms	472.2
	Table Compare Float		FBCP	—	5	6	n=0	—	48.0	36.0	36.0	14.0	3.0
							n=999	—	38.3ms	28.7ms	28.7ms	11.2ms	990.1
	Table Search		TSRCH	—	5	6	n=0	11.8	7.9	5.9	5.9	2.3	2.4
							n=999	2.6ms	1.7ms	1.3ms	1.3ms	0.5ms	230.4
	Long-word Table Search		TSRCH L	—	5	6	n=0	11.8	7.9	5.9	5.9	2.3	2.4
							n=999	2.6ms	1.7ms	1.3ms	1.3ms	0.5ms	240.6
Arithmetic Instructions	Add		CAL	—	4	5	High speed proc.	0.54	0.36	0.27	0.27	0.105	0.015
								12.0	8.0	6.0	6.0	2.4	1.6
	Subtract						High speed proc.	0.54	0.36	0.27	0.27	0.105	0.015
								12.0	8.0	6.0	6.0	2.4	1.6
	Multiply						High speed proc.	—	—	—	0.81	0.245	0.03
								10.2	6.8	5.1	5.1	2.0	1.6
	Divide						High speed proc.	—	—	—	1.62	0.63	0.0825
								13.6	9.1	6.8	6.8	2.7	1.9
	Add Long-word		CAL L	—	4	5	High speed proc.	—	—	—	1.81	0.315	0.015
								11.2	7.5	5.6	5.6	2.2	1.6
	Subtract Long-word						High speed proc.	—	—	—	1.81	0.315	0.015
								11.2	7.5	5.6	5.6	2.2	1.6
	Multiply Long-word						High speed proc.	—	—	—	4.1	1.6	0.0825
								20.8	13.9	12.5	12.5	4.9	1.9
	Divide Long-word						High speed proc.	—	—	—	4.9	1.9	0.1725
								23.6	15.7	11.8	11.8	4.6	2.7
	Add Double Long-word		CAL D	—	6	7	High speed proc.	—	—	—	—	—	0.03
								—	—	—	—	—	1.4
	Subtract Double Long-word						High speed proc.	—	—	—	—	—	0.03
								—	—	—	—	—	1.2
	Multiply Double Long-word							—	—	—	—	—	4.8
	Divide Double Long-word							—	—	—	—	—	38.9
	Add Float		FCAL	—	5	6	High speed proc.	—	—	—	—	—	0.0375
								—	20.9	15.7	15.7	6.1	1.9
	Subtract Float						High speed proc.	—	—	—	—	—	0.0375
								—	24.0	18.0	18.0	7.0	1.9
	Multiply Float						High speed proc.	—	—	—	—	—	0.045
								—	25.3	19.0	19.0	7.4	1.9
	Divide Float						High speed proc.	—	—	—	—	—	0.1275
								—	21.2	15.9	15.9	6.2	3.0
	Add Double-precision Float		FCAL E	—	6	7	High speed proc.	—	—	—	—	—	0.0825
								—	—	—	—	—	0.5
	Subtract Double-precision Float						High speed proc.	—	—	—	—	—	0.0825
								—	—	—	—	—	0.5
	Multiply Double-precision Float						High speed proc.	—	—	—	—	—	0.1125
								—	—	—	—	—	0.5
	Divide Double-precision Float						High speed proc.	—	—	—	—	—	0.285
								—	—	—	—	—	0.5

■ Ladder Sequence Application Instructions (2/5)

Classifi-cation	Instruction	Mnemonic	Step Count			Condition	Instruction Execution Time (μs)					
			Always-execute	Execute-while-ON	Differential		F3SP05 F3SP08 F3SP21	F3SP25	F3SP35	F3SP22 F3SP28 F3SP38	F3SP53 F3SP58 F3SP59 F3SP66 F3SP67	F3SP71 F3SP76
Arithmetic Instructions	Increment	INC	—	2	3	High speed proc.	—	—	—	0.27	0.105	0.0075
							3.0	2.0	1.5	1.5	0.59	1.2
	Increment Long-word	INC L	—	2	3	High speed proc.	—	—	—	—	—	0.0075
							4.4	2.9	2.2	2.2	0.86	1.4
	Decrement	DEC	—	2	3	High speed proc.	—	—	—	0.27	0.105	0.0075
							3.0	2.0	1.5	1.5	0.59	1.2
	Decrement Long-word	DEC L	—	2	3	High speed proc.	—	—	—	—	—	0.0075
							4.4	2.9	2.2	2.2	0.86	1.4
	Square Root	SQR	—	2	3	D=1	8.2	5.5	4.1	4.1	1.3	0.4
						D=1000	51.4	34.3	25.7	25.7	5.0	0.4
	Square Root Long-word	SQR L	—	2	3	D=1	14.6	9.7	7.3	7.3	1.3	0.5
						D=1000	169.2	112.8	84.6	84.6	5.0	0.5
	Square Root Double Long-word	SQR D	—	3	4	D=1	—	—	—	—	—	1.2
						D=1000	—	—	—	—	—	1.2
	Square Root Float	FSQR	—	4	5		—	132	99	99	38.5	1.5
	Square Root Double-precision Float	FSQR E	—	4	5		—	—	—	—	—	2.1
	SIN	FSIN	—	4	5		—	296	222	222	86	60.0
	COS	FCOS	—	4	5		—	296	222	222	86	64.0
	TAN	FTAN	—	4	5		—	339	254	254	99	55.0
	SIN^{-1}	FASIN	—	4	5		—	557	418	418	163	57.0
	COS^{-1}	FACOS	—	4	5		—	557	433	433	168	54.7
	TAN^{-1}	FATAN	—	4	5		—	311	233	233	91	50.5
	LOG	FLOG	—	4	5	Log1	—	55.3	41.5	41.5	15.1	77.0
						Log10	—	364	273	273	125	77.0
	EXP	FEXP	—	4	5		—	376	282	282	110	14.0
Logical Instructions	Logical AND	CAL	—	4	5	High speed proc.	0.54	0.36	0.27	0.27	0.105	0.015
							7.8	5.2	3.9	3.9	1.5	1.8
	Logical OR					High speed proc.	0.54	0.36	0.27	0.27	0.105	0.015
							7.8	5.2	3.9	3.9	1.5	1.8
	Logical XOR					High speed proc.	0.54	0.36	0.27	0.27	0.105	0.015
							7.8	5.2	3.9	3.9	1.5	1.8
	Logical NXOR					High speed proc.	0.54	0.36	0.27	0.27	0.105	0.015
							7.8	5.2	3.9	3.9	1.5	1.8
	Logical AND Long-word	CAL L	—	4	5	High speed proc.	—	—	—	0.81	0.315	0.015
							11.4	7.6	5.7	5.7	2.2	2.0
	Logical OR Long-word					High speed proc.	—	—	—	—	—	0.015
							11.4	7.6	5.7	5.7	2.2	2.0
	Logical XOR Long-word					High speed proc.	—	—	—	—	—	0.015
							11.4	7.6	5.7	5.7	2.2	2.0
	Logical NXOR Long-word					High speed proc.	—	—	—	—	—	0.015
							11.4	7.6	5.7	5.7	2.2	2.0
	Two's Complement	NEG	—	2	3	High speed proc.	—	—	—	—	—	0.0075
							3.2	2.1	1.6	1.6	0.6	1.1
	Two's Complement Long-word	NEG L	—	2	3	High speed proc.	—	—	—	—	—	0.0075
							8.2	5.5	4.1	4.1	1.6	1.4
	Not	NOT	—	2	3	High speed proc.	—	—	—	—	—	0.0075
							3.0	2.0	1.5	1.5	0.6	1.2
	Not Long-word	NOT L	—	2	3	High speed proc.	—	—	—	—	—	0.0075
							8.0	5.3	4.0	4.0	1.6	1.4
Rotate Instructions	Right Rotate	RROT	—	3	4	High speed proc.	—	—	—	—	—	0.09
							9.2	6.1	4.6	4.6	1.8	1.5
	Right Rotate Long-word	RROT L	—	3	4	High speed proc.	—	—	—	—	—	0.09
							12.0	8.0	6.0	6.0	2.3	1.8
	Left Rotate	LROT	—	3	4	High speed proc.	—	—	—	—	—	0.09
							9.2	6.1	4.6	4.6	1.8	1.5
	Left Rotate Long-word	LROT L	—	3	4	High speed proc.	—	—	—	—	—	0.09
							13.0	8.7	6.5	6.5	2.5	1.8
	Right Rotate with Carry	RROTC	—	3	4		12.2	8.1	6.1	6.1	2.4	1.1
	Right Rotate Long-word with Carry	RROTC L	—	3	4		14.6	9.7	7.3	7.3	2.8	1.1
	Left Rotate with Carry	LROTC	—	3	4		11.8	7.9	5.9	5.9	2.3	1.1
	Left Rotate Long-word with Carry	LROTC L	—	3	4		14.8	9.6	7.2	7.2	2.8	1.1

■ Ladder Sequence Application Instructions (3/5)

Classifi-cation	Instruction	Mnemonic	Step Count			Condition	Instruction Execution Time (μs)					
			Always-execute	Execute-while-ON	Differential		F3SP05 F3SP08 F3SP21	F3SP25	F3SP35	F3SP22 F3SP28 F3SP38	F3SP53 F3SP58 F3SP59 F3SP66 F3SP67	F3SP71 F3SP76
Shift Instructions	Right Shift	RSFT	—	3	4	High speed proc.	—	—	—	0.90	0.35	0.09
							10.0	6.7	5.0	5.0	1.9	1.6
	Right Shift Long-word	RSFT L	—	3	4	High speed proc.	—	—	—	—	—	0.09
							12.6	8.4	6.3	6.3	2.5	1.9
	Right Shift m-bit Data by n bits	RSFTN	—	4	5		28.2	19.2	14.4	14.4	5.6	3.3
	Left Shift	LSFT	—	3	4	High speed proc.	—	—	—	0.90	0.35	0.09
							10.0	6.7	5.0	5.0	1.9	1.6
	Left Shift Long-word	LSFT L	—	3	4		—	—	—	—	—	0.09
							13.2	8.8	6.6	6.6	2.6	1.9
	Left Shift m-bit Data by n bits	LSFTN	—	4	5		35.6	23.7	17.8	17.8	6.9	3.5
	Shift Register	SFTR	—	—	4	n=16	29.0	20.0	14.7	14.7	4.2	16.9
						n=160	99.0	61.7	45.8	45.8	11.3	109.5
Data Transfer Instructions	Move	MOV	—	3	4	High speed proc.	0.36	0.24	0.18	0.18	0.07	0.0075
							6.4	4.3	3.2	3.2	1.2	1.2
	Move Long-word	MOV L	—	3	4	High speed proc.	—	—	—	0.36	0.14	0.0075
							8.4	5.6	4.2	4.2	1.6	1.4
	Move Double Long-word	MOV D	—	4	5	High speed proc.	—	—	—	—	—	0.0225
							—	—	—	—	—	1.3
	Partial Move	PMOV	—	4	5		11.8	7.9	5.9	5.9	2.3	0.7
	Block Move	BMOV	—	4	5	n=1	15.8	10.5	7.9	4.6	1.8	0.9
						n=2048	9.2ms	6.1ms	4.6ms	0.56ms	0.22ms	57.1
	Block Set	BSET	—	4	5	n=1	15.8	15.8	7.9	4.9	1.9	0.7
						n=2048	5ms	5ms	2.5ms	0.56ms	0.22ms	41.5
	Right word Shift	RWS	—	3	4	n=1	11.0	10.4	5.5	5.5	2.1	1.4
						n=1000	3.2ms	3.3ms	1.6ms	1.6ms	0.6ms	162.5
	Left Word Shift	LWS	—	3	4	n=1	11.0	10.4	5.5	5.5	2.1	1.4
						n=1000	3.2ms	3.3ms	1.6ms	1.6ms	0.6ms	162.5
	Indexed Move	IXMOV	—	5	6		19.0	12.7	9.5	9.5	3.7	2.7
	Indexed Move Long-word	IXMOV L	—	5	6		20.2	13.5	10.1	10.1	3.9	2.8
	Exchange	XCHG	—	3	4		6.4	4.3	3.2	3.2	1.2	0.8
	Exchange Long-word	XCHG L	—	3	4		8.2	5.6	4.1	4.1	1.6	0.9
	Negated Move	NMOV	—	3	4		4.8	3.2	2.4	2.4	0.9	0.5
	Negated Move Long-word	NMOV L	—	3	4		9.8	6.5	4.9	4.9	1.9	0.5
	Extended Partial Move	PMOVX	—	5	6		9.4	6.3	4.7	4.7	1.8	1.6
	Bit Move	BITM	—	5	6		8.2	5.5	4.1	4.1	1.6	1.3
	Digit Move	DGTM	—	5	6		9.6	6.4	4.8	4.8	1.9	1.3
	Block Swap Move*	BSWAP	—	4	5	n=1	—	—	—	—	—	1.9
						n=100	—	—	—	—	—	11.5
	Byte Index Move*	BIXMV	—	4	5	t+0=1 t+1=1 t+2=2	—	—	—	—	—	4.0
						t+1=100 t+1=1 t+2=2	—	—	—	—	—	9.0
Data Processing Instructions	FIFO Read	FIFRD	—	3	4		18.4	12.3	9.2	9.2	3.6	1.8
	FIFO Write	FIFWR	—	3	4		17.8	11.9	8.9	8.9	3.5	1.8
	Binary Conversion	BIN	—	3	4	High speed proc.	—	—	—	—	—	0.015
							8.0	5.3	4.0	4.0	1.6	1.5
	Long-word Binary Conversion	BIN L	—	3	4	High speed proc.	—	—	—	—	—	0.0225
							10.8	7.2	5.4	5.4	2.1	1.7
	BCD Conversion	BCD	—	3	4	High speed proc.	—	—	—	—	—	0.0375
							9.0	6.0	4.5	4.5	1.8	1.6
	Long-word BCD Conversion	BCD L	—	3	4	High speed proc.	—	—	—	—	—	0.0525
							15.4	10.3	7.7	7.7	3.0	2.3
	Float to BCD	FBCD	—	5	6		—	75.5	56.6	56.6	22.0	5.1
	BCD to Float	BCDF	—	5	6		—	78.5	58.9	58.9	22.9	5.7
	Integer to Float	ITOF	—	4	5	High speed proc.	—	—	—	—	—	0.03
							—	20.4	15.3	15.3	6.0	2.0
	Long-word Integer to Float	ITOF L	—	4	5	High speed proc.	—	—	—	—	—	0.03
							—	20.3	15.2	15.2	5.9	1.5
	Long-word Integer to Double-precision Float	ITOE L	—	4	5	High speed proc.	—	—	—	—	—	0.0525
							—	—	—	—	—	1.4
	Double Long-word Integer to Double-precision Float	ITOE D	—	4	5	High speed proc.	—	—	—	—	—	0.06
							—	—	—	—	—	1.3
	Float to Integer	FTOI	—	4	5	High speed proc.	—	—	—	—	—	0.0225
							—	10.0	7.5	7.5	2.9	1.5
	Float to Long-word Integer	FTOI L	—	4	5	High speed proc.	—	—	—	—	—	0.0255
							—	9.5	7.1	7.1	2.8	2.1
	Double-precision Float to Long-word Integer	ETOI L	—	4	5	High speed proc.	—	—	—	—	—	0.0375
							—	—	—	—	—	1.9

*: F3SP7□-□S only

■ Ladder Sequence Application Instructions (4/5)

Classification	Instruction	Mnemonic	Step Count			Condition	Instruction Execution Time (μs)					
			Always-execute	Execute-while-ON	Differential		F3SP05 F3SP08 F3SP21	F3SP25	F3SP35	F3SP22 F3SP28 F3SP38	F3SP53 F3SP58 F3SP59 F3SP66 F3SP67	F3SP71 F3SP76
Data Processing Instructions	Double-precision Float to Double Long-word Integer	ETOI D	—	4	5	High speed proc.	—	—	—	—	—	0.0525
							—	—	—	—	—	1.4
	Float to Double-precision Float	FTOE	—	4	5	High speed proc.	—	—	—	—	—	0.0525
							—	—	—	—	—	1.4
	Double-precision Float to Float	ETOF	—	4	5	High speed proc.	—	—	—	—	—	0.045
							—	—	—	—	—	1.9
	Convert Degree to Radian	FRAD	—	4	5		—	25.7	19.3	19.3	7.5	1.0
	Convert Radian to Degree	FDEG	—	4	5		—	26.8	20.1	20.1	7.8	1.5
	7-Segment Decoder	SEG	—	4	5		7.8	5.2	3.9	3.9	1.5	1.2
	Convert ASCII	ASC	—	4	5		7.8	5.2	3.9	3.9	1.5	1.2
	Bit set	BITS	—	3	4		5.0	3.3	2.5	2.5	1.0	0.5
	Long-word Bit Set	BITS L	—	3	4		6.6	4.4	3.2	3.2	1.3	1.0
	Bit Reset	BITR	—	3	4		5.0	3.3	2.5	2.5	1.0	0.5
	Long-word Bit Reset	BITR L	—	3	4		6.6	4.4	3.3	3.3	1.3	1.0
	Carry Set	CSET	—	1	2		1.2	0.8	0.6	0.6	0.2	0.2
	Carry Reset	CRST	—	1	2		1.2	0.8	0.6	0.6	0.2	0.2
	Distribute Data	DIST	—	3	4		15.6	10.4	7.8	7.8	3.0	1.7
	Distribute Long-word Data	DIST L	—	3	4		27.2	18.1	13.6	13.6	0.4	3.4
	Unit Data	UNIT	—	3	4		18.0	12.0	9.0	9.0	3.5	1.6
	Unit Long-word Data	UNIT L	—	3	4		28.0	18.7	14.0	14.0	5.4	2.5
	Decode	DECO	—	5	6	n=1	19.8	13.1	9.8	9.8	3.8	2.4
						n=8	49.6	33.1	24.8	24.8	9.6	4.9
	Encode	ENCO	—	5	6	n=1	19.8	12.7	9.5	9.5	3.7	2.1
						n=8	84.0	56.7	42.0	42.0	16.3	2.1
	Bit Counter	BCNT	—	4	5		33.4	22.3	16.7	16.7	6.5	0.8
	Long-word Bit Counter	BCNT L	—	4	5		57.0	38.0	28.5	28.5	11.1	0.9
	Approximate Broken Line	APR	—	5	6	n=4	91.0	60.7	45.5	45.5	17.7	2.3
	Long-word Approximate Broken Line	APR L	—	5	6	n=4	238	158.7	119	119	46.3	3.5
	Float Approximate Broken Line	FAPR	—	5	6		—	221	166	166	64.5	20.1
	Extend Sign	SIGN	—	3	4		—	6.9	5.2	5.2	2.0	0.7
	Long-word Extend Sign	SIGN D	—	3	4		—	—	—	—	—	0.7
	Binary to Gray-code *	BTOG	—	4	5		—	—	—	—	—	1.0
	Long-word Binary to Gray-code *	BTOG L	—	4	5		—	—	—	—	—	1.0
	Gray-code to Binary *	GTOB	—	4	5		—	—	—	—	—	2.2
	Long-word Gray-code to Binary *	GTOB L	—	4	5		—	—	—	—	—	2.2
Refresh Instruction	Direct Refresh	DREF	—	3	4	X (input)	467.4	333.4	322.0	22.0	15.0	11.4
						Y (output)	469.6	333.6	322.0	30.7	20.9	20.4
Program Control Instructions	Direct Refresh	JMP	—	1	2		29.0	19.3	14.5	1.8	0.7	0.4
	Subroutine Call	CALL	—	1	2		16.8	11.2	8.4	4.4	1.72	1.0
	Subroutine Entry	SUB	1	—	—		—	—	—	—	—	—
	Subroutine Return	RET	1	—	—		5.6	3.7	2.8	2.6	1.02	0.6
	Interrupt	INTP	1	—	—		—	—	—	—	—	—
	Interrupt Return	IRET	1	—	—		—	—	—	—	—	—
	Disable Interrupt	DI	1	—	—		0.18	0.12	0.09	0.09	0.035	0.2
	Enable Interrupt	EI	1	—	—		0.18	0.12	0.09	0.09	0.035	0.2
	Activate Block	ACT	—	2	3		1.6	1.1	0.8	0.8	0.3	0.7
	Inactivate Block	INACT	—	2	3		1.6	1.1	0.8	0.8	0.3	0.7
	For Loop	FOR	4	—	—		12.4	8.3	6.2	5.76	2.24	1.3
	Next Loop	NEXT	2	—	—	When repeated	18.0	12.0	9.0	3.42	1.33	0.6
						When ended	13.8	9.2	6.9	4.68	1.82	0.7
	Break Loop	BRK	—	1	2		8.8	5.9	4.4	3.96	1.54	3.1
	Activate Sensor Control Block	CBACT	—	1	2		—	—	—	80	70	0.5
	Inactivate Sensor Control Block	CBINA	—	1	2		—	—	—	70	60	0.5
	Disable Sensor Control Block	CBD	1	—	—		—	—	—	0.93	0.36	0.7
	Enable Sensor Control Block	CBE	1	—	—		—	—	—	0.93	0.36	0.7
Special Module Instructions	Read	READ	—	5	6	n=1	244.4	184.6	172.4	25.3	14.1	7.1
						n=16	376.6	307.7	295.0	177	120	55.5
	Read Long-word	READ L	—	5	6	n=1	294.4	188.9	169.6	29.5	17.6	9.4
						n=16	423.5	355.4	345.8	180	164	81.6
	Write	WRITE	—	5	6	n=1	230.6	175.4	165.4	24.2	13.4	6.3
						n=16	364.5	300.5	280.6	129	118	43.8
	Write Long-word	WRITE L	—	5	6	n=1	234.6	178.9	169.0	27.7	16.9	6.8
						n=16	411.6	346.3	322.2	178	162	51.2
	High-speed Read	HRD	—	5	6	n=1	13.6	9.1	6.8	6.8	2.6	0.8
						n=8	37.4	24.9	18.7	18.7	7.3	1.7
	High-speed Read Long-word	HRD L	—	5	6	n=1	14.4	9.6	7.2	7.2	2.8	0.9
						n=4	26.8	17.9	13.4	13.4	5.2	1.4

*: F3SP7□-□S only

■ Ladder Sequence Application Instructions (5/5)

Classifi-cation	Instruction	Mnemonic	Step Count			Condition	Instruction Execution Time (µs)					
			Always-execute	Execute-while-ON	Differential		F3SP05 F3SP08 F3SP21	F3SP25	F3SP35	F3SP22 F3SP28 F3SP38	F3SP53 F3SP58 F3SP59 F3SP66 F3SP67	F3SP71 F3SP76
Special Module Instructions	High-speed Write	HWR	—	5	6	n=1	11.2	7.5	5.6	5.6	2.2	0.9
						n=8	37.6	25.1	18.8	18.8	7.3	1.7
	High-speed Write Long-word	HWR L	—	5	6	n=1	11.8	7.9	5.9	5.9	2.3	1.0
						n=4	24.8	16.5	12.4	12.4	4.8	1.5
String Manipulation Instructions	Convert String to Numeric	VAL	—	5	6		—	56.7	42.5	42.5	16.5	5.2
	Convert String to Long-word Numeric	VAL L	—	5	6		—	73.1	54.8	54.8	21.3	4.5
	Convert Numeric to String	STR	—	5	6		—	58.5	43.9	43.9	17.1	5.3
	Convert Long-word Numeric to String	STR L	—	5	6		—	106.8	80.1	80.1	31.	8.3
	String Chain	SCHN	—	5	6	1 + 1 char.	—	25.9	19.4	19.4	7.5	9.9
						1024 + 1024 char.	—	7413	5560	5560	2162	222.9
	String Move	SMOV	—	4	5	n=1	—	15.6	11.7	11.7	4.6	0.4
						n=2047	—	6820	5115	5115	1989	60.0
	String Length Count	SLEN	—	4	5	n=1	—	9.6	7.2	7.2	2.8	1.6
						n=2047	—	2416	1782	1782	693	44.2
	Compare String	SCMP	—	5	6	n=1	—	20.8	15.6	15.6	6.1	4.2
						n=2047	—	9580	7185	7185	2794	103.4
	String Middle	SMID	—	5	6	1 from 2 characters	—	23.9	17.9	17.9	7.0	4.8
						2046 from 2047 char.	—	5653	4240	4240	1649	65.7
	String Left	SLFT	—	5	6	1 from 2 char.	—	22.4	16.8	16.8	6.5	4.3
						2046 from 2047 characters	—	4933	3700	3700	1439	347.2
	String Right	SRIT	—	5	6	1 from 2 char.	—	23.9	17.9	17.9	7.0	4.1
						2046 from 2047 char.	—	5657	4243	4243	1650	393.4
	String Search	SIST	—	5	6	1 from 2 char.	—	25.2	18.9	18.9	7.4	5.0
						1024 from 2047 char.	—	6507	4880	4880	1898	347.8
Structure and Macro Instructions	Structure Pointer Declaration	STRCT	3	—	—		—	—	—	1.0	0.3	0.0075
	Structure Move	STMOV	—	28	29	1 register	—	—	—	4.4	1.7	5.4
						2048 registers	—	—	—	563	221	73.7
	Macro Call	MCALL	—	5	6		—	22.0	16.5	16.5	6.4	2.7
	Parameter	PARA	—	4	5		—	8.0	6.0	6.0	2.3	0.8
	Macro Return	MRET	1	—	—		—	10.0	7.5	7.5	2.9	1.5
	Input Macro Instruction Call	NCALL	5	—	—		—	—	—	10.0	4.0	3.5
	Output of Input Macro	NMOUT	—	2	3		—	—	—	2.4	1.0	0.5
Indirect Specification Instructions	Indirect Address Set	SET@	—	3	4		—	—	—	1.1	0.5	0.6
	Indirect Address Add	ADD@	—	3	4		—	—	—	2.4	1.0	1.3
	Indirect Address Move	MOV@	—	3	4		—	—	—	0.8	0.4	0.6
Miscellaneous Instructions	Refresh Watchdog Timer	WDT	—	1	2		0.9	0.60	0.45	0.45	0.175	0.6
	Read Free Run Timer	FTIMR	—	2	3		—	—	—	0.66	0.25	0.5
	Start Elapsed Time Measurement	TMS L	—	2	3		—	—	—	—	—	0.6
	Elapsed Time Measurement	TME L	—	3	4		—	—	—	—	—	1.0
	Signal to BASIC	SIG	—	—	5		706.8	525.4	521	508	500	150
	Sampling Trace	TRC	–	1	2		—	16.8	12.6	12.6	4.9	0.8
	Save User Log	ULOG	–	4	5		18.2	12.1	9.1	9.1	3.5	2.1
	Read User Log	ULOGR	–	5	6		117.6	78.4	58.8	58.8	22.9	14.8
	Clear User Log	UCLR	–	2	3		4.2	2.8	2.1	2.1	0.8	0.3
	Set Date	DATE	–	–	3		—	—	—	0.3	0.2	2.0
	Set Time	TIME	–	–	3		—	—	—	0.3	0.2	1.8
	Set Date String	SDATE	–	–	3		—	—	–	0.3	0.2	3.7
	Set Time String	STIME	–	–	3		–	–	–	0.3	0.3	3.6

■ Ladder Sequence Continuous Type Application Instructions (1/4)

Classifi-cation	Instruction	Mnemonic	Step Count	Text Parameter Count	Status Word Count	Foreground Instruction Execution Time (μs)		
						Condition	F3SP66, F3SP67	F3SP71 F3SP76
Disk Operation	Mount Memory Card	MOUNT	6	0	1	At start At end In ON state	50 30 20	50 30 20
	Unmount Memory Card	UNMOUNT	6	0	1	At start At end In ON state	50 20 20	50 30 20
	Format Disk	FORMAT	5	0	1	At start At end In ON state	50 30 20	50 30 20
	Disk Info	DISKINFO	6	0	1	At start At end In ON state	60 30 20	50 30 20
File Access	Open File	FOPEN	6	1	1	At start At end In ON state	70 30 20	70 30 20
	Close File	FCLOSE	6	0	1	At start At end In ON state	50 30 20	60 30 20
	Read File Line	FGETS	6	0	3	At start At end In ON state	60 30 20	70 30 20
	Write File Line	FPUTS	5	1	2	At start At end In ON state	70 30 20	70 30 20
	Read File Block	FREAD	6	0	3	At start At end In ON state	70 30 20	70 30 20
	Write File Block	FWRITE	6	0	3	At start At end In ON state	70 30 20	70 30 20
	File Seek	FSEEK	5	0	3	At start At end In ON state	60 30 20	70 30 20
	File Text Search	FSEARCHT	5	1	3	At start At end In ON state	60 30 20	70 30 20
	File Binary Search	FSEARCHB	6	0	3	At start At end In ON state	60 30 20	70 30 20
	Convert CSV File to Device	F2DCSV	6	0	5	At start At end In ON state	70 30 20	70 30 20
	Convert Device to CSV File	D2FCSV	6	0	5	At start At end In ON state	70 30 20	70 30 20
	Convert Binary File to Device	F2DBIN	6	0	5	At start At end In ON state	70 30 20	70 30 20
	Convert Device to Binary File	D2FBIN	6	0	5	At start At end In ON state	60 30 20	70 30 20

Note: The foreground instruction execution time is the estimated time consumed per scan cycle by a continuous type application instruction.
"At start" refers to a scan cycle during which the input condition changes from OFF to ON and instruction execution begins.
"At end" refers to a scan cycle during which background instruction execution processing completes and the result signal is held to ON.
"In ON state" refers to a scan cycle during which the input condition stays ON but excluding the above two cases.
"In OFF state" refers to a scan cycle during which the input condition stays OFF. The instruction execution time is 0.7 μs (the same for all continuous type application instructions.)

■ Ladder Sequence Continuous Type Application Instructions (2/4)

Classifi-cation	Instruction	Mnemonic	Step Count	Text Parameter Count	Status Word Count	Foreground Instruction Execution Time (μs)		
						Condition	F3SP66, F3SP67	F3SP71 F3SP76
File Operation	Copy File	FCOPY	6	2	1	At start At end In ON state	90 40 20	80 30 20
	Move File	FMOVE	6	2	1	At start At end In ON state	90 40 20	80 30 20
	Delete File	FDEL	6	1	1	At start At end In ON state	80 30 20	70 30 20
	Make Directory	FMKDIR	5	1	1	At start At end In ON state	70 30 20	70 30 20
	Remove Directory	FRMDIR	6	1	1	At start At end In ON state	70 30 20	70 30 20
	Rename File	FREN	5	2	1	At start At end In ON state	80 30 20	80 30 20
	File Status	FSTAT	6	1	1	At start At end In ON state	60 30 20	60 30 20
	File List Start	FLSFIRST	5	1	1	At start At end In ON state	60 30 20	60 30 20
	File List Next	FLS	6	0	1	At start At end In ON state	50 30 20	50 30 20
	File List End	FLSFIN	5	0	1	At start At end In ON state	40 30 20	50 30 20
	Change Directory	FCD	5	1	1	At start At end In ON state	70 30 20	70 30 20
	Concatenate File	FCAT	5	2	1	At start At end In ON state	90 40 20	80 30 20
	Change File Attribute	FATRW	5	1	1	At start At end In ON state	80 30 20	70 30 20
UDP/IP	UDP/IP Open	UDPOPEN	6	0	1	At start At end In ON state	50 30 20	60 30 20
	UDP/IP Close	UDPCLOSE	6	0	1	At start At end In ON state	60 40 30	60 30 30
	UDP/IP Send Request	UDPSND	6	0	1	At start At end In ON state	60 30 40	60 30 40
	UDP/IP Receive Request	UDPRCV	6	0	5	At start At end In ON state	70 20 20	60 30 20

Note: The foreground instruction execution time is the estimated time consumed per scan cycle by a continuous type application instruction.
"At start" refers to a scan cycle during which the input condition changes from OFF to ON and instruction execution begins.
"At end" refers to a scan cycle during which background instruction execution processing completes and the result signal is held to ON.
"In ON state" refers to a scan cycle during which the input condition stays ON but excluding the above two cases.
"In OFF state" refers to a scan cycle during which the input condition stays OFF. The instruction execution time is 0.7 μs (the same for all continuous type application instructions.)

■ Ladder Sequence Continuous Type Application Instructions (3/4)

Classification	Instruction	Mnemonic	Step Count	Text Parameter Count	Status Word Count	Foreground Instruction Execution Time (μs)		
						Condition	F3SP66, F3SP67	F3SP71 F3SP76
TCP/IP	TCP/IP Open	TCPOPEN	5	0	1	At start At end In ON state	50 30 20	60 30 20
	TCP/IP Close	TCPCLOSE	6	0	1	At start At end In ON state	50 30 20	60 30 20
	TCP/IP Connect Request	TCPCNCT	5	0	1	At start At end In ON state	60 30 20	60 30 20
	TCP/IP Listen Request	TCPLISN	5	0	6	At start At end In ON state	60 30 10	60 30 20
	TCP/IP Send Request	TCPSND	6	0	1	At start At end In ON state	60 20 20	60 30 20
	TCP/IP Receive Request	TCPRCV	6	0	1	At start At end In ON state	60 30 20	60 30 20
	Socket Option *	SOCKOPT	6	0	1	At start At end In ON state	— — —	60 30 20
FTP Server	FTP Server Run Request Service	FTPSRUN	5	0	1	At start At end In ON state	40 20 20	50 30 20
	FTP Server Stop Request Service	FTPSSTOP	5	0	1	At start At end In ON state	40 20 20	50 30 20
Properties	Write CPU Properties	PWRITE	6	1	1	At start At end In ON state	— — 20	— — 20
	Read CPU Properties	PREAD	6	1	1	At start At end In ON state	— — 20	— — 20

*: F3SP7□-□S only

Note: The foreground instruction execution time is the estimated time consumed per scan cycle by a continuous type application instruction.
"Start cycle" refers to a scan cycle during which the input condition changes from OFF to ON and instruction execution begins.
"At end" refers to a scan cycle during which background instruction execution processing completes and the result signal is held to ON.
"In ON state" refers to a scan cycle during which the input condition stays ON, but excluding the above two cases.
"In OFF state" refers to a scan cycle during which the input condition stays OFF. The instruction execution time is 0.7 μs (the same for all continuous type application instructions.)

■ Ladder Sequence Continuous Type Application Instructions (4/4)

Classification	Instruction	Mnemonic	Step Count	Text Parameter Count	Status Word Count	Foreground Instruction Execution Time (μs)		
						Condition	F3SP66, F3SP67	F3SP71 F3SP76
FTP Client	FTP Client Open	FTPOPEN	6	0	1	At start	60	50
						At end	20	30
						In ON state	20	20
	FTP Client Quit	FTPQUIT	5	0	1	At start	60	50
						At end	20	30
						In ON state	20	20
	FTP Client Put File	FTPPUT	5	2	1	At start	90	70
						At end	20	30
						In ON state	20	20
	FTP Client Put Unique File	FTPPUTU	6	2	18	At start	90	70
						At end	30	30
						In ON state	20	20
	FTP Client Append File	FTPAPEND	5	2	1	At start	90	70
						At end	20	30
						In ON state	20	20
	FTP Client Get File	FTPGET	5	2	1	At start	90	70
						At end	20	30
						In ON state	20	20
	FTP Client Change Directory	FTPCD	5	1	1	At start	70	60
						At end	20	30
						In ON state	20	20
	FTP Client Change Local Directory	FTPLCD	5	1	1	At start	70	60
						At end	20	30
						In ON state	20	20
	FTP Client Current Directory Info	FTPPWD	6	0	1	At start	60	50
						At end	20	30
						In ON state	20	20
	FTP Client Get File List	FTPLS	5	3	1	At start	100	100
						At end	20	30
						In ON state	20	20
	FTP Client Delete File	FTPDEL	5	1	1	At start	70	60
						At end	20	30
						In ON state	20	20
	FTP Client Rename File	FTPREN	5	2	1	At start	90	70
						At end	20	30
						In ON state	20	20
	FTP Client Make Directory	FTPMKDIR	5	1	1	At start	70	60
						At end	20	30
						In ON state	20	20
	FTP Client Remove Directory	FTPRMDIR	5	1	1	At start	70	60
						At end	20	30
						In ON state	20	20
	FTP Client Representation Type	FTPTYPE	6	0	1	At start	60	50
						At end	20	30
						In ON state	20	20

Note: The foreground instruction execution time is the estimated time consumed per scan cycle by a continuous type application instruction.
"At start" refers to a scan cycle during which the input condition changes from OFF to ON and instruction execution begins.
"At end" refers to a scan cycle during which background instruction execution processing completes and the result signal is held to ON.
"In ON state" refers to a scan cycle during which the input condition stays ON, but excluding the above two cases.
"In OFF state" refers to a scan cycle during which the input condition stays OFF. The instruction execution time is 0.7 μs (the same for all continuous type application instructions.)

Blank Page

FA-M3
Sequence CPU Instruction Manual - Instructions

IM 34M06P12-03E 5th Edition

Index

Revision Information

Document Name : Sequence CPU Instruction Manual – Instructions
Document No. : IM 34M06P12-03E

Edition	Date	Revised Item
1st	Sep. 1995	New publication
2nd	May. 2000	Added information for F3SP28, F3SP38, F3SP53, F3SP58
3rd	Oct. 2002	Added information for F3SP59, -□S Reorganized appendix, errata, overall revision
4th	June 2007	Added information for F3SP66-4S, F3SP67-6S
5th	Jan. 2012	Added information for F3SP71-4N, F3SP76-7N, F3SP71-4S, F3SP76-7N, F3SP22-0S and WideFields3 R1.01, errata

Written by PLC Group
International Sales Promotion Dept.
IA Systems Business Headquarters
Yokogawa Electric Corporation
Published by Yokogawa Electric Corporation
2-9-32 Nakacho, Musashino-shi, Tokyo, 180-8750, JAPAN
Printed by Kohoku Publishing & Printing Inc.

Blank Page