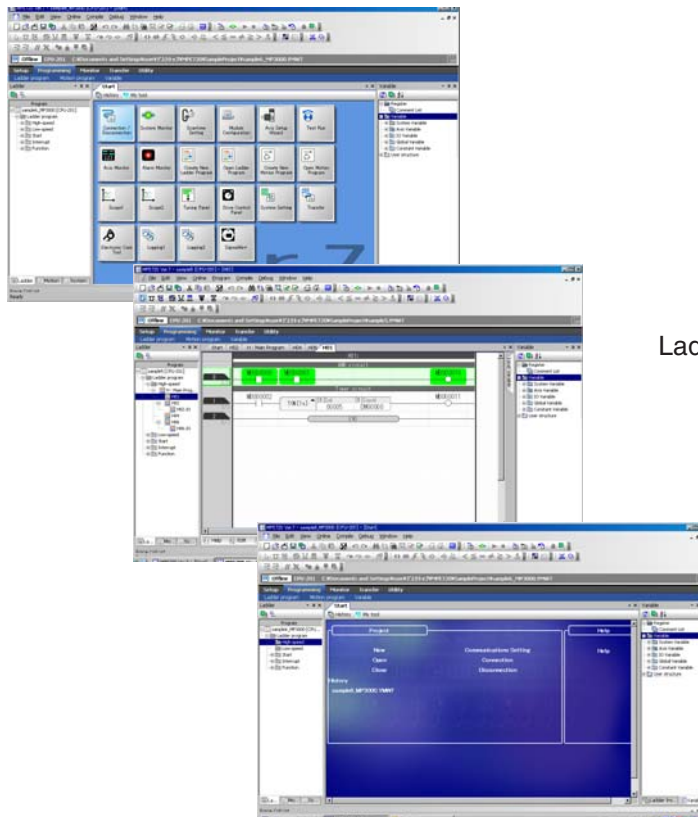


Machine Controller MP3000 Series Ladder Program PROGRAMMING MANUAL



Features and Overview
of Ladder Programs

1

Ladder Program Development Flow

2

Registers

3

Ladder Language
Instructions

4

Features of the MPE720
Engineering Tool

5

System Service
Registers

AppA

Sample Programs

AppB

Format for EXPRESSION
Instructions

AppC

Precautions on Motion
Parameters

AppD

Machine Controller Specifications

AppE

Error Codes

AppF

Copyright © 2012 YASKAWA ELECTRIC CORPORATION

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, mechanical, electronic, photocopying, recording, or otherwise, without the prior written permission of Yaskawa. No patent liability is assumed with respect to the use of the information contained herein. Moreover, because Yaskawa is constantly striving to improve its high-quality products, the information contained in this manual is subject to change without notice. Every precaution has been taken in the preparation of this manual. Nevertheless, Yaskawa assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained in this publication.

About this Manual

This manual provides information on ladder programming for MP3000-series Machine Controllers. Read this manual carefully to ensure the correct usage of the Machine Controller and apply the Machine Controller to control your manufacturing system. Keep this manual in a safe place so that it can be referred to whenever necessary.

Using this Manual

◆ Basic Terms

Unless otherwise specified, the following definitions are used:

Basic Terms	Meaning
Machine Controller	MP3000-series Machine Controller
MPE720	The Engineering Tool or a personal computer running the Engineering Tool
PLC	A Programmable Logic Controller
MP3200	A generic name for the Power Supply Unit, CPU Unit, Base Unit, and Rack Expansion Interface Unit
MP3300	A generic name for the CPU Module and Base Unit.
MP3100	CPU Module

◆ MPE720 Engineering Tool Version Number

In this manual, the operation of MPE720 is described using screen captures of MPE720 version 7.

◆ Indication of Reverse Signals

In this manual, the names of reverse signals (ones that are valid when low) are written with a forward slash (/) before the signal name, as shown in the following example:

Notation Examples

- $\overline{S\text{-ON}}$ = /S-ON
- $\overline{P\text{-CON}}$ = /P-CON

◆ The Meaning of “Torque” in This Manual

Although the term “torque” is commonly used when describing rotary Servomotors and “force” is used when describing linear Servomotors, this manual uses “torque” when describing either one (excluding parameter names).

◆ Copyrights

- MECHATROLINK is a trademark of the MECHATROLINK Members Association.
- Ethernet is a registered trademark of the Xerox Corporation.
- Other product names and company names are the trademarks or registered trademarks of the respective company. “TM” and the ® mark do not appear with product or company names in this manual.

◆ Visual Aids

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



Indicates precautions or restrictions that must be observed.

Indicates alarm displays and other precautions that will not result in machine damage.

Example Indicates operating or setting examples.

Information Indicates supplemental information to deepen understanding or useful information.



Indicates definitions of difficult terms or terms that have not been previously explained in this manual.

Related Manuals

The following table lists the manuals that are related to the MP3000-series Machine Controllers. Refer to these manuals as required.

Be aware of all product specifications and restrictions to product application before you attempt to use any product.

Category	Manual Name	Manual Number	Contents
Basic functionality	Machine Controller MP3000 Series Machine Controller System Setup Manual	SIEP C880725 00	Describes the functions of the MP3000-series Machine Controllers and the procedures that are required to use the Machine Controller, from installation and connections to settings, programming, trial operation, and debugging.
	Machine Controller MP3000 Series Machine Controller System Troubleshooting Manual	SIEP C880725 01	Describes troubleshooting an MP3000-series Machine Controller.
	Machine Controller MP3000 Series MP3100 Product Manual	SIEP C880725 24	Describes the specifications and system configuration of an MP3000-series MP3100 Machine Controller and the functions of the CPU.
	Machine Controller MP3000 Series MP3200 Product Manual	SIEP C880725 10	Describes the specifications and system configuration of an MP3000-series MP3200 Machine Controller and the functions of the CPU Unit.
	Machine Controller MP3000 Series MP3300 Product Manual	SIEP C880725 21	Describes the specifications and system configuration of an MP3000-series MP3300 Machine Controller and the functions of the CPU Module.
Communications functionality	Machine Controller MP3000 Series Communications User's Manual	SIEP C880725 12	Describes the specifications, system configuration, and communications connection methods for the Ethernet communications that are used with an MP3000-series Machine Controller.
Motion control functionality	Machine Controller MP3000 Series Motion Control User's Manual	SIEP C880725 11	Describes the specifications, system configuration, and operating methods for the SVC/SVR or SVC32/SVR32 Motion Function Modules that are used in an MP3000-series Machine Controller.
Programming	Machine Controller MP3000 Series Motion Program Programming Manual	SIEP C880725 14	Describes the motion programming and sequence programming specifications and instructions of MP3000-series Machine Controller.
Engineering Tool	Machine Controller MP2000/MP3000 Series Engineering Tool MPE720 Version 7 User's Manual	SIEP C880761 03	Describes how to operate MPE720 version 7.

Safety Precautions

◆ Safety Information

To prevent personal injury and equipment damage in advance, the following signal words are used to indicate safety precautions in this document. The signal words are used to classify the hazards and the degree of damage or injury that may occur if a product is used incorrectly. Information marked as shown below is important for safety. Always read this information and heed the precautions that are provided.



DANGER

- Indicates precautions that, if not heeded, are likely to result in loss of life, serious injury, or fire.



WARNING

- Indicates precautions that, if not heeded, could result in loss of life, serious injury, or fire.



CAUTION

- Indicates precautions that, if not heeded, could result in relatively serious or minor injury, or in fire.

NOTICE

- Indicates precautions that, if not heeded, could result in property damage.

◆ General Precautions



WARNING

- The installation must be suitable and it must be performed only by an experienced technician. There is a risk of electrical shock or injury.
- Before connecting the machine and starting operation, make sure that an emergency stop procedure has been provided and is working correctly. There is a risk of injury.
- Do not approach the machine after a momentary interruption to the power supply. When power is restored, the product and the device connected to it may start operation suddenly. Provide safety measures in advance to ensure human safety when operation restarts. There is a risk of injury.
- Do not touch anything inside the product. There is a risk of electrical shock.
- Do not remove the front cover, cables, connector, or options while power is being supplied. There is a risk of electrical shock, malfunction, or damage.
- Do not damage, pull on, apply excessive force to, place heavy objects on, or pinch the cables. There is a risk of electrical shock, operational failure of the product, or burning.
- Do not attempt to modify the product in any way. There is a risk of injury or device damage.

◆ Storage and Transportation Precautions



CAUTION

- Hold onto the main body of the product when transporting it. Holding the cables or connectors may damage them or result in injury.
- Do not overload the product during transportation. (Follow all instructions.) There is a risk of injury or an accident.
- Never subject the product to an atmosphere containing halogen (fluorine, chlorine, bromine, or iodine) during transportation. There is a risk of malfunction or damage.
- If disinfectants or insecticides must be used to treat packing materials such as wooden frames, pallets, or plywood, the packing materials must be treated before the product is packaged, and methods other than fumigation must be used.
Example: Heat treatment, where materials are kiln-dried to a core temperature of 56°C for 30 minutes or more.
If the electronic products, which include stand-alone products and products installed in machines, are packed with fumigated wooden materials, the electrical components may be greatly damaged by the gases or fumes resulting from the fumigation process. In particular, disinfectants containing halogen, which includes chlorine, fluorine, bromine, or iodine can contribute to the erosion of the capacitors.

NOTICE

- Do not install the product in any of the following locations.
 - Locations that are subject to direct sunlight
 - Locations that are subject to ambient temperatures that exceed product specifications
 - Locations that are subject to relative humidities that exceed product specifications
 - Locations that are subject to condensation as the result of extreme changes in temperature
 - Locations that are subject to corrosive or flammable gases
 - Locations that are near flammable materials
 - Locations that are subject to dust, salts, or iron powder
 - Locations that are subject to water, oil, or chemicals
 - Locations that are subject to vibration or shock that exceeds product specificationsIf you store the product in any of the above locations, the product may fail or be damaged.

◆ Installation Precautions



CAUTION

- **Never install the product in an atmosphere containing halogen (fluorine, chlorine, bromine, or iodine).**
There is a risk of malfunction or damage.
- **Do not step on the product or place heavy objects on the product.**
There is a risk of injury or an accident.
- **Do not block the air exhaust ports on the product. Do not allow foreign objects to enter the product.**
There is a risk of internal element deterioration, malfunction, or fire.
- **Always mount the product in the specified orientation.**
There is a risk of malfunction.
- **Leave the specified amount of space between the product, and the interior surface of the control panel and other devices.**
There is a risk of fire or malfunction.
- **Do not subject the product to strong shock.**
There is a risk of malfunction.
- **Suitable battery installation must be performed and it must be performed only by an experienced technician.**
There is a risk of electrical shock, injury, or device damage.
- **Do not touch the electrodes when installing the Battery.**
Static electricity may damage the electrodes.

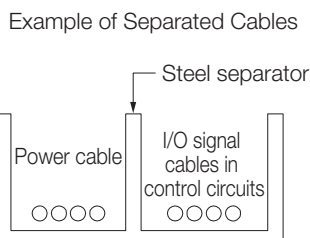
NOTICE

- **Do not install the product in any of the following locations.**
 - Locations that are subject to direct sunlight
 - Locations that are subject to ambient temperatures that exceed product specifications
 - Locations that are subject to relative humidities that exceed product specifications
 - Locations that are subject to condensation as the result of extreme changes in temperature
 - Locations that are subject to corrosive or flammable gases
 - Locations that are near flammable materials
 - Locations that are subject to dust, salts, or iron powder
 - Locations that are subject to water, oil, or chemicals
 - Locations that are subject to vibration or shock that exceeds product specifications
 - Locations near devices that generate strong magnetic fields
 - Locations that are subject to radiation
- If you install the product in any of the above locations, the product may fail or be damaged.

◆ Wiring Precautions

⚠ CAUTION

- **Do not change any wiring while power is being supplied.**
There is a risk of electrical shock, injury, or device damage.
- **Check the wiring to be sure it has been performed correctly.**
There is a risk of motor run-away, injury, or accidents.
- **Always use a power supply of the specified voltage.**
There is a risk of fire or accident.
- **In places with poor power supply conditions, ensure that the input power is supplied within the specified voltage range.**
There is a risk of device damage.
- **Install breakers and other safety measures to provide protection against shorts in external wiring.**
There is a risk of fire.
- **Provide sufficient shielding when using the product in the following locations.**
 - Locations that are subject to noise, such as from static electricity
 - Locations that are subject to strong electromagnetic or magnetic fields
 - Locations that are subject to radiation
 - Locations that are near power linesThere is a risk of device damage.
- **Configure the circuits to turn ON the power supply to the CPU Unit/CPU Module before the 24-V I/O power supply.**
If the power supply to the CPU Unit/CPU Module is turned ON after the external power supply, e.g., the 24-V I/O power supply, the outputs from the CPU Unit/CPU Module may momentarily turn ON when the power supply to the CPU Unit/CPU Module turns ON. This can result in unexpected operation that may cause injury or device damage.
- **Provide emergency stop circuits, interlock circuits, limit circuits, and any other required safety measures in control circuits outside of the product.**
There is a risk of injury or device damage.
- **If you use MECHATROLINK I/O Modules, use the establishment of MECHATROLINK communications as an interlock output condition.**
There is a risk of device damage.
- **Connect the Battery with the correct polarity.**
There is a risk of battery damage or explosion.
- **Select the I/O signal wires for external wiring to connect the product to external devices based on the following criteria:**
 - Mechanical strength
 - Noise interference
 - Wiring distance
 - Signal voltage
- **Separate the I/O signal cables for control circuits from the power cables both inside and outside the control panel to reduce the influence of noise from the power cables.**
If the I/O signal lines and power lines are not separated properly, malfunction may occur.



◆ Operation Precautions

CAUTION

- Follow the procedures and instructions in the user's manuals for the relevant product to perform normal operation and trial operation.
Operating mistakes while the Servomotor and machine are connected may damage the machine or even cause accidents resulting in injury or death.
- Implement interlock signals and other safety circuits external to the product to ensure safety in the overall system even if the following conditions occur.
 - Product failure or errors caused by external factors
 - Shutdown of operation due to product detection of an error in self-diagnosis and the subsequent turning OFF or holding of output signals
 - Holding of the ON or OFF status of outputs from the product due to fusing or burning of output relays or damage to output transistors
 - Voltage drops from overloads or short-circuits in the 24-V output from the product and the subsequent inability to output signals
 - Unexpected outputs due to errors in the power supply, I/O, or memory that cannot be detected by the product through self-diagnosis.There is a risk of injury, device damage, or burning.
- Observe the setting methods that are given in the manual of the Motion Control Function Modules to be used for the following parameters.
 - Parameters for absolute position detection when the axis type is set to a finite-length axis
 - Parameters for simple absolute infinite-length position control when the axis type is set to an infinite length axisIf any other methods are used, offset in the current position when the power supply is turned OFF and ON again may result in device damage.
- OL□□□48 (Zero Point Position Offset in Machine Coordinate System) is always valid when the axis type is set to a finite-length axis. Do not change the setting of OL□□□48 while the Machine Controller is operating.
There is a risk of machine damage or an accident.

◆ Maintenance and Inspection Precautions

CAUTION

- Do not attempt to disassemble or repair the product.
There is a risk of electrical shock, injury, or device damage.
- Do not change any wiring while power is being supplied.
There is a risk of electrical shock, injury, or device damage.
- Suitable battery installation must be performed and it must be performed only by an experienced technician.
There is a risk of electrical shock, injury, or device damage.
- Replace the Battery only while power is supplied to the product.
Replacing the Battery while the power supply to the product is turned OFF may result in loss of the data stored in memory in the product.
- Do not touch the electrodes when installing the Battery.
Static electricity may damage the electrodes.
- Do not forget to perform the following tasks when you replace the CPU Unit/CPU Module:
 - Back up all programs and parameters from the CPU Unit/CPU Module that is being replaced.
 - Transfer all saved programs and parameters to the new CPU Unit/CPU Module.If you operate the CPU Unit/CPU Module without transferring this data, unexpected operation may occur. There is a risk of injury or device damage.
- Do not touch the heat sink on the CPU Unit/CPU Module while the power supply is turned ON or for a sufficient period of time after the power supply is turned OFF.
The heat sink may be very hot, and there is a risk of burn injury.

◆ Disposal Precautions



CAUTION

- Correctly discard the product and used batteries as stipulated by regional, local, and municipal laws and regulations. Be sure to include these contents in all labelling and warning notifications on the final product as necessary.



◆ General Precautions

Observe the following general precautions to ensure safe application.

- Figures provided in this document are typical examples or conceptual representations. There may be differences between them and actual wiring, circuits, and products.
- The products shown in illustrations in this document are sometimes shown without covers or protective guards. Always replace all covers and protective guards before you use the product.
- If you need a new copy of this document because it has been lost or damaged, contact your nearest Yaskawa representative or one of the offices listed on the back of this document.
- This document is subject to change without notice for product improvements, specifications changes, and improvements to the manual itself.
We will update the document number of the document and issue revisions when changes are made.
- Any and all quality guarantees provided by Yaskawa are null and void if the customer modifies the product in any way. Yaskawa disavows any responsibility for damages or losses that are caused by modified products.

Warranty

◆ Details of Warranty

■ Warranty Period

The warranty period for a product that was purchased (hereinafter called “delivered product”) is one year from the time of delivery to the location specified by the customer or 18 months from the time of shipment from the Yaskawa factory, whichever is sooner.

■ Warranty Scope

Yaskawa shall replace or repair a defective product free of charge if a defect attributable to Yaskawa occurs during the warranty period above. This warranty does not cover defects caused by the delivered product reaching the end of its service life and replacement of parts that require replacement or that have a limited service life.

This warranty does not cover failures that result from any of the following causes.

- Improper handling, abuse, or use in unsuitable conditions or in environments not described in product catalogs or manuals, or in any separately agreed-upon specifications
- Causes not attributable to the delivered product itself
- Modifications or repairs not performed by Yaskawa
- Abuse of the delivered product in a manner in which it was not originally intended
- Causes that were not foreseeable with the scientific and technological understanding at the time of shipment from Yaskawa
- Events for which Yaskawa is not responsible, such as natural or human-made disasters

◆ Limitations of Liability

- Yaskawa shall in no event be responsible for any damage or loss of opportunity to the customer that arises due to failure of the delivered product.
- Yaskawa shall not be responsible for any programs (including parameter settings) or the results of program execution of the programs provided by the user or by a third party for use with programmable Yaskawa products.
- The information described in product catalogs or manuals is provided for the purpose of the customer purchasing the appropriate product for the intended application. The use thereof does not guarantee that there are no infringements of intellectual property rights or other proprietary rights of Yaskawa or third parties, nor does it construe a license.
- Yaskawa shall not be responsible for any damage arising from infringements of intellectual property rights or other proprietary rights of third parties as a result of using the information described in catalogs or manuals.

◆ Suitability for Use

- It is the customer's responsibility to confirm conformity with any standards, codes, or regulations that apply if the Yaskawa product is used in combination with any other products.
- The customer must confirm that the Yaskawa product is suitable for the systems, machines, and equipment used by the customer.
- Consult with Yaskawa to determine whether use in the following applications is acceptable. If use in the application is acceptable, use the product with extra allowance in ratings and specifications, and provide safety measures to minimize hazards in the event of failure.
 - Outdoor use, use involving potential chemical contamination or electrical interference, or use in conditions or environments not described in product catalogs or manuals
 - Nuclear energy control systems, combustion systems, railroad systems, aviation systems, vehicle systems, medical equipment, amusement machines, and installations subject to separate industry or government regulations
 - Systems, machines, and equipment that may present a risk to life or property
 - Systems that require a high degree of reliability, such as systems that supply gas, water, or electricity, or systems that operate continuously 24 hours a day
 - Other systems that require a similar high degree of safety
- Never use the product for an application involving serious risk to life or property without first ensuring that the system is designed to secure the required level of safety with risk warnings and redundancy, and that the Yaskawa product is properly rated and installed.
- The circuit examples and other application examples described in product catalogs and manuals are for reference. Check the functionality and safety of the actual devices and equipment to be used before using the product.
- Read and understand all use prohibitions and precautions, and operate the Yaskawa product correctly to prevent accidental harm to third parties.

◆ Specifications Change

The names, specifications, appearance, and accessories of products in product catalogs and manuals may be changed at any time based on improvements and other reasons. The next editions of the revised catalogs or manuals will be published with updated code numbers. Consult with your Yaskawa representative to confirm the actual specifications before purchasing a product.

Contents

About this Manual	iii
Using this Manual	iii
Related Manuals	v
Safety Precautions	vi
Warranty	xii

1

Features and Overview of Ladder Programs

1.1	What Is a Ladder Program?	1-2
1.2	Features	1-3
1.2.1	The Various Execution Timing of Ladder Drawings	1-3
1.2.2	Program Modules	1-4
1.2.3	Programming Complicated Numeric Operations	1-4
1.2.4	Communications Control with External Devices	1-5
1.2.5	Complete Synchronization with Motion Control	1-5
1.3	Introduction	1-6
1.3.1	Ladder Program Editor	1-6
1.3.2	Ladder Drawings	1-7
1.3.3	User Functions	1-13
1.3.4	Table Data	1-18

2

Ladder Program Development Flow

2.1	Introduction	2-2
2.2	Preparation for Devices to be Connected	2-3
2.2.1	Connecting the Hardware	2-3
2.2.2	Installing MPE720 Version 7	2-3
2.3	Creating a Project	2-4
2.4	Self Configuration	2-5
2.5	Going Online	2-6
2.6	Creating Ladder Programs	2-7
2.7	Writing the Ladder Programs	2-11
2.8	Checking the Operation of the Ladder Programs	2-13
2.8.1	Preparations for Checking Operation	2-13
2.8.2	Confirming the Operation of the 0000th Line (AND Circuit)	2-14
2.8.3	Confirming the Operation of the 0001st Line (Timer Circuit)	2-15
2.9	Save the Ladder Program to Flash Memory	2-16

3

Registers

3.1	Global Registers	3-2
3.2	Local Registers.	3-4
3.2.1	Precautions When Using Local Registers within a User Function	3-5
3.2.2	Setting the D Register Clear When Start Option	3-6
3.2.3	Setting for D Registers	3-7
3.3	Structure of Register Addresses	3-8
3.3.1	Register Types	3-8
3.3.2	Data Types	3-8
3.4	Index Registers (i, j)	3-12
3.5	Array Registers ([]).	3-14

4

Ladder Language Instructions

4.1	Introduction	4-6
4.1.1	Ladder Language Instructions	4-6
4.1.2	How to Read the Ladder Language Instructions	4-10
4.2	Relay Circuit Instructions	4-11
4.2.1	NO Contact (NOC)	4-11
4.2.2	Rising-edge NO Contact (ONP-NOC)	4-12
4.2.3	Falling-edge NO Contact (OFFP-NOC)	4-13
4.2.4	NC Contact (NCC)	4-14
4.2.5	Rising-edge NC Contact (ONP-NCC)	4-14
4.2.6	Falling-edge NC Contact (OFFP-NCC)	4-15
4.2.7	1-ms ON-Delay Timer (TON(1ms))	4-16
4.2.8	1-ms OFF-Delay Timer (TOFF(1 ms))	4-18
4.2.9	10-ms ON-Delay Timer (TON(10ms))	4-19
4.2.10	10-ms OFF-Delay Timer (TOFF(10ms))	4-21
4.2.11	1-s ON-Delay Timer (TON(1s))	4-22
4.2.12	1-s OFF-Delay Timer (TOFF(1s))	4-24
4.2.13	Rising-edge Pulses (ON-PLS)	4-25
4.2.14	Falling-edge Pulses (OFF-PLS)	4-27
4.2.15	Coil (COIL)	4-29
4.2.16	Reverse Coil (REV-COIL)	4-30
4.2.17	Rising-edge Detection Coil (ONP-COIL)	4-31
4.2.18	Falling-edge Detection Coil (OFFP-COIL)	4-31
4.2.19	Set Coil (S-COIL)	4-32
4.2.20	Reset Coil (R-COIL)	4-33
4.3	Numeric Operation Instructions	4-34
4.3.1	Store (STORE)	4-34
4.3.2	Add (ADD (+))	4-35
4.3.3	Extended Add (ADDX (++))	4-36
4.3.4	Subtract (SUB (-))	4-38
4.3.5	Extended Subtract (SUBX (- -))	4-39
4.3.6	Multiply (MUL (x))	4-41
4.3.7	Divide (DIV (÷))	4-42
4.3.8	Integer Remainder (MOD)	4-43
4.3.9	Real Remainder (REM)	4-45

4.3.10	Increment (INC)	4-46
4.3.11	Decrement (DEC)	4-47
4.3.12	Add Time (TMADD)	4-48
4.3.13	Subtract Time (TMSUB)	4-50
4.3.14	Spend Time (SPEND)	4-52
4.3.15	Invert Sign (INV)	4-54
4.3.16	One's Complement (COM)	4-55
4.3.17	Absolute Value (ABS)	4-56
4.3.18	Binary Conversion (BIN)	4-57
4.3.19	BCD Conversion (BCD)	4-58
4.3.20	Parity Conversion (PARITY)	4-59
4.3.21	ASCII Conversion 1 (ASCII)	4-60
4.3.22	ASCII Conversion 2 (BINASC)	4-61
4.3.23	ASCII Conversion 3 (ASCBIN)	4-62

4.4 Logic Operations and Comparison Instructions 4-64

4.4.1	Inclusive AND (AND)	4-64
4.4.2	Inclusive OR (OR)	4-65
4.4.3	Exclusive OR (XOR)	4-66
4.4.4	Less Than (<)	4-67
4.4.5	Less Than or Equal (\leq)	4-68
4.4.6	Equal (=)	4-69
4.4.7	Not Equal (\neq)	4-70
4.4.8	Greater Than or Equal (\geq)	4-71
4.4.9	Greater Than (>)	4-72
4.4.10	Range Check (RCHK)	4-73

4.5 Program Control Instructions 4-75

4.5.1	Call Sequence Program (SEE)	4-75
4.5.2	Call Motion Program (MSEE)	4-76
4.5.3	Call User Function (FUNC)	4-78
4.5.4	Direct Input String (INS)	4-79
4.5.5	Direct Output String (OUTS)	4-81
4.5.6	Call Extended Program (XCALL)	4-83
4.5.7	WHILE Construct (WHILE, END_WHILE)	4-84
4.5.8	FOR Construct (FOR, END_FOR)	4-86
4.5.9	IF Construct (IF, END_IF)	4-88
4.5.10	IF-ELSE Construct (IF, ELSE, END_IF)	4-90
4.5.11	Expression (EXPRESSION)	4-91

4.6 Basic Function Instructions 4-93

4.6.1	Square Root (SQRT)	4-93
4.6.2	Sine (SIN)	4-94
4.6.3	Cosine (COS)	4-95
4.6.4	Tangent (TAN.)	4-97
4.6.5	Arc Sine (ASIN)	4-98
4.6.6	Arc Cosine (ACOS)	4-99
4.6.7	Arc Tangent (ATAN)	4-100
4.6.8	Exponential (EXP)	4-101
4.6.9	Natural Logarithm (LN)	4-102
4.6.10	Common Logarithm (LOG)	4-103

4.7 Data Shift Instructions 4-104

4.7.1	Bit Rotate Left (ROTL)	4-104
4.7.2	Bit Rotate Right (ROTR)	4-105
4.7.3	Move Bit (MOVB)	4-106
4.7.4	Move Word (MOVW)	4-108
4.7.5	Exchange (XCHG)	4-110
4.7.6	Table Initialization (SETW)	4-111
4.7.7	Byte-to-word Expansion (BEXTD)	4-113
4.7.8	Word-to-byte Compression (BPRESS)	4-114

4.7.9	Binary Search (BSRCH)	4-116
4.7.10	Sort (SORT)	4-117
4.7.11	Bit Shift Left (SHFTL)	4-118
4.7.12	Bit Shift Right (SHFTR)	4-120
4.7.13	Copy Word (COPYW)	4-121
4.7.14	Byte Swap (BSWAP)	4-122

4.8 DDC Instructions 4-123

4.8.1	Dead Zone A (DZA)	4-123
4.8.2	Dead Zone B (DZB)	4-124
4.8.3	Upper/Lower Limit (LIMIT)	4-126
4.8.4	PI Control (PI)	4-128
4.8.5	PD Control (PD)	4-133
4.8.6	PID Control (PID)	4-137
4.8.7	First-order Lag (LAG)	4-142
4.8.8	Phase Lead Lag (LLAG)	4-144
4.8.9	Function Generator (FGN)	4-147
4.8.10	Inverse Function Generator (IFGN)	4-151
4.8.11	Linear Accelerator/Decelerator 1 (LAU)	4-155
4.8.12	Linear Accelerator/Decelerator 2 (SLAU)	4-161
4.8.13	Pulse Width Modulation (PWM)	4-170

4.9 Table Manipulation Instructions 4-173

4.9.1	Read Table Block (TBLBR/TBLBRE)	4-173
4.9.2	Write Table Block (TBLBW/TBLBWE)	4-177
4.9.3	Search for Table Row (TBL SRL/TBL SRLE)	4-181
4.9.4	Search for Table Column (TBL SRC/TBL SRCE)	4-184
4.9.5	Clear Table Block (TBLCL/TBLCLE)	4-187
4.9.6	Move Table Block (TBLMV/TBLMVE)	4-190
4.9.7	Read Queue Table (QTBLR/QTBLRE and QTBLRI/QTBLRIE)	4-194
4.9.8	Write Queue Table (QTBLW/QTBLWE and QTBLWI/QTBLWIE)	4-198
4.9.9	Clear Queue Table Pointer (QTBLCL/QTBLCLE)	4-202

4.10 System Function Instructions 4-204

4.10.1	Counter (COUNTER)	4-204
4.10.2	First-in First-out (FINFOUT)	4-207
4.10.3	Trace (TRACE)	4-210
4.10.4	Read Data Trace (DTRC-RD/DTRC-RDE)	4-212
4.10.5	Send Message (MSG-SND)	4-216
4.10.6	Send Message Extended (MSG-SNDE)	4-218
4.10.7	Receive Message (MSG-RCV)	4-220
4.10.8	Receive Message Extended (MSG-RCVE)	4-221
4.10.9	Write SERVOPACK Parameter (MLNK-SVW)	4-223
4.10.10	Read SERVOPACK Parameter (MLNK-SVR)	4-228
4.10.11	Flash Operation (FLASH-OP)	4-233
4.10.12	Write Motion Register (MOTREG-W)	4-236
4.10.13	Read Motion Register (MOTREG-R)	4-238
4.10.14	Import (IMPORT/IMPORTL/IMPORTLE)	4-240
4.10.15	Export (EXPORT/EXPORTL/EXPORTLE)	4-248

4.11 Storage Operation Instructions 4-254

4.11.1	Open File (FOPEN)	4-254
4.11.2	Close File (FCLOSE)	4-257
4.11.3	Read Data from File (FREAD)	4-258
4.11.4	Write Data to File (FWRITE)	4-260
4.11.5	Set File Position Indicator (FSEEK)	4-262
4.11.6	Read Line from File to String (FGETS)	4-264
4.11.7	Write String to File (FPUTS)	4-266
4.11.8	Copy File (FCOPY)	4-268
4.11.9	Delete File (FREMOVE)	4-270
4.11.10	Rename File (FRENAME)	4-271

4.11.11 Create Directory (DCREATE)	4-274
4.11.12 Delete Directory (DREMOVE)	4-276
4.11.13 Send File to FTP Server (FTPPUT)	4-277

4.12 String Operation Instructions 4-280

4.12.1 Convert Integer to String (INT2STR)	4-280
4.12.2 Convert Real Number to String (REAL2STR)	4-282
4.12.3 Convert String to Integer (STR2INT)	4-283
4.12.4 Convert String to Real Number (STR2REAL)	4-284
4.12.5 Store String (STRSET)	4-286
4.12.6 Partially Delete String (STRDEL)	4-287
4.12.7 Copy String (STRCPY)	4-288
4.12.8 Get String Length (STRLEN)	4-290
4.12.9 Concatenate Strings (STRCAT)	4-291
4.12.10 Compare Strings (STRCMP)	4-293
4.12.11 Insert String (STRINS)	4-294
4.12.12 Find String (STRFIND)	4-296
4.12.13 Extract String (STREXTR)	4-297
4.12.14 Extract String from End (STREXTRE)	4-299
4.12.15 Delete Spaces at String Ends (STRTRIM)	4-300

5

Features of the MPE720 Engineering Tool

5.1 Ladder Program Runtime Monitoring 5-4

5.2 Search/Replace 5-5

5.2.1 Searching and Replacing in Programs	5-5
5.2.2 Searching and Replacing in Project Files	5-7

5.3 Cross References 5-10

5.4 Checking for Multiple Coils 5-13

5.5 Forcing Coils ON and OFF 5-14

5.5.1 Forcing Coils ON or OFF from a Ladder Program	5-14
5.5.2 Changing the Forced ON/OFF Status from the Force Coil List Pane	5-14

5.6 Viewing Called Programs 5-17

5.7 Register Lists 5-18

5.7.1 Displaying the Register Map	5-18
5.7.2 Switching the Register Map Display	5-19
5.7.3 Editing Data	5-20

5.8 Tuning Panel 5-21

5.9 Enabling and Disabling Ladder Programs 5-22

5.10 Watching 5-23

5.10.1 Displaying Watch Data	5-23
5.10.2 Editing the Value Column	5-23

5.11 Security 5-24

5.12 Tracing 5-25

5.13	Advanced Programming	5-26
	5.13.1 Motion Programs	5-26

Appendix **A** System Service Registers

A.1	Overview of System Registers	A-2
A.2	Common to All Drawings	A-3
A.3	Exclusive to DWG.H (High-speed Scan Process Drawings) . . .	A-4
A.4	Exclusive to DWG.L (Low-speed Scan Process Drawings)	A-5
A.5	Scan Execution Status and Calendar.	A-6
A.6	System Program Software Numbers and Remaining Program Memory Capacity . .	A-7

Appendix **B** Sample Programs

B.1	Jogging from the Control Panel	B-2
B.2	Motion Program Control.	B-3
B.3	Simple Synchronized Operation of Two Axes with a Virtual Axis . .	B-4

Appendix **C** Format for EXPRESSION Instructions

C.1	Elements That You Can Use in Numeric Expressions	C-2
	C.1.1 Operators	C-2
	C.1.2 Operands	C-3
	C.1.3 Instructions That You Can Use with EXPRESSION Instructions	C-4
C.2	Notational Limitations	C-5
	C.2.1 Arithmetic and Logic Operators.	C-5
	C.2.2 Comparison Operators	C-5
	C.2.3 Logic Operators.	C-5
	C.2.4 Substitution Operator	C-6
	C.2.5 Functions	C-6
	C.2.6 Parentheses	C-6

Appendix **D** Precautions on Motion Parameters

Appendix **E** Machine Controller Specifications

Index

Revision History

Features and Overview of Ladder Programs

1

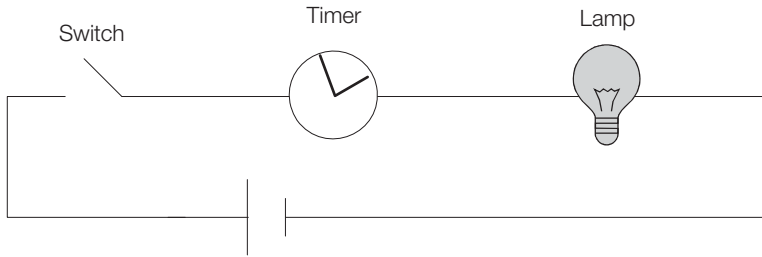
This section describes the features and gives an overview of ladder programs.

1.1	What Is a Ladder Program?	1-2
1.2	Features	1-3
1.2.1	The Various Execution Timing of Ladder Drawings	1-3
1.2.2	Program Modules	1-4
1.2.3	Programming Complicated Numeric Operations	1-4
1.2.4	Communications Control with External Devices	1-5
1.2.5	Complete Synchronization with Motion Control	1-5
1.3	Introduction	1-6
1.3.1	Ladder Program Editor	1-6
1.3.2	Ladder Drawings	1-7
1.3.3	User Functions	1-13
1.3.4	Table Data	1-18

1.1 What Is a Ladder Program?

A ladder program uses ladder language instructions and registers to symbolically represent electrical circuits consisting of switches, timers, lamps, and other devices.

<Conceptual Circuit>



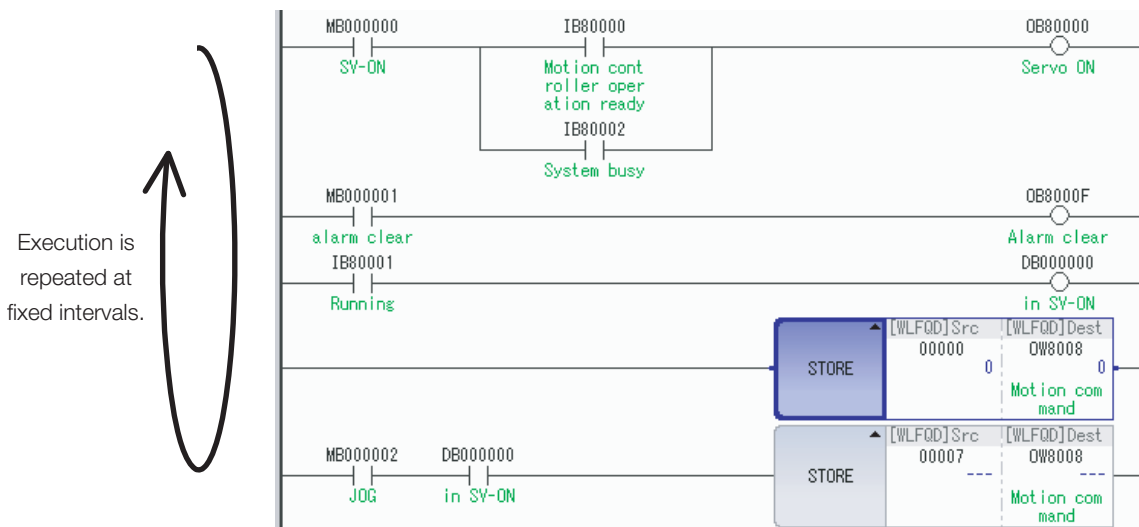
<Ladder Programming>



Ladder programming allows you to easily program large, complex circuits.

Each of the ladder programs that you create is executed in a single scan and then executed repeatedly at fixed intervals.

<Ladder Programming Example>



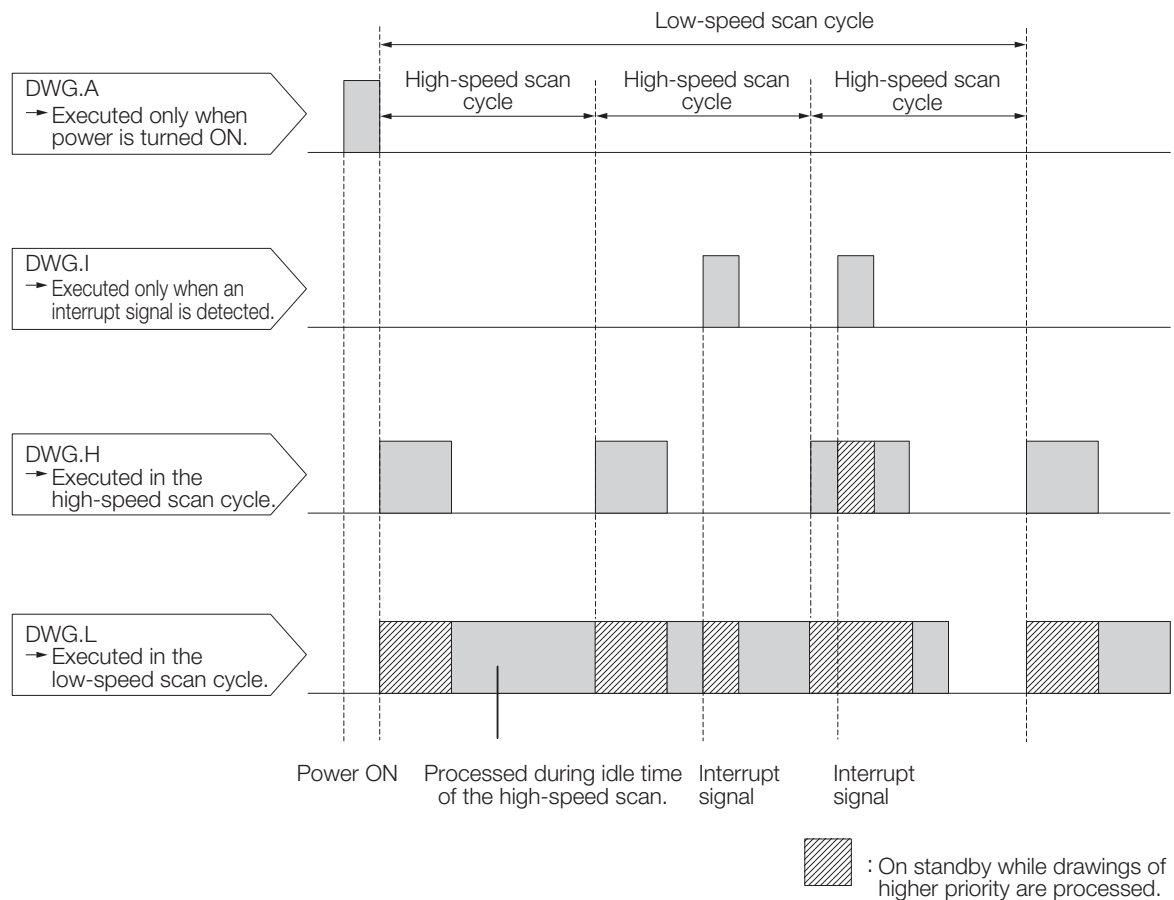
1.2 Features

This section describes the features of ladder programs.

1.2.1 The Various Execution Timing of Ladder Drawings

Ladder programs are managed in units of drawings (DWG). These are called ladder drawings. In the Machine Controller, ladder drawings are executed at various times, as illustrated in the following figure.

Processing can be executed at the appropriate time by programming it in the appropriate ladder drawing.



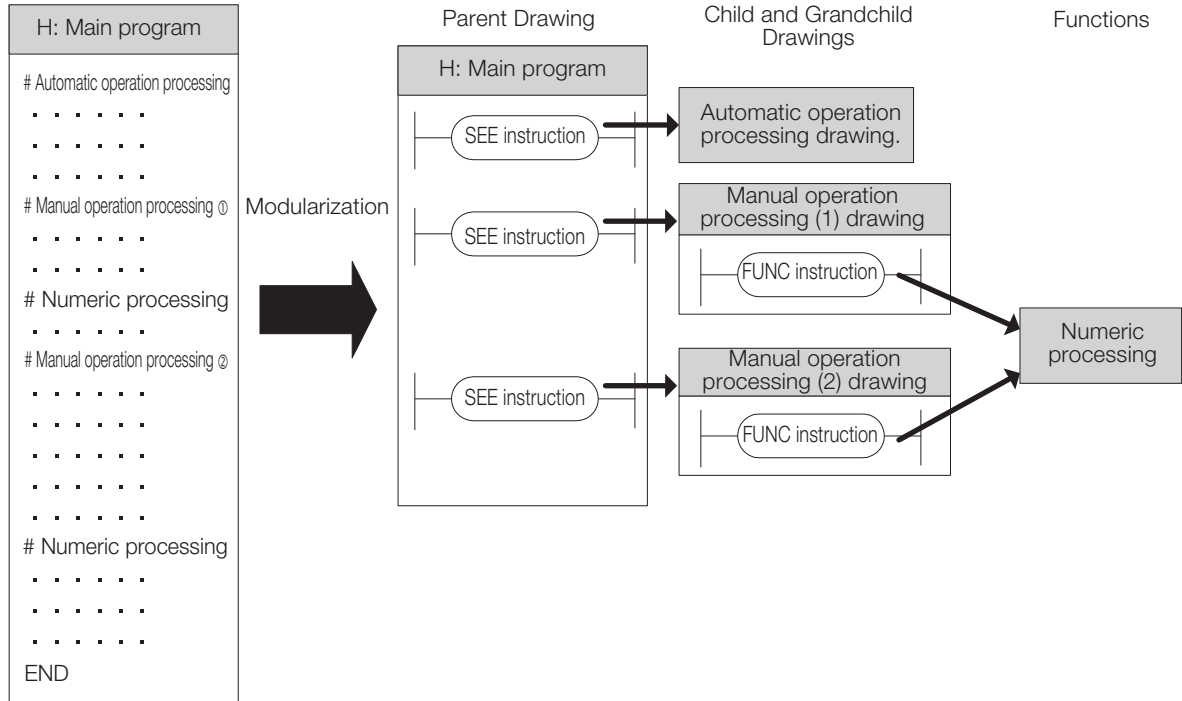
The following table gives the execution timing for each drawing.

Ladder Drawing	Priority*	Execution Timing (Processing Example)
DWG.A	1 (High)	This drawing is executed only once when the power supply is turned ON (e.g., for data initialization).
DWG.I	2 (↑)	This drawing is executed when an interrupt signal is detected (e.g., for interrupt processing for external signals).
DWG.H	3 (↑)	This drawing is executed every high-speed scan cycle (e.g., for motion control).
DWG.L	4 (Low)	This drawing is executed every low-speed scan cycle (e.g., for touch panel display processing).

* The drawings with lower numbers have higher execution priority.

1.2.2 Program Modules

The main program can be separated into modular units to suit different processing requirements, such as child drawings, grandchild drawings, and functions, to make the program easier to read. The following example illustrates a modular program.

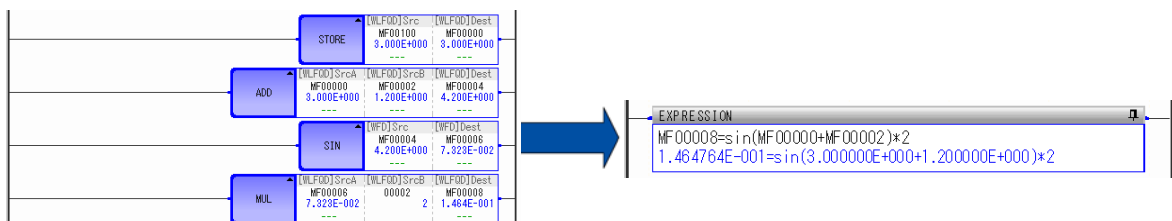


1.2.3 Programming Complicated Numeric Operations

Complicated calculations written over several lines can be written easily by using a single EXPRESSION instruction.

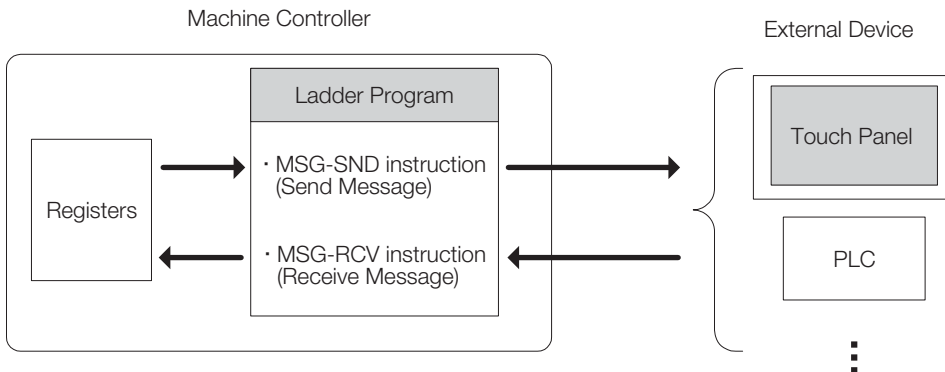
Variables, structures, and basic functions, such as those for sine and cosine calculations, can be programmed using familiar C-like expressions.

You can display the current value inside expressions in the same way as you can for other ladder language instructions.



1.2.4 Communications Control with External Devices

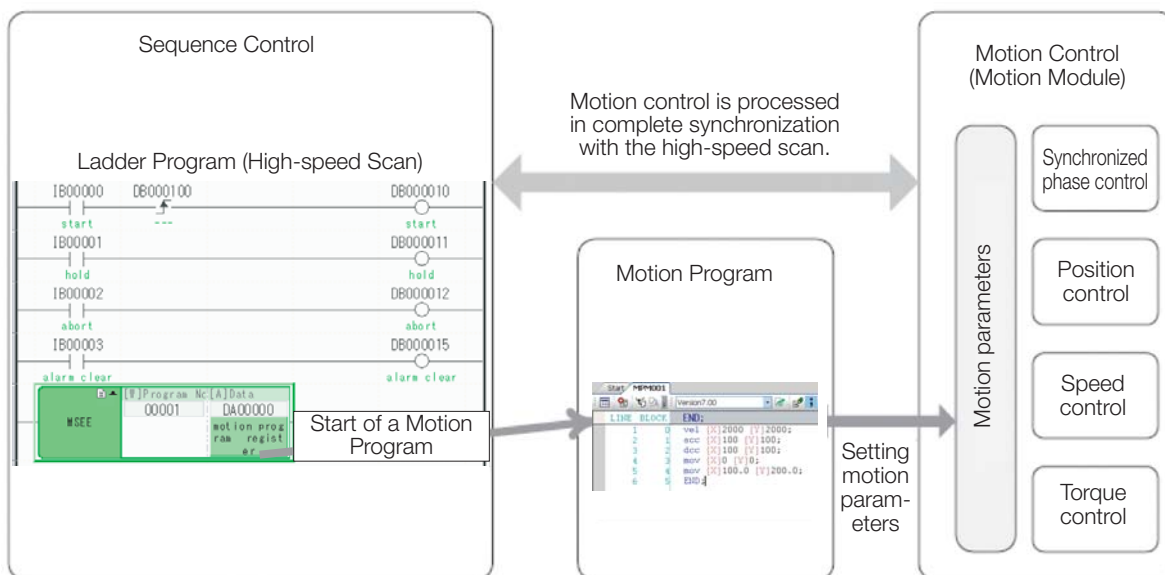
The MSG-SND and MSG-RCV ladder language instructions support various protocols and can be used to control communications with many external devices, such as a touch panels or host PLCs. This allows external devices to access registers in the Machine Controller.



Information Instead of using a ladder program, the Machine Controller can also communicate with external devices by using I/O message communications or automatic reception. Refer to the following manual for details.
 📖 MP3000 Series Communications User's Manual (Manual No.: SIEP C880725 12)

1.2.5 Complete Synchronization with Motion Control

Ladder programs that are started in the high-speed scan are processed in complete synchronization with motion control operations. This allows you to call and process a motion program that performs complicated motion control synchronously with a ladder program.

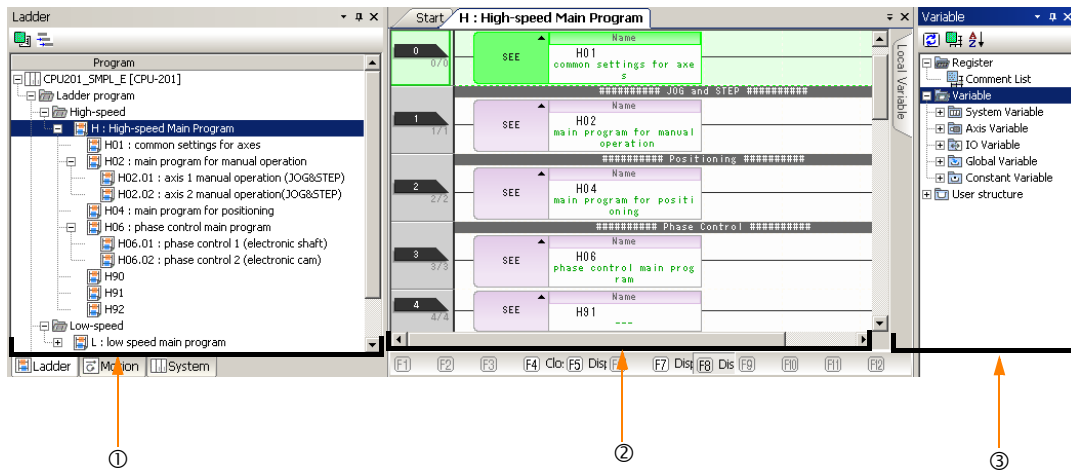


1.3 Introduction

This section provides an overview of ladder programming.

1.3.1 Ladder Program Editor

In MPE720 version 7, ladder programs are created and edited in the panes that are shown below.



① **Ladder Pane**

Ladder programs are displayed by drawing.
 Refer to the following section for details on drawings.
 📖 1.3.2 Ladder Drawings on page 1-7

② **Tab Page to Edit Ladder Program**

This tab page is used to edit ladder programs.

③ **Variable Pane**

This pane displays variables. Refer to the following section for details on registers.
 📖 Chapter 3 Registers

In addition to the panes and tab pages that were just described, various other panes, tab pages, and tool bars also exist.

1.3.2 Ladder Drawings

Ladder programs are managed as drawings (ladder diagrams) that are identified by their drawing numbers (DWG numbers). The ladder drawings form the basis of the ladder programs.

Drawing Types and Hierarchical Configuration

This section describes the types of ladder drawings and their hierarchical configuration.

◆ Types

Ladder drawings are divided into four different types based on their purpose.

- **DWG.A (Startup Drawings)**
This type of ladder drawing is used to set register data. These ladder drawings are executed before high-speed scan process drawings and low-speed scan process drawings.
- **DWG.I (Interrupt Drawings)**
This type of ladder drawing is used to perform processing with priority given to signals input from an Optional Module. These ladder drawings are executed with higher priority than high-speed scan process drawings regardless of the scan cycle.
- **DWG.H (High-speed Scan Process Drawings)**
This type of ladder drawing is used to perform motion control or high-speed I/O control.
- **DWG.L (Low-speed Scan Process Drawings)**
This type of ladder drawing is used for communications with HMIs and external devices as well as for standard I/O control.

The following table lists the priority, execution conditions, and maximum number of drawings for each type of ladder drawing.

Drawing Type	Priority*	Execution Condition	Maximum Number of Drawings
DWG.A (Startup Drawings)	1	Power ON (These drawings are executed once when the power supply is turned ON.)	64
DWG.I (Interrupt Drawings)	2	External interrupt (These drawings are executed when a DI interrupt or counter match interrupt is received from an Option Module.)	64
DWG.H (High-speed Scan Process Drawings)	3	Started at fixed intervals. (These drawings are executed once every high-speed scan.)	1000
DWG.L (Low-speed Scan Process Drawings)	4	Started at fixed intervals. (These drawings are executed once every low-speed scan.)	2000

* Drawings with lower numbers have higher priority.

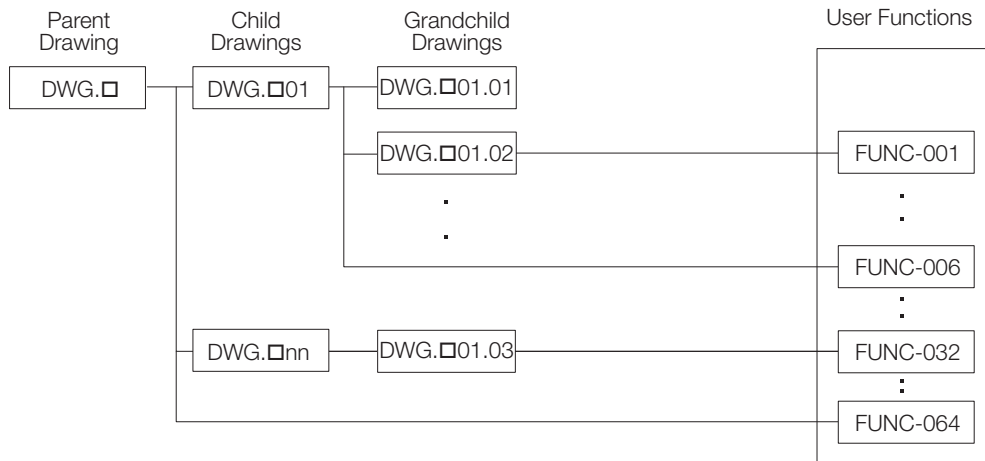
◆ Hierarchical Configuration

There are four types of ladder drawings: parent drawings, child drawings, grandchild drawings, and operation error drawings.

- **Parent Drawings**
These drawings are automatically executed by the system program when the execution conditions are met.
- **Child Drawings**
These drawings are executed when they are called from a parent drawing with a SEE instruction.
- **Grandchild Drawings**
These drawings are executed when they are called from a child drawing with a SEE instruction.
- **Operation Error Drawings**
These drawings are automatically executed by the system program when an operation error occurs.

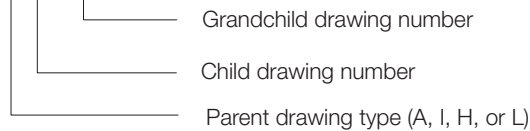
A parent drawing cannot call a child drawing from a different type of drawing. Similarly, a child drawing cannot call a grandchild drawing from a different type of drawing. A parent drawing cannot call a grandchild drawing directly. The parent drawing first must call the child drawing, and then the child drawing must call the grandchild drawing. This is called the hierarchical configuration of drawings.

The following figure shows the parent-child-grandchild structure in which a program is created.



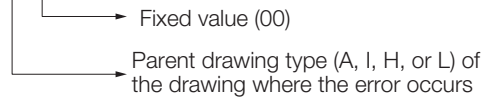
Note: □ = A, I, H, or L

DWG notation: DWG.□□.□



Note: The following notation is used for operation error drawings.

DWG.□ 00



The breakdown of the number of ladder drawings in each category is given in the following table.

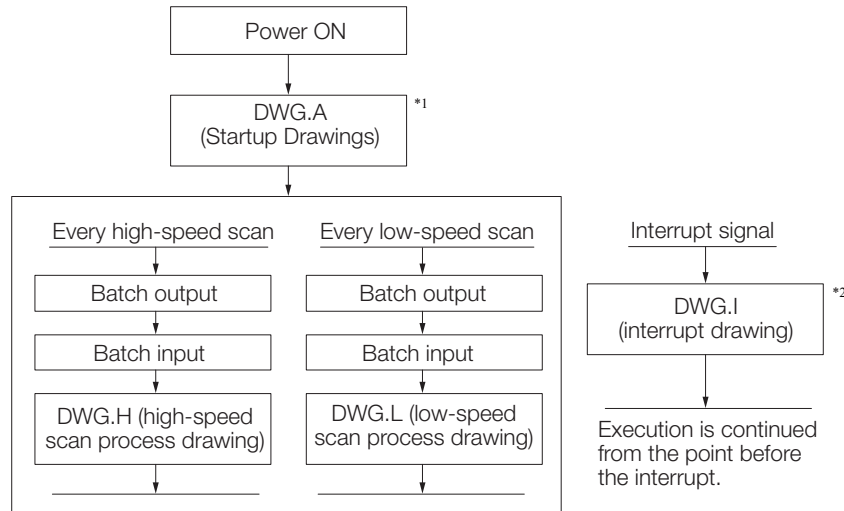
Drawings	Number of Drawings			
	DWG.A	DWG.I	DWG.H	DWG.L
Parent Drawings	1	1	1	1
Operation Error Drawings	1	1	1	1
Child Drawings	Total of 62 max.	Total of 62 max.	Total of 998 max.	Total of 1,998 max.
Grandchild Drawings				

Information

There are separate functions that can be called from the drawings as required. Functions are executed when they are called from a parent, child, or grandchild drawing with the FUNC instruction. You can create up to 2,000 functions.

Controlling the Execution of Drawings

Drawings are executed based on their priorities, as shown in the following figure.



*1. DWG.A drawings are executed immediately after the power supply is turned ON.

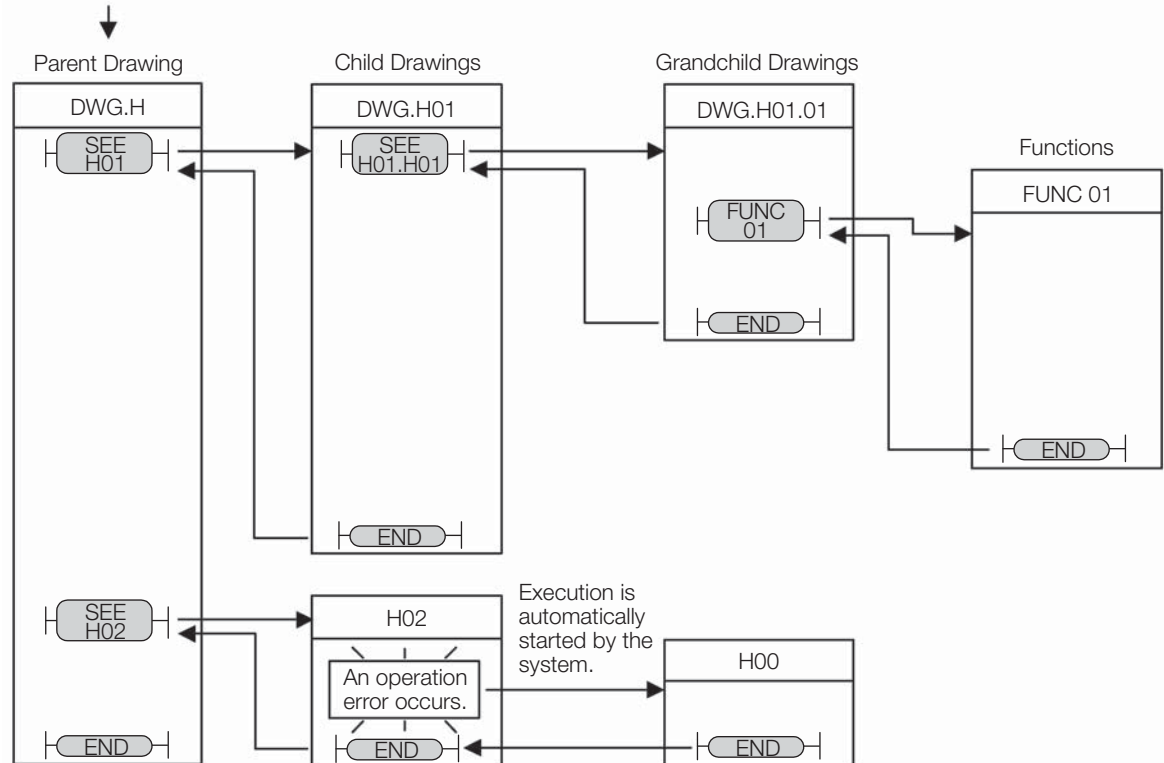
*2. When an interrupt signal is input, execution of the DWG.I drawing is given priority even if execution of a DWG.H or DWG.L drawing is currently in progress.

Note: The parent drawing of each drawing is automatically called and executed by the system.

◆ Execution Processing of Drawings

The drawings are executed by calling them from the top to the bottom, following the hierarchy of the drawings. The following figure illustrates the execution processing of a high-speed scan drawing (DWG.H).

Execution is started by the system program when the execution condition is met.



Note: 1. The parent drawing is automatically called and executed by the system. Child drawings and grandchild drawings are executed by calling them from a parent or a child drawing using the SEE instruction.

2. You can call functions from any drawing. You can also call functions from other functions.

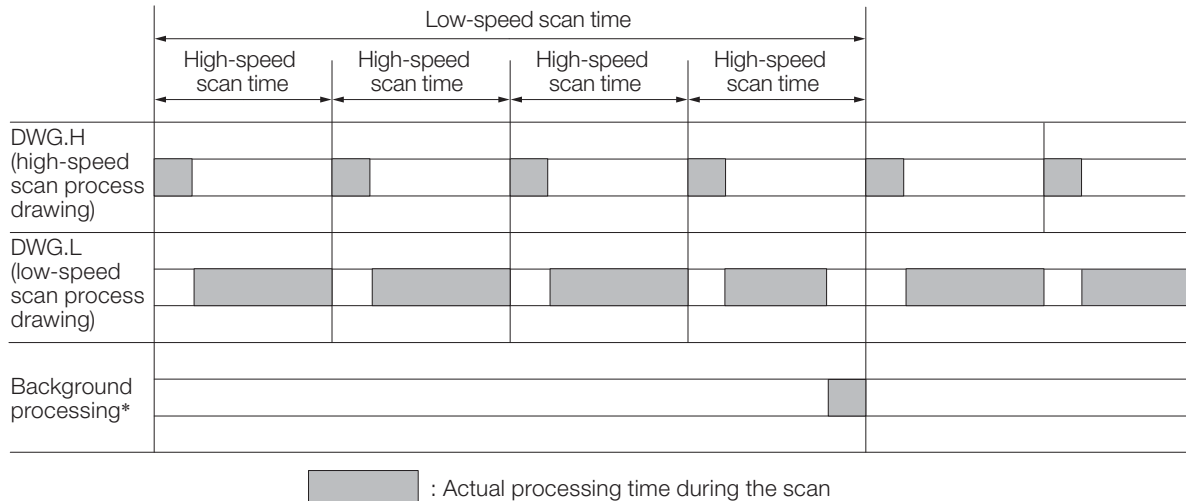
3. If an operation error occurs, the operation error drawing for the drawing type will be started automatically.

4. Always specify 00 as the drawing number for operation error drawings.

◆ Scheduling the Execution of High-speed and Low-speed Scan Process Drawings

High-speed scan process drawings (DWG.H) and low-speed scan process drawings (DWG.L) cannot be executed at the same time. DWG.L drawings are executed during the idle time of DWG.H drawings.

The period during which DWG.H drawings are executed is called the high-speed scan time. The period during which DWG.L drawings are executed is called the low-speed scan time.

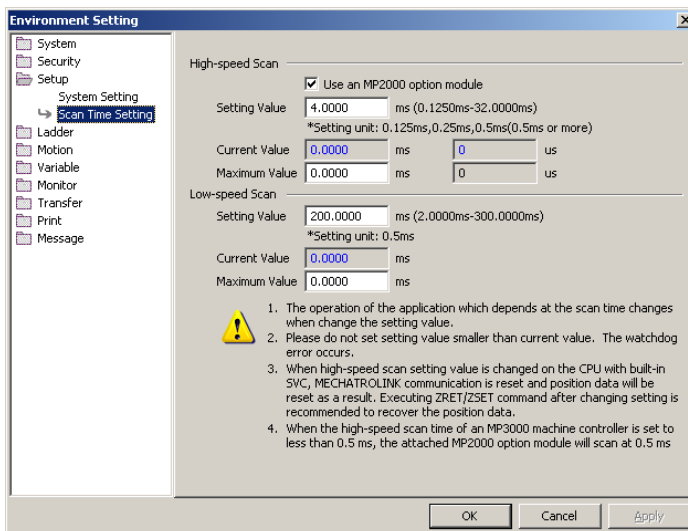


* This time is used to execute internal system processing, such as communications processing.

◆ Setting the High-speed and Low-speed Scan Times

Use MPE720 version 7 and perform the procedure given below to set the high-speed and low-speed scan times.

1. Select **File – Environment Settings** from the menu bar. Alternatively, click the **System Setting Icon** on the **My Tool View** of the **Start Tab Page**. The **Environment Setting Dialog Box** is displayed.
2. Select **Setup – Scan Time Setting**. The following dialog box will be displayed.



Setting Value: Enter the scan time settings.

Current Value: A value of 0.0 ms is displayed when the MPE720 is offline. Otherwise, the actual processing times for the scans are displayed.

Maximum Value: The maximum processing time for the scan is displayed. You can set the maximum value. The setting is retained until it is exceeded.

3. Enter the high-speed scan time in the Setting Value Box under High-speed Scan. Enter the low-speed scan time in the Setting Value Box under Low-speed Scan.

The following table shows the possible set values and default values for each scan time.

Item	Possible Set Values	Default
High-speed Scan Time	0.125 to 32 ms (in 0.125-ms increments)	4.0 ms
Low-speed Scan Time	2.0 to 300.0 ms (in 0.5-ms increments)	200.0 ms

Note: The possible set values and default values depend on the model. Refer to the user's manual for the Module you are using for details.



Observe the following precautions when setting the high-speed scan time and low-speed scan time.

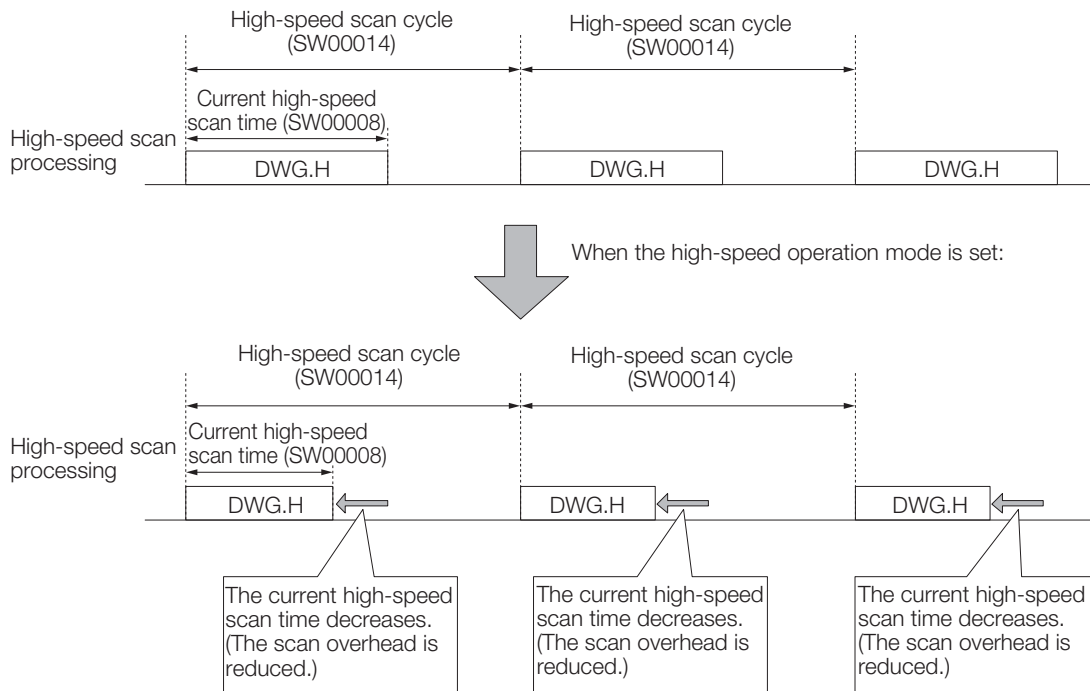
1. Set the scan set value so that it is 1.25 times greater than the maximum value. If the scan set value is too close to the maximum value, the refresh rate of the MPE720 window will noticeably drop and can cause communications timeout errors to occur. If the maximum value exceeds the scan set value, a watchdog error may occur and cause the Machine Controller system to shut down.
2. If you are using MECHATROLINK-II or MECHATROLINK-III, set values that are an integral multiple of the communications cycle. If you change the communications cycle, check the scan time set values.
3. Do not change the scan set value while the Servo is ON. Never change the scan set value while an axis is in motion (i.e., while the motor is rotating). Doing so may cause the motor to rotate out of control.
4. After changing or setting a scan time, always save the data to flash memory.

◆ High-speed Drawing Operation Mode Settings

The high-speed drawing operation mode is the mode that is set for DWG.H drawings.

If no DWG.I drawings are used, select the high-speed mode. This optimizes the processing of DWG.H drawings.

If DWG.I drawings are used, select the normal mode. If the high-speed mode is selected, DWG.I drawings will not be executed.

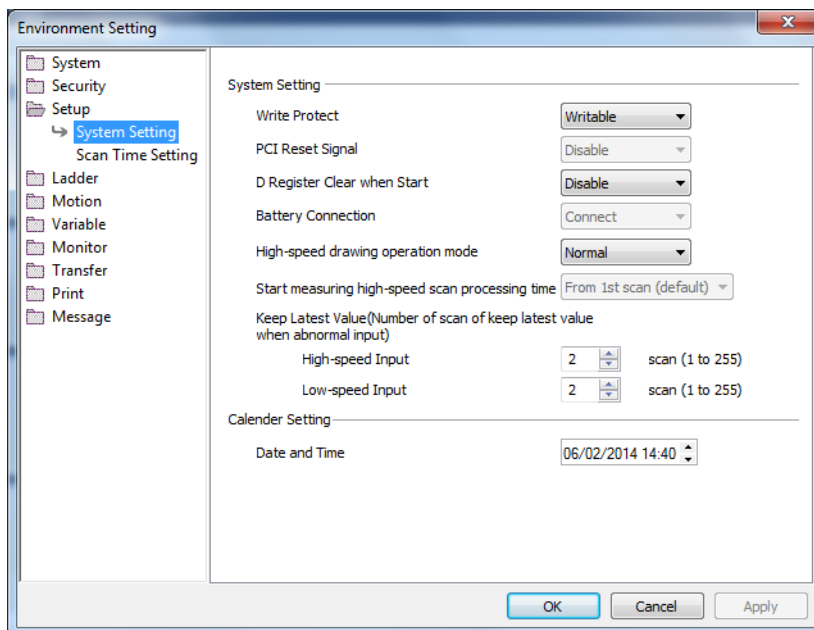


- Information** • DWG.A, DWG.I, and DWG.L drawings do not have operation mode settings.
- The more often the following instructions are used, the greater the effect that the optimization will have on DWG.H processing.

Type	Function
Relay Circuit Instructions	Rising-edge NO Contact
	Falling-edge NO Contact
	Rising-edge NC Contact
	Falling-edge NC Contact
	Rising-edge Pulse
	Falling-edge Pulse
	Coil
	Reverse Coil
	Rising-edge Detection Coil
	Falling-edge Detection Coil
	Set Coil
Reset Coil	
Numeric Operation Instructions	+1 Increment
	-1 Decrement

Perform the following procedure with MPE720 version 7 to set the high-speed drawing operation mode.

1. Select **File – Environment Settings** from the menu bar. Alternatively, click the **System Setting** icon on the **My Tool View** of the **Start Tab Page**. The **Environment Setting Dialog Box** is displayed.
2. Select **Setup – System Setting**. The following dialog box will be displayed.



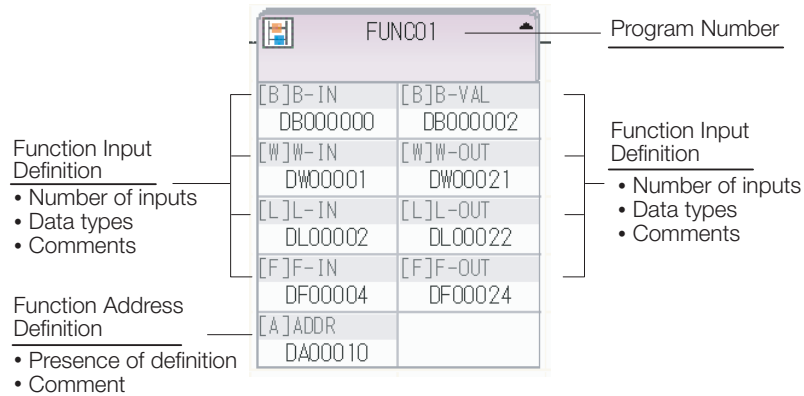
3. Select **High-speed** or **Normal** for the **High-speed Drawing Operation Mode**.

1.3.3 User Functions

What Is a User Function?

A user function contains a function definition (program number and I/O definitions) and processing instructions that are defined by the user.

The following figure shows an example of a function definition.



Overview of User Functions

The processing to be performed by a user function is created using a ladder program.

User functions are executed when they are called from a parent, child, or grandchild drawing with the FUNC instruction.

The following user function calls are also allowed.

- User functions can be freely called from any drawing.
- User functions can be called simultaneously from drawings of different types and different hierarchy levels.
- User functions can call other user functions.
- User functions can be called any number of times from different programs.

The use of functions provides the following advantages.

- Easy user program modularization
- Easy user programming and program maintenance

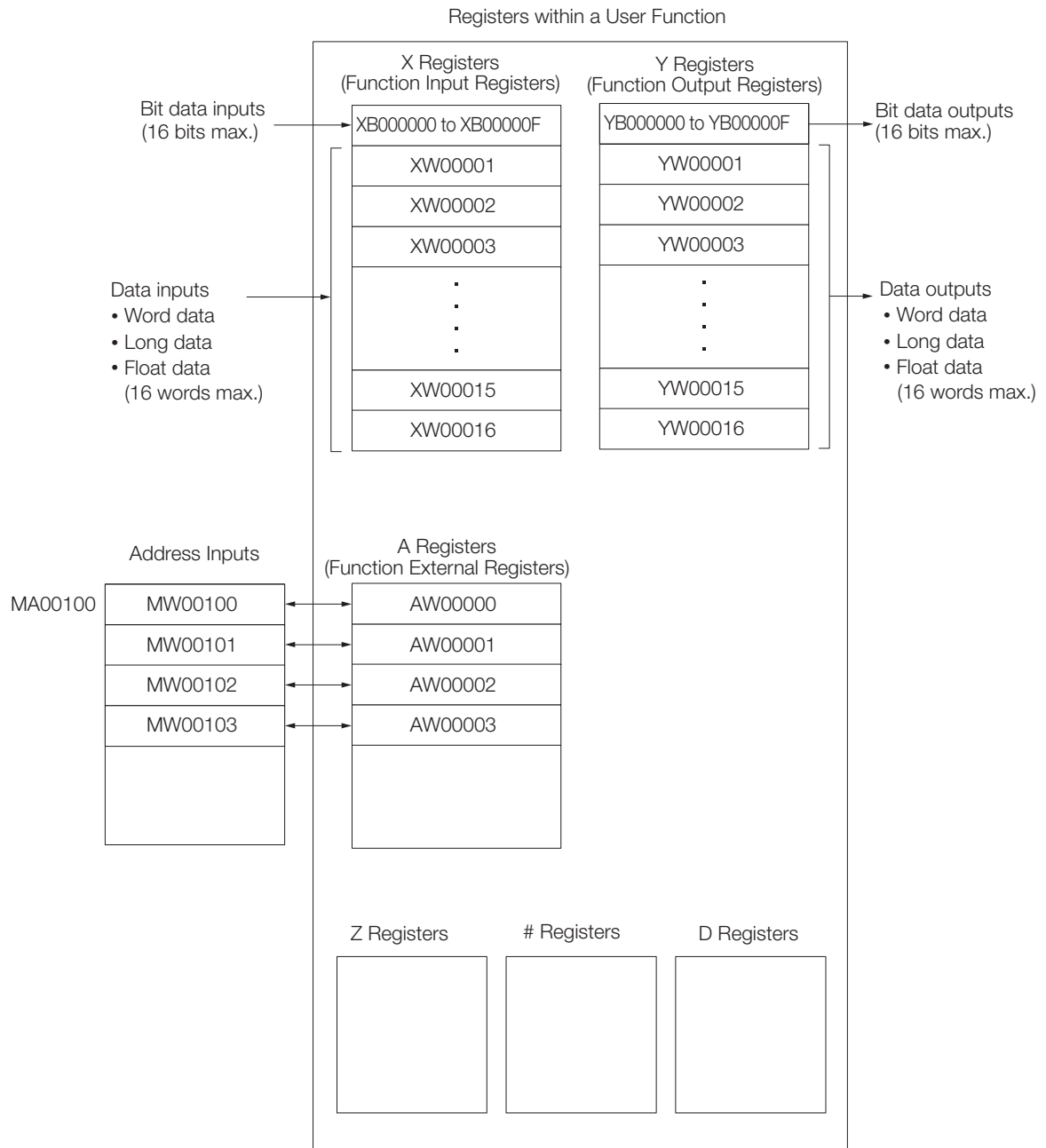


Important

When you call a user function, consider what values could be in the registers in each function, and perform initialization as needed. Refer to the following section for details.

Chapter 3 Registers – 3.2.1 Precautions When Using Local Registers within a User Function on page 3-5

The following diagram shows the relation between I/O data for a user function and the registers within that user function.



Information

- The X, Y, Z, and D registers are initialized to different values when a user function is called. Refer to the following section for details.
📖 Chapter 3 Registers – 3.2.1 Precautions When Using Local Registers within a User Function on page 3-5
- The S, M, I, O, and C registers can also be accessed from within a function.

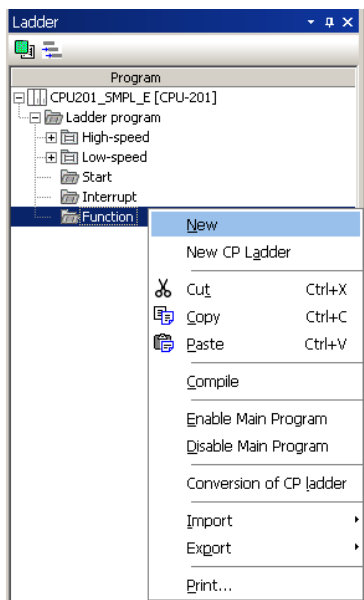
Creating User Functions

This section describes how to create a user function that has, as an example, the following specifications.

Function Definition Item	Name	Remarks
Program Number	FUNC01	–
Function Input Value	IN	Integer data
Function Output Value 1	OUT1	Integer data
Function Output Value 2	OUT2	Integer data
Processing Details		
Multiply the function input value (IN) by 2 and output it to function output value 1 (OUT1). Multiply the function input value (IN) by 3 and output it to function output value 2 (OUT2).		

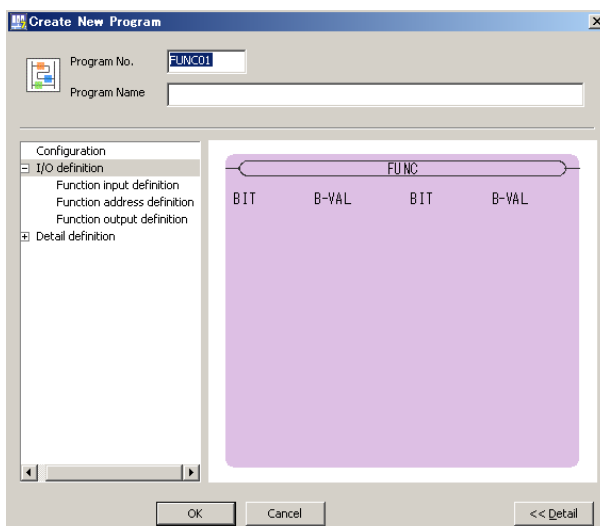
◆ Procedure

1. Select **Programming – Ladder program** from the Launcher.
The Ladder Pane will be displayed.
2. Right-click **Function** under **Ladder program**, and select **New**.

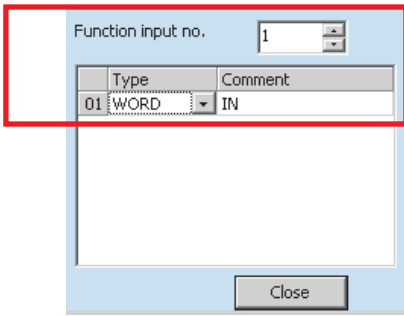


The Create New Program Dialog Box will be displayed.

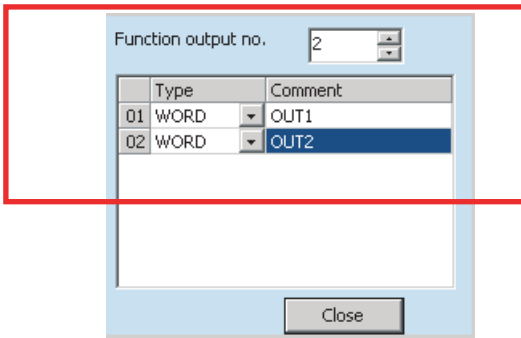
3. Enter "FUNC01" in the **Program No. Box**.



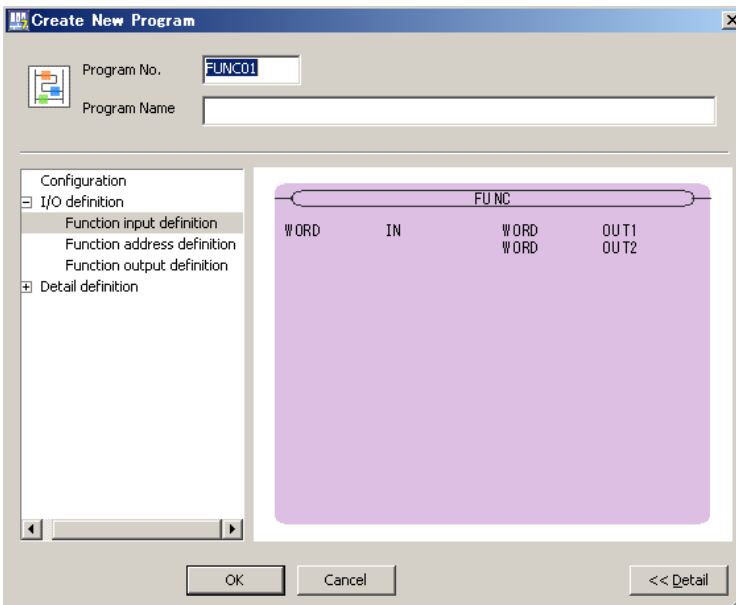
4. Select Function input definition under I/O definition and enter the following information.



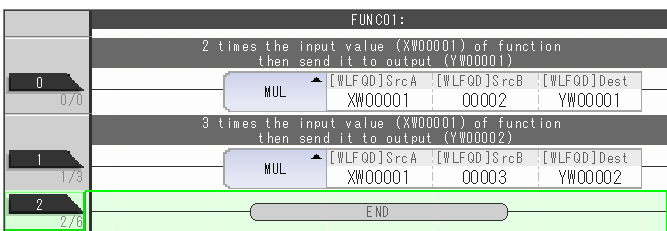
5. Select Function output definition under I/O definition and enter the following information.



6. Click the OK Button. This concludes setting the function definition.



7. Create the following ladder program in the drawing of the FUNC01 sample user function that was created in step 5.



8. While displaying the ladder program, select **Compile – Compile** from the menu bar to compile the program.
When the compilation is finished, the ladder program will be saved automatically.

Important

If an error is displayed in the Output Pane during compilation, the ladder program will not be saved.

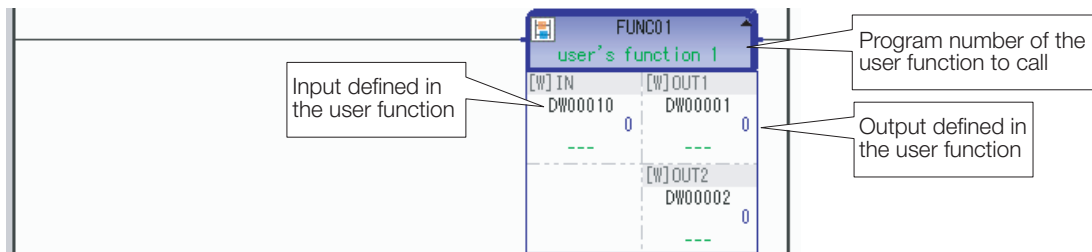
This concludes the creation of the user function.

Calling the User Function

The user function is ready to be called by using a FUNC instruction in the ladder drawing. This section describes how to call the sample user function from the high-speed drawing (DWG.H).

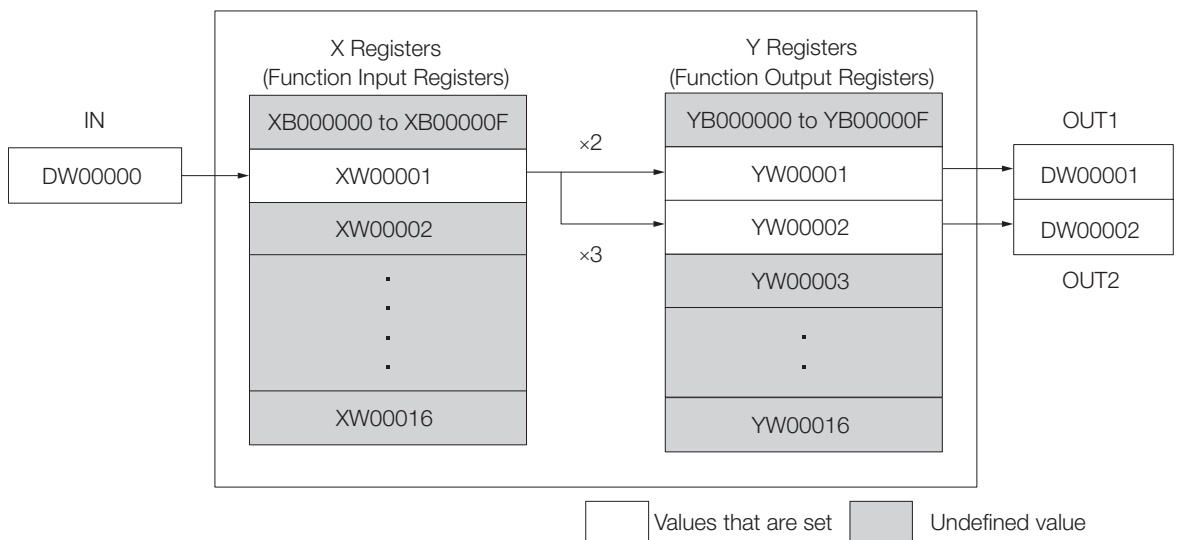
■ Example for Calling the FUNC01 User Function from DWG.H

Program a FUNC instruction in DWG.H as shown below.

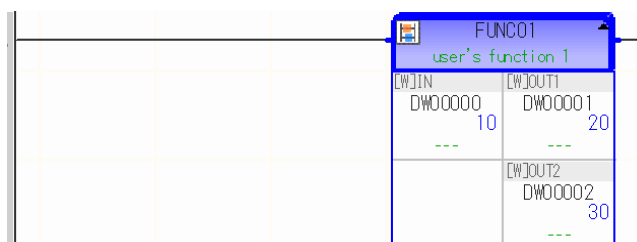


This diagram shows a conceptual image of what the programming shown above accomplishes.

Registers within the FUNC01 User Function



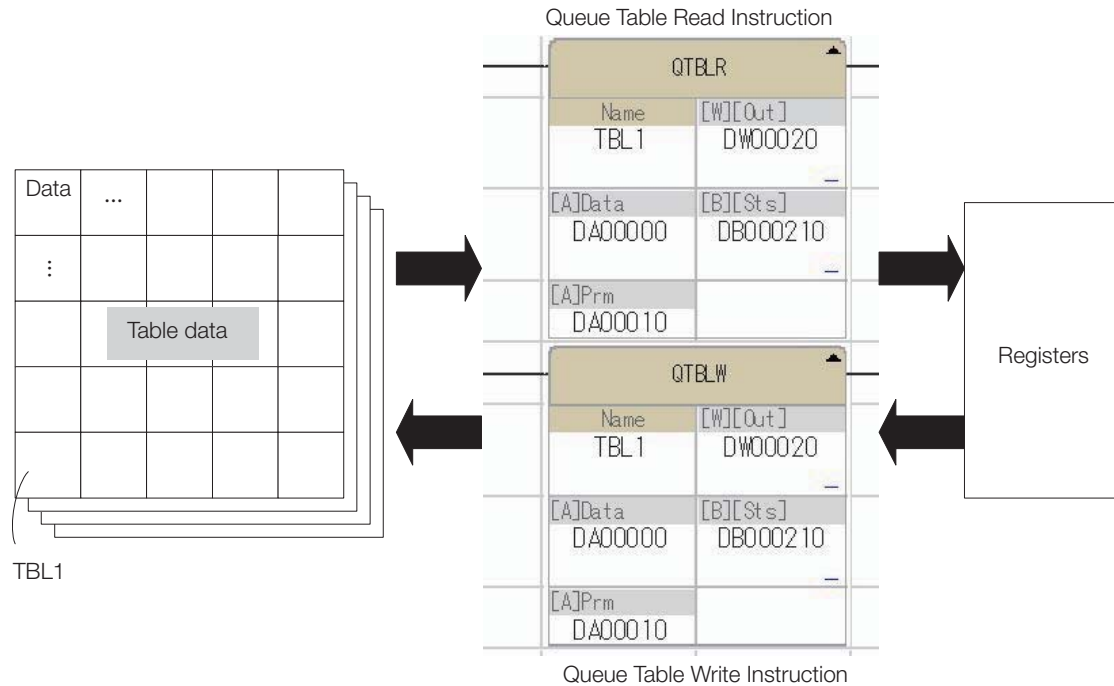
When DW00000 in DWG.H is set to 10, DW00001 becomes 20 and DW00002 becomes 30, demonstrating that the sample user function was called correctly.



1.3.4 Table Data

What Is Table Data?

Table data is data that is managed in tabular form. The data is stored separately from the registers. Data can be copied from a table to registers or from registers to a table by executing table instructions in the ladder program. Tables can also be used to hold data when there is not a sufficient range of registers.



Creating Table Data

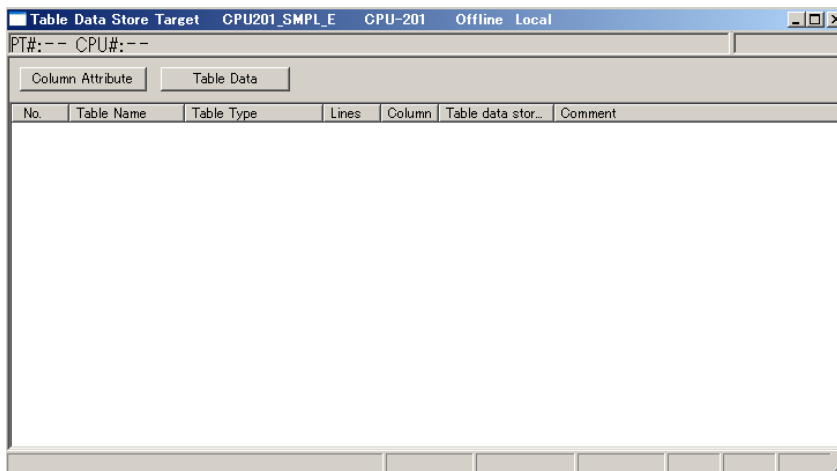
To create table data, set the table definition information and column attributes as listed below.

Table Definition Information	Description
Table Name	This is the name of the table.
Table Type	Select an array-type or record-type table. <ul style="list-style-type: none"> • Array type: Specifies a table where all columns have the same attributes. • Record type: Specifies a table where each column has a different attribute.
Number of Columns	This is the number of columns in the table. (32,767 columns max.)
Number of Rows	This is the number of rows in the table. (65,535 rows max.)
Table Comment	This is a comment for the table.
Table Data Storage Location	Select normal or battery backup. <ul style="list-style-type: none"> • Normal: The maximum size per table is 5 MB. • Battery backup: The maximum size per table is 3 MB. The details on the maximum size of tables and which models have battery backup storage are given below. <ul style="list-style-type: none"> • MP3100 <ul style="list-style-type: none"> 16 axes: Total size: 15 MB, size for battery backup: 1 MB 32 axes: Total size: 31 MB, size for battery backup: 3 MB • MP3200 <ul style="list-style-type: none"> 16 and 32 axes: Total size: 32 MB, size for battery backup: 3 MB • MP3300 <ul style="list-style-type: none"> 16 axes: Total size: 15 MB, size for battery backup: 1 MB 32 axes: Total size: 31 MB, size for battery backup: 3 MB

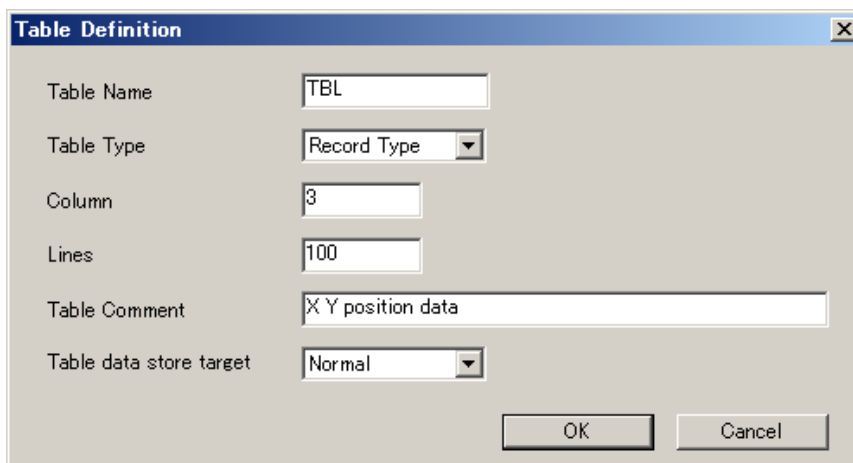
Column Attributes	Description
Column Name	This is the name of the column.
Data Type	The data type can be integer, double-length integer, quadruple-length integer, real number, double-precision real number, or text string.
Size	This is the length of the data type.
Display Type	The display type can be binary, decimal, hexadecimal, real number, or text string.
Column Comment	This is a comment for the column.

■ Procedure

1. Select **Utility – Engineering Builder** from the Launcher.
The Engineering Builder will start.
2. Select **File – Open – Define Data Table – Data Table Map** from the menu bar in the Engineering Builder.
The Table Data Store Target Dialog Box will be displayed.



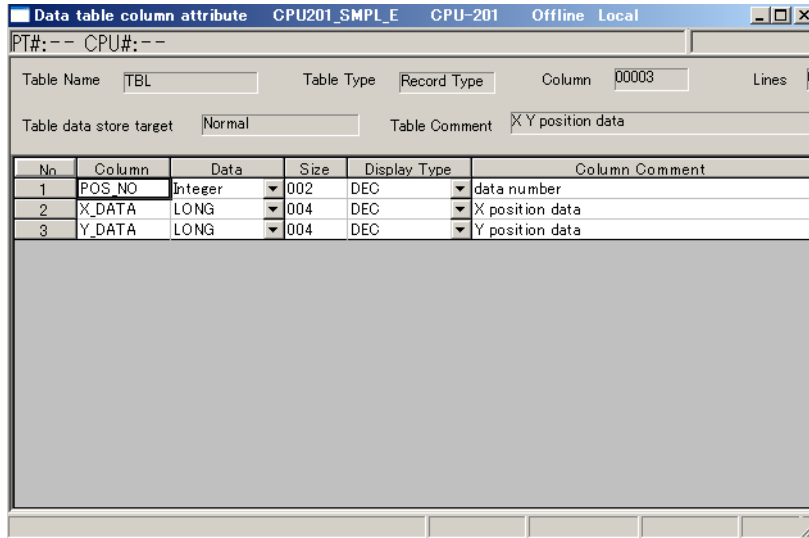
3. With the Table Data Store Target Dialog Box displayed, select **File – Create New** from the menu bar in the Engineering Builder.
The Table Definition Dialog Box will be displayed.
4. Set the table definition information and click the **OK** Button.



The Data Table Column Attribute Dialog Box will be displayed.

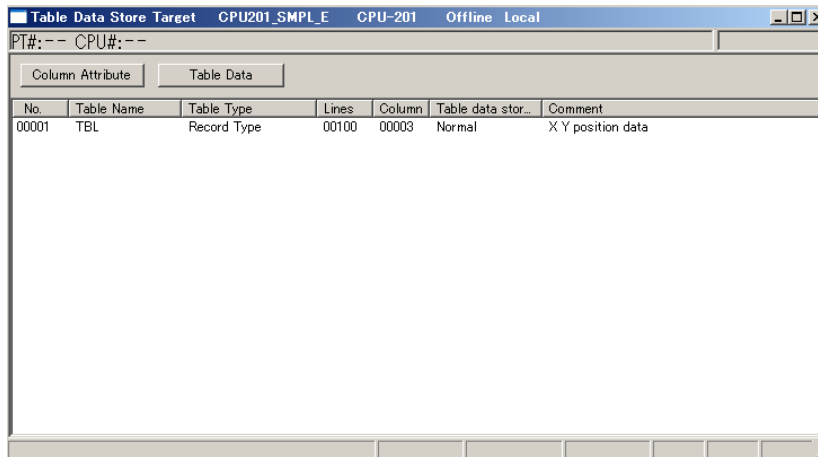
5. Set the table data column attributes.

Note: If the table is set to an array-type table in the table definition, set only one column attribute.



6. Select **File – Save Project** from the menu bar.

The Table Data Store Target Dialog Box displayed in step 2 will show the table created with this procedure.



This concludes the creation of the data table.

Information

1. When a table is created, the contents are initialized to 0.
2. Select the table that was created in the Table Data Store Target Dialog Box, and click the **Table Data** Button to read or write table data.
3. Use the table instructions to perform operations on the table data from a ladder program.

Ladder Program Development Flow

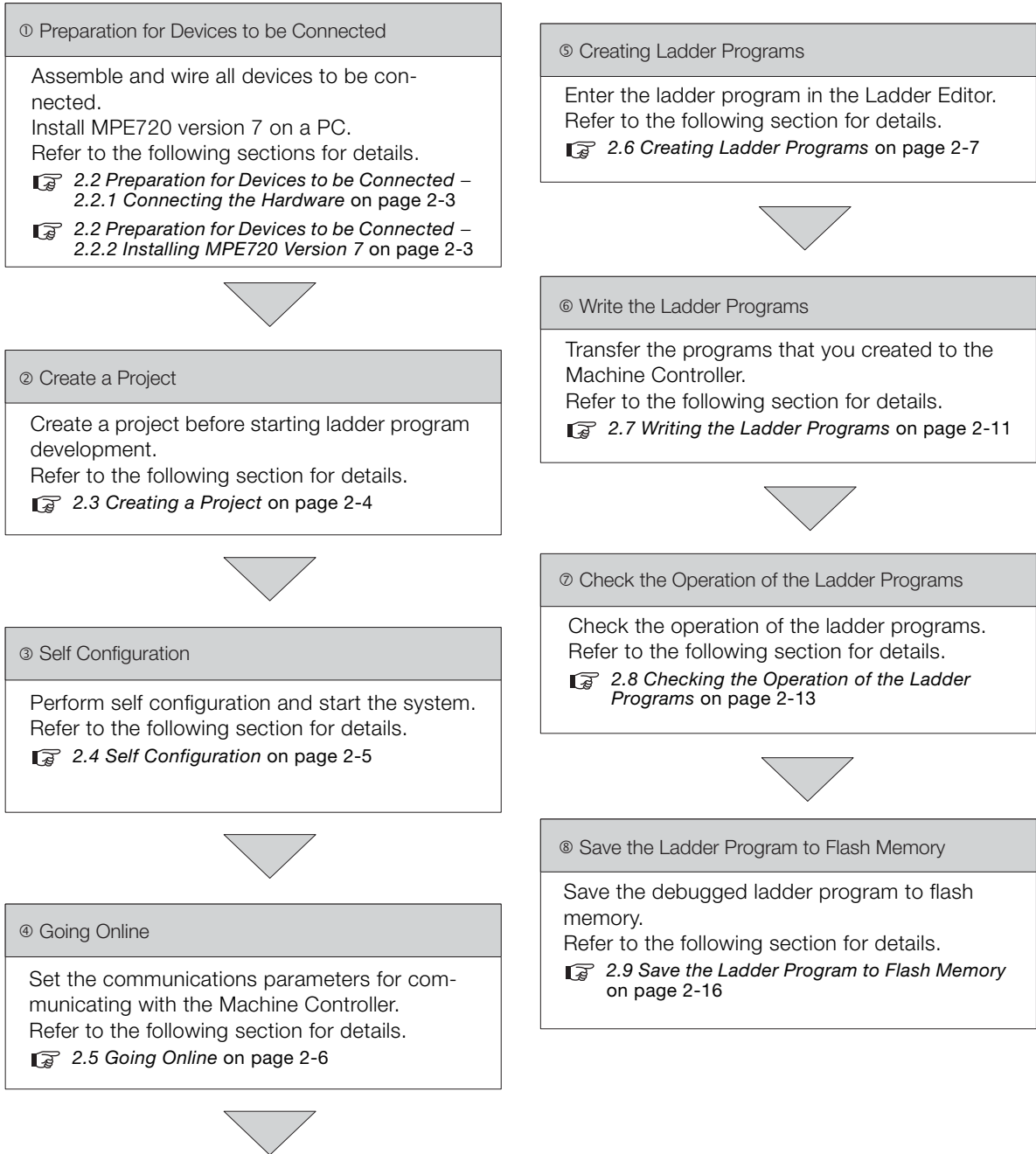
2

This chapter describes the development flow for ladder programs.

2.1	Introduction	2-2
2.2	Preparation for Devices to be Connected . .	2-3
2.2.1	Connecting the Hardware	2-3
2.2.2	Installing MPE720 Version 7	2-3
2.3	Creating a Project	2-4
2.4	Self Configuration	2-5
2.5	Going Online	2-6
2.6	Creating Ladder Programs	2-7
2.7	Writing the Ladder Programs	2-11
2.8	Checking the Operation of the Ladder Programs . .	2-13
2.8.1	Preparations for Checking Operation	2-13
2.8.2	Confirming the Operation of the 0000th Line (AND Circuit)	2-14
2.8.3	Confirming the Operation of the 0001st Line (Timer Circuit)	2-15
2.9	Save the Ladder Program to Flash Memory . .	2-16

2.1 Introduction

This section describes the flow for developing ladder programs as outlined below.



Note: The above flowchart is an example of the ladder program development process. Settings to interface the external devices must be completed to use programs on the actual system.

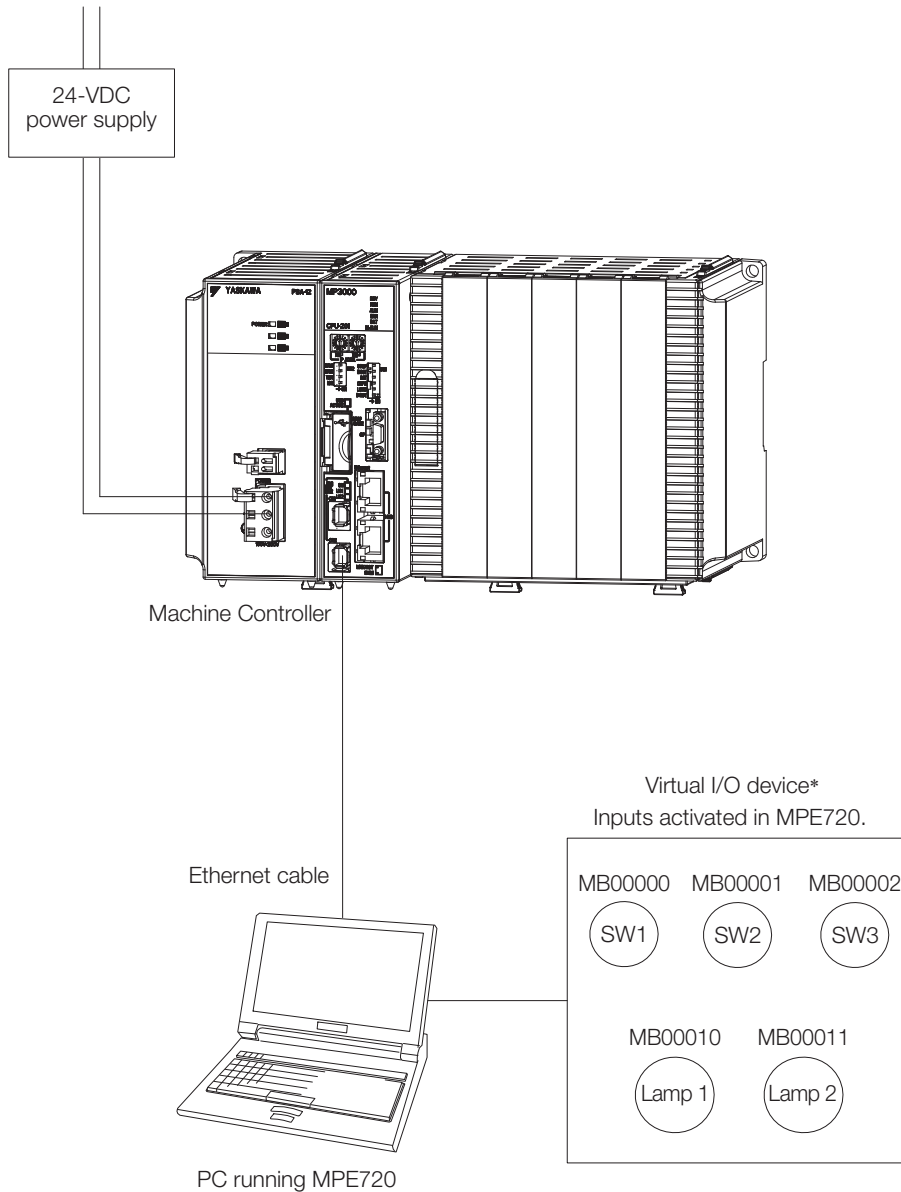
2.2 Preparation for Devices to be Connected

This section describes the hardware connections and the installation of MPE720 version 7.

2.2.1 Connecting the Hardware

Assemble and wire all devices to be connected.

The hardware connections are described using the system configuration shown below.



* In this example, M registers in the Machine Controller are used to simulate a virtual I/O device. In practice, the input and output signals would be connected to I/O Modules on the Machine Controller, and the ladder program would be created using I and O registers.

2.2.2 Installing MPE720 Version 7

Install MPE720 version 7 on a PC.

Refer to the following manual for the installation procedure.

📖 MP3000 Series Machine Controller System Setup Manual (Manual No.: SIEP C880725 00)

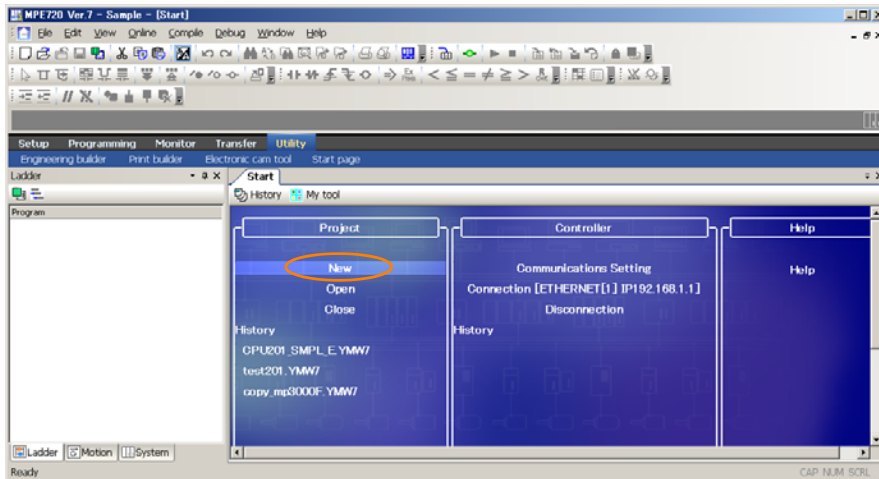
2.3 Creating a Project

Use the following procedure to create a project.

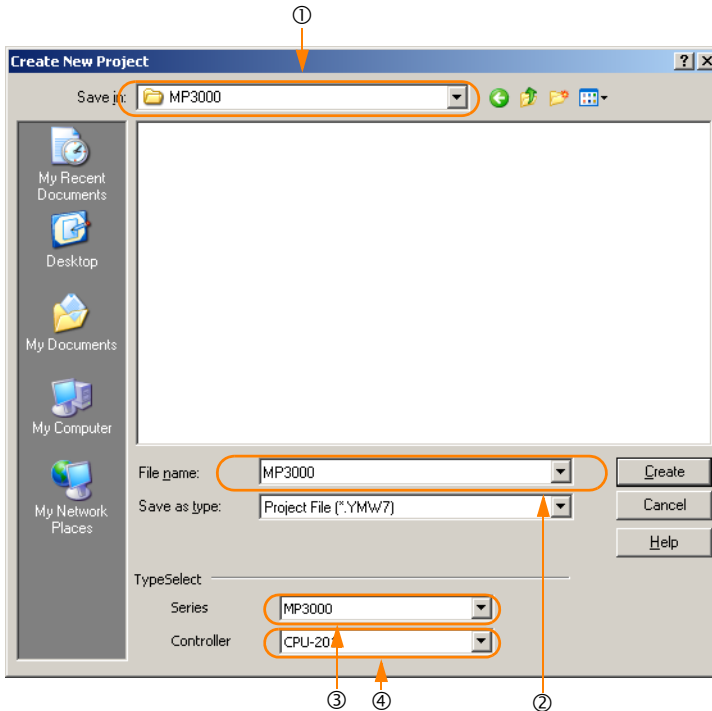
1. Double-click the icon shown below on the PC desktop to start MPE720 version 7.



2. Select **New** on the Start Tab Page.



3. Specify the file name, file storage location, Machine Controller series, and model.



- ① Specify the destination folder in the **Save in** Box.
- ② Enter the file name in the **File name** Box.
- ③ Select the applicable series in the **Series** Box.
- ④ Select the applicable model in the **Controller** Box.

4. Click the **Create** Button.

2.4 Self Configuration

Set up the system by performing self configuration. Self configuration automatically recognizes the Modules that are installed in the Machine Controller and the devices that are connected through the MECHATROLINK connector. This allows you to quickly and easily set up the system. You can perform self configuration by using the DIP switch on the Machine Controller or by using the MPE720.


Refer to the following manual for details on self configuration.

- 📖 MP3000 Series MP3200 Product Manual (Manual No.: SIEP C880725 10)
- 📖 MP3000 Series MP3300 Product Manual (Manual No.: SIEP C880725 21)
- 📖 MP3000 Series MP3100 Product Manual (Manual No.: SIEP C880725 24)

2.5 Going Online

Set the communications parameters to perform communications between the Machine Controller and PC.

Refer to the following manual for the procedure to set up communications.

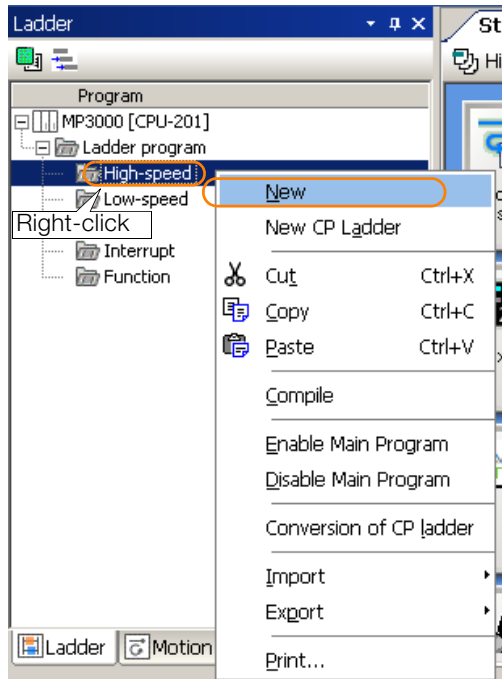
 MP3000 Series Machine Controller System Setup Manual (Manual No.: SIEP C880725 00)

2.6 Creating Ladder Programs

Use the following procedure to create a ladder program.

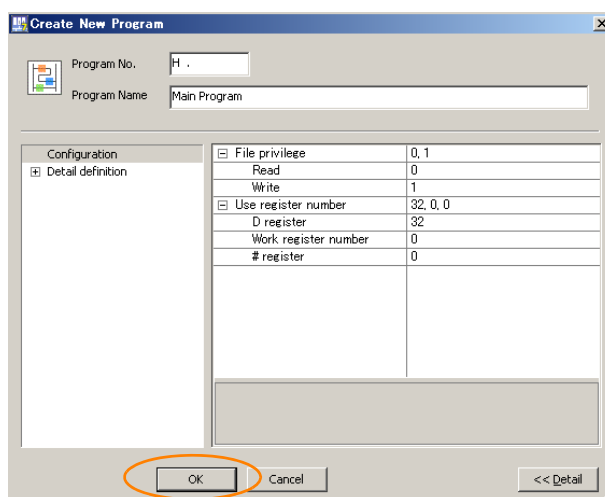
Information The following example shows how to create a high-speed program, but low-speed and startup programs can be created in essentially the same way.

1. Select **Programming – Ladder program** from the Launcher. The Ladder Pane will be displayed.
2. Right-click **High-speed** under **Ladder program**, and select **New**.




The Create New Program Dialog Box will be displayed.


3. Click the OK Button.



The Ladder Editor will start.

4. Enter the ladder program. Ladder programs are entered by inserting rungs, then instructions, and finally parameters for the instructions. Refer to the following section for details.
 *Ladder Program Creation Example* on page 2-8

- While displaying the ladder program, select **Compile – Compile** from the menu bar to compile the program.
When the compilation is finished, the ladder program will be saved automatically.

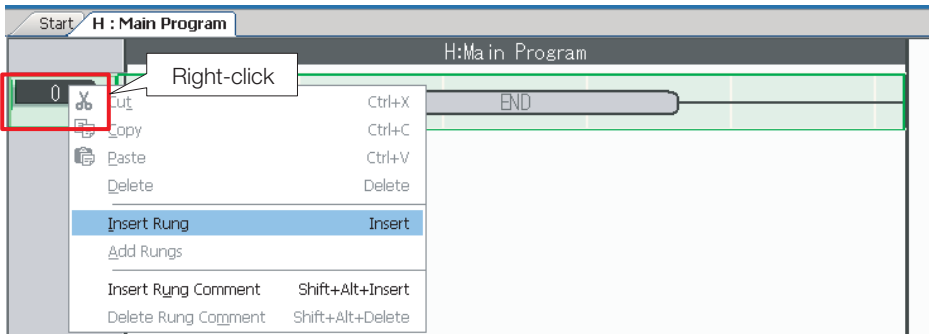


Important If an error is displayed in the Output Pane during compilation, the ladder program will not be saved.

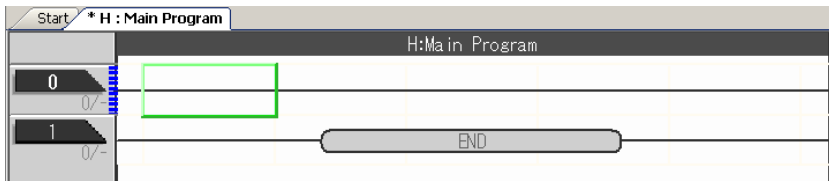
Ladder Program Creation Example

The following example shows how to insert an NOC instruction.

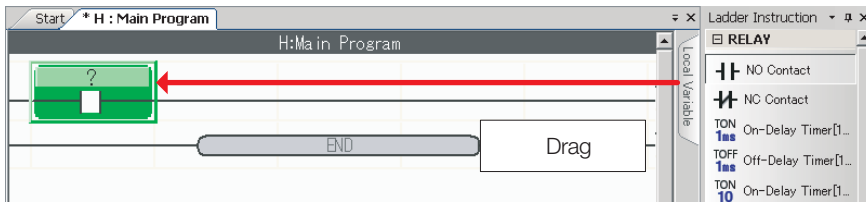
- Right-click the tab with the row number, and select **Insert Rung**.



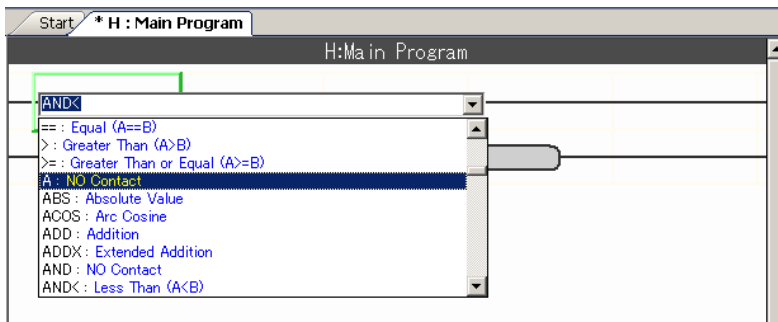
A rung will be inserted.




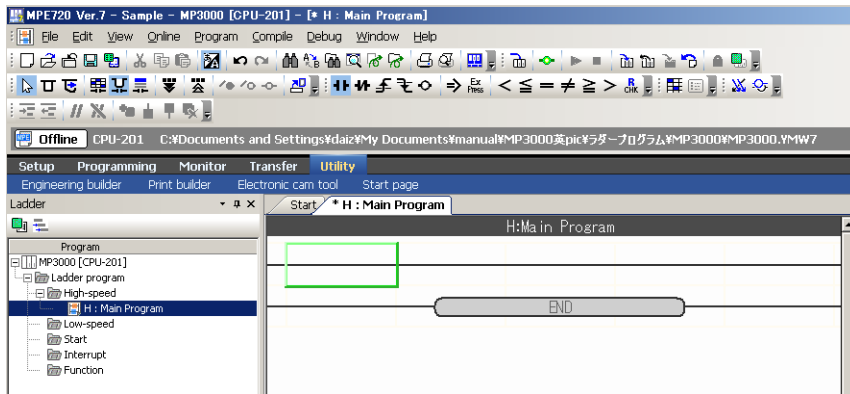
- Create the NOC instruction with one of the following methods.
 - Drag **NO Contact** under **RELAY** in the Task Pane to the inserted rung.



- Double-click at the location at which to insert the NOC instruction, and select **A: NO Contact** from the list.



- Select the location at which to insert the NOC instruction, and click the NOC Instruction Button .

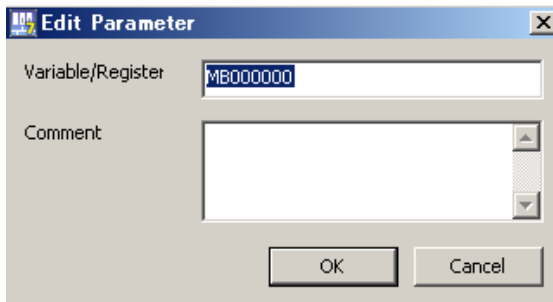


3. Double-click the box with a question mark.



The Edit Parameter Dialog Box will be displayed.


4. Enter “MB000000” in the Variable/Register Box and click the OK Button.



“MB000000” will be displayed for the NOC instruction.



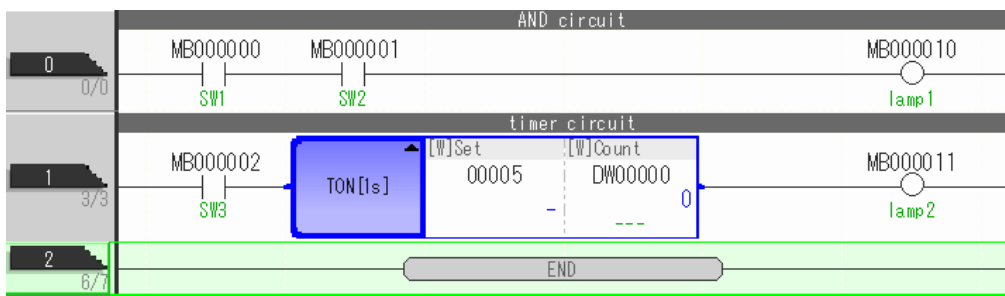
Note: The types of registers and data you can use depend on the actual instruction. Refer to the following chapter for details on the different types of instructions.

 [Chapter 4 Ladder Language Instructions](#)

Information To insert a comment, right-click the tab with the row number, and select **Insert Rung Comment**.

5. Repeat steps 1 to 4 until you have entered the entire ladder program. The following example shows a ladder program and its timing chart.

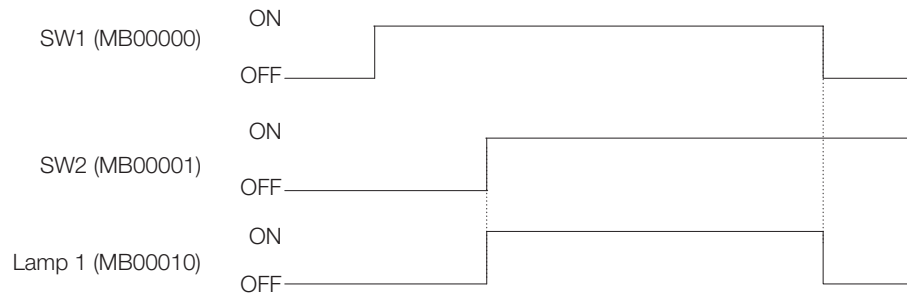
<Ladder Programming Example>



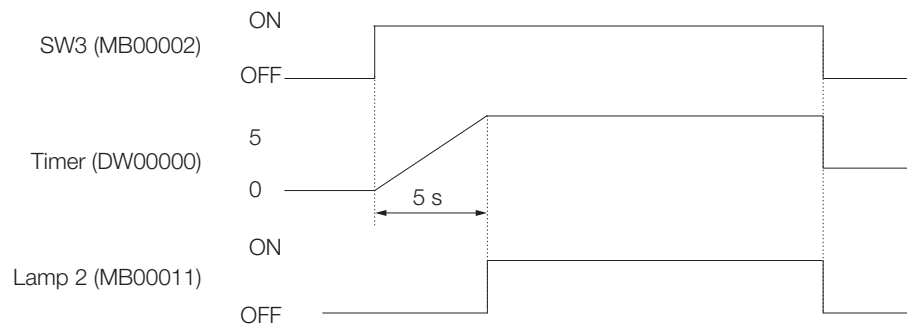
Note: The ladder programming example that is shown above uses M registers for switches and lamps. When you enter a ladder program for an actual system, use the appropriate I and O registers.

<Timing Chart Example>

AND Circuit Operation



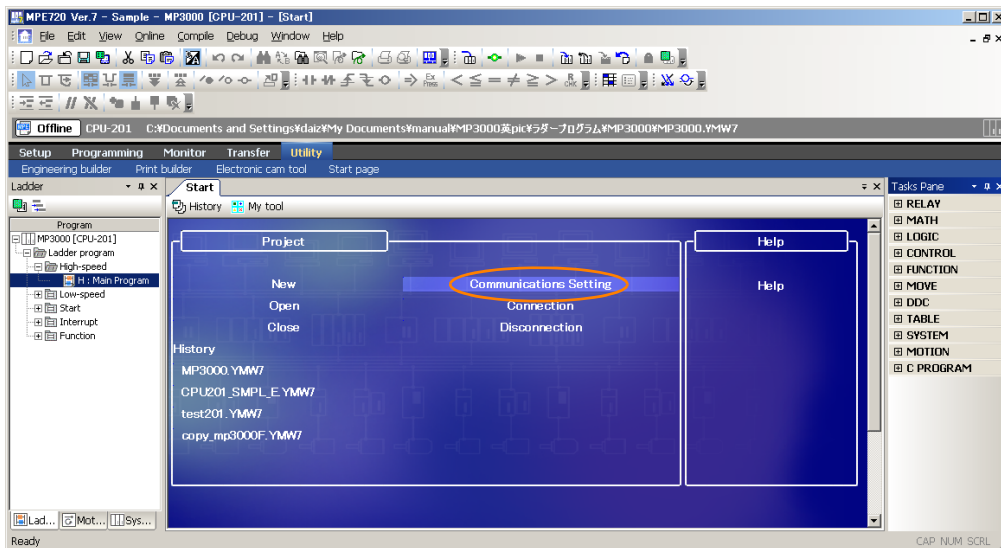
Timer Circuit Operation



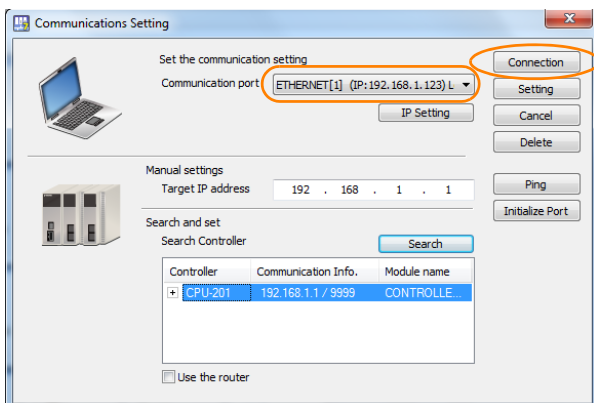
2.7 Writing the Ladder Programs

Use the following procedure to transfer the ladder program to the Machine Controller. This procedure is not necessary if you created the ladder program online.

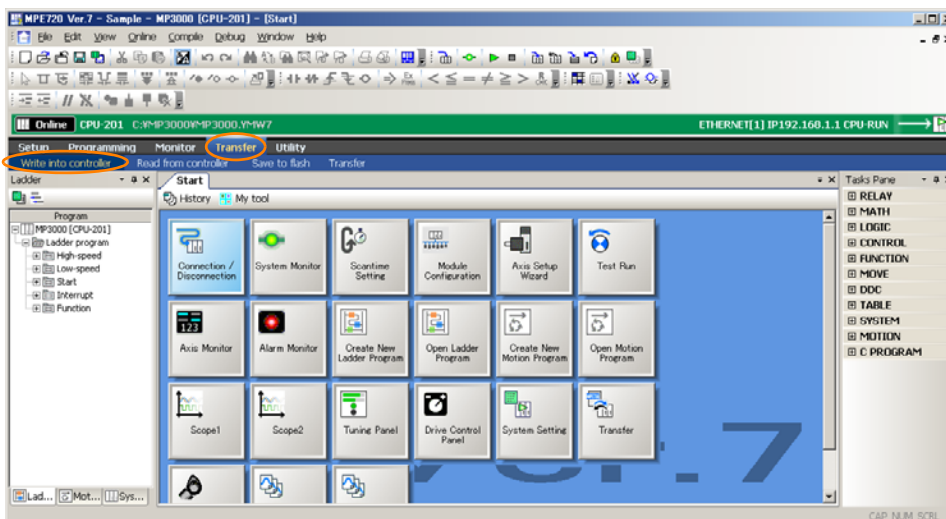
1. Click **Communications Setting** on the Start Tab Page.



2. Select the desired communications port in the **Communication Port Box** on the **Communications Setting** Dialog Box. Click the **Connection** Button.

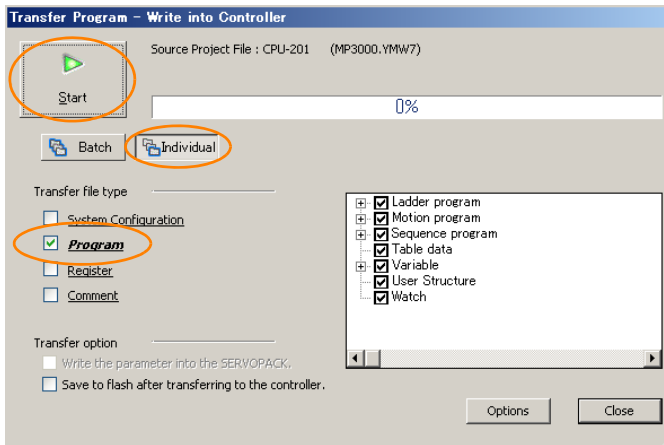


3. Select **Transfer – Write into controller** from the Launcher.



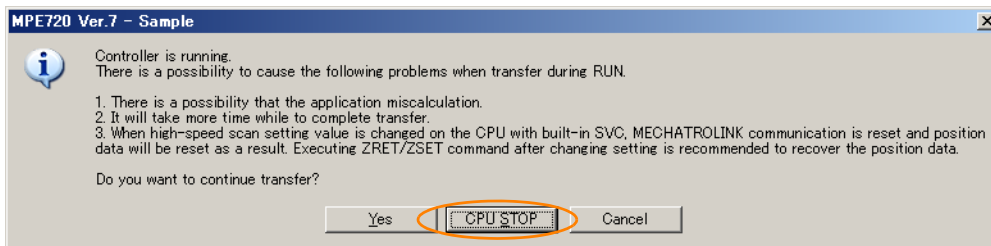
Ladder Program Development Flow

- Click the **Individual** Button, then select only the **Program** Check Box. Click the **Start** Button.



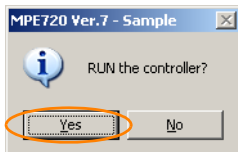
Note: 1. When an individual transfer is selected, the same file in the Machine Controller will be overwritten with the selected project file data.
 2. When a batch transfer is selected, the Machine Controller's RAM will be cleared before the transfer, and all project file data will be written in the RAM.

- Click the **CPU STOP** Button.



The transfer will start.

- Click the **Yes** Button in the following dialog box.



The Machine Controller will switch to RUN Mode.

2.8 Checking the Operation of the Ladder Programs

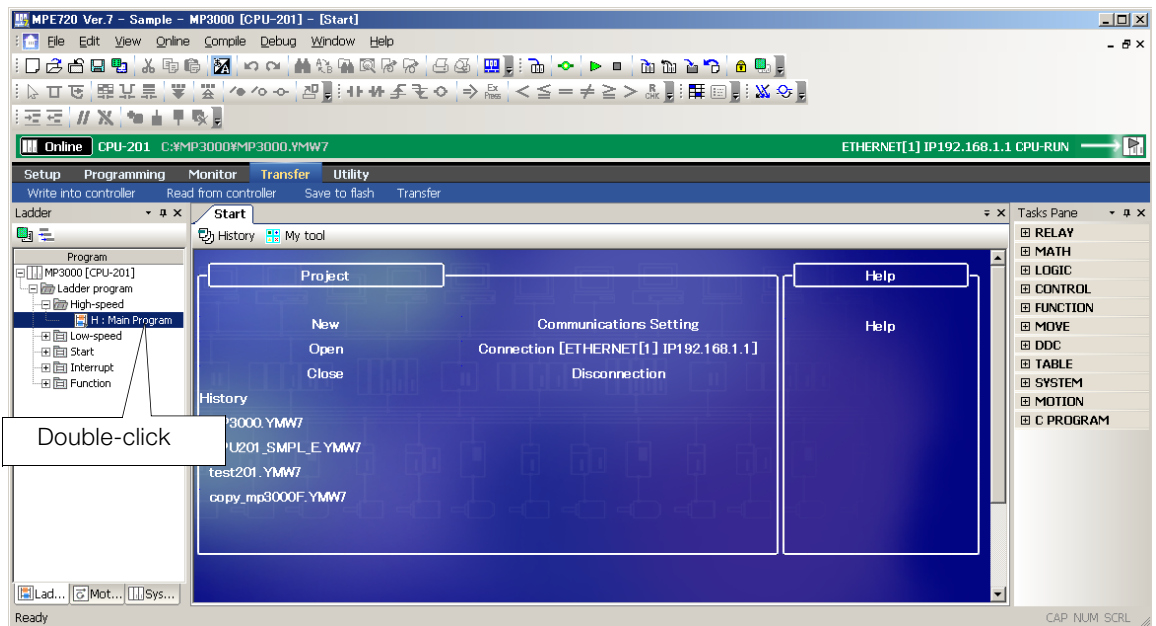
Use the following procedure to confirm the operation of your ladder program.

Confirm that your program operates correctly by manipulating registers with the Register List, and by checking the runtime monitor in the Register List and Ladder Editor.

2.8.1 Preparations for Checking Operation

Use the following procedure to prepare to check the operation of your program.

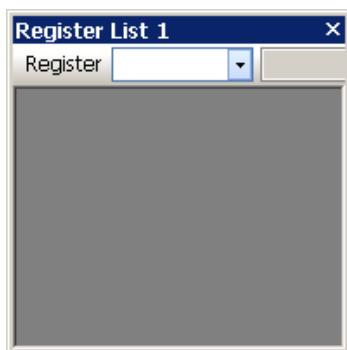
1. Double-click the target ladder program in the Ladder Pane.



2. Click the Register List 1 Tab.
The Register List 1 Dialog Box will be displayed.

Information If the **Register List 1** Tab is not visible, display the Register List 1 Dialog Box by performing one of these steps.

- Select **View – Register List – Register List 1** from the menu bar.
- Select **Monitor – Register List** from the Launcher.



3. Enter “MB000000” in the Register Box.
The register list will expand as shown below.

Register	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
MB000000	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
MB000010	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
MB000020	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
MB000030	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
MB000040	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
MB000050	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
MB000060	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
MB000070	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

2.8.2 Confirming the Operation of the 0000th Line (AND Circuit)

Use the following procedure to check the operation of the 0000th line.

1. Set MB000000 to ON in the Register List. Confirm that the NO contact for MB000000 in the Ladder Editor changes to blue.

Note: When a coil or contact is highlighted in blue, it means that it is ON.

Confirm that the contact changes to blue.

Input ON.

2. Set MB000001 to ON in the Register List. Confirm the following points.
 - In the Ladder Editor, the NO contact for MB000001 and coil for MB000010 must be blue.
 - In the Register List, MB000010 must be ON.

Confirm that the contact changes to blue.

Confirm that the coil changes to blue.

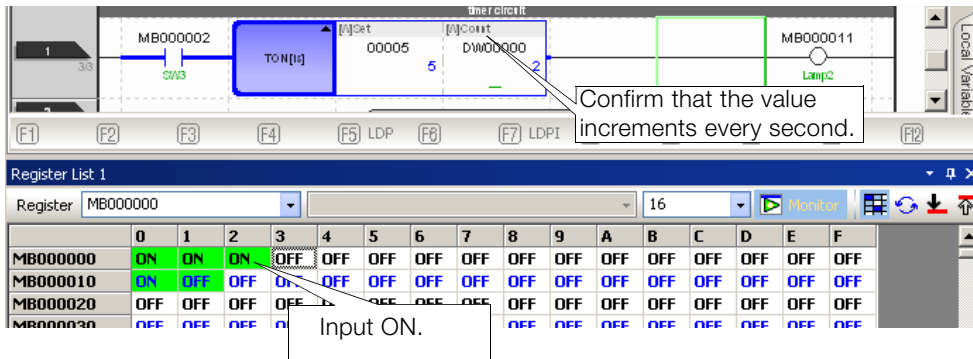
Input ON.

Confirm that the register is ON.

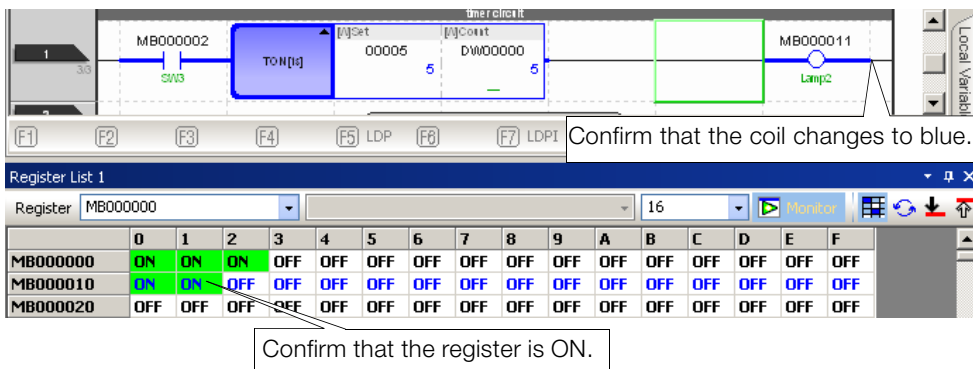
2.8.3 Confirming the Operation of the 0001st Line (Timer Circuit)

Use the following procedure to check the operation of the 0001st line.
Set MB000002 to ON in the Register List. Confirm the following points.

- The DW00000 timer must increment every second.



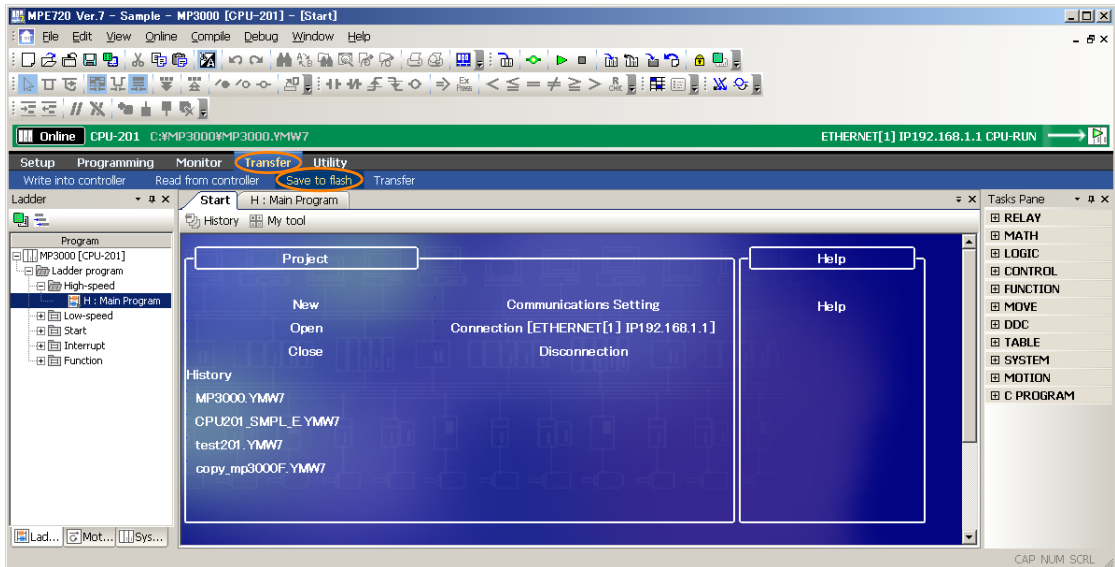
- After five seconds, the coil for MB000011 must turn blue in the Ladder Editor.
- In the Register List, MB000011 must be ON.



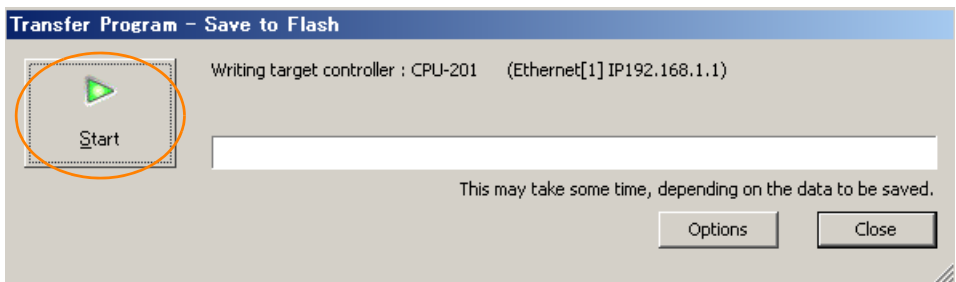
2.9 Save the Ladder Program to Flash Memory

Use the following procedure to save the Machine Controller RAM data to the flash memory in the Machine Controller.

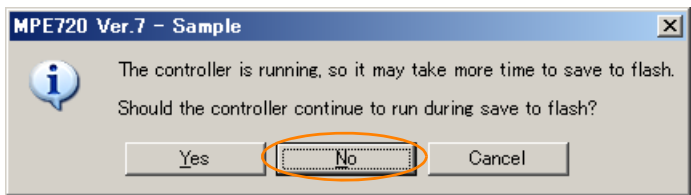
1. Select **Transfer – Save to Flash** from the Launcher.



2. Click the **Start** Button.

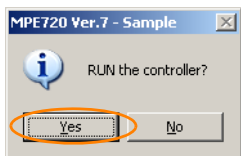


3. Click the **No** Button.




The MPE720 begins saving the data to flash memory.

4. Click the **Yes** Button.



The Machine Controller will switch to RUN Mode.



Important Make sure to save the data to flash memory after writing it to the Machine Controller's RAM. Failure to save the data to flash memory will result in losing data when the power is turned OFF and ON again, causing the Machine Controller to run on the data that was last saved in the flash memory.

Registers

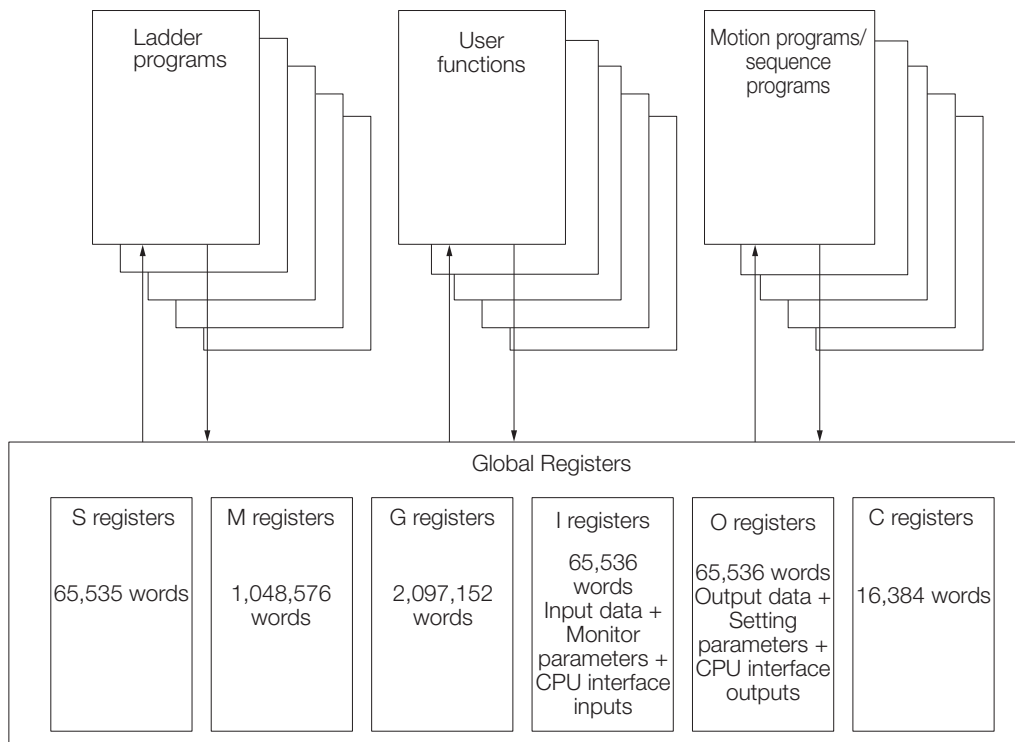
3

Registers are areas that store data within the Machine Controller. Variables are registers with labels (variable names). This chapter describes registers.

3.1	Global Registers	3-2
3.2	Local Registers	3-4
3.2.1	Precautions When Using Local Registers within a User Function	3-5
3.2.2	Setting the D Register Clear When Start Option	3-6
3.2.3	Setting for D Registers	3-7
3.3	Structure of Register Addresses	3-8
3.3.1	Register Types	3-8
3.3.2	Data Types	3-8
3.4	Index Registers (i, j)	3-12
3.5	Array Registers ([])	3-14

3.1 Global Registers

Global registers are shared by ladder programs, user functions, motion programs, and sequence programs. Memory space for global registers is reserved by the system for each register type.



The following table gives details about global registers.

Type	Name	Designation Method	Usable Range	Description
S	System registers (S registers)	SBnnnnnh, SWnnnnn, SLnnnnn, SQnnnnn, SFnnnnn, SDnnnnn, SAnnnnn	SW00000 to SW65534	These registers are prepared by the system. They report the status of the Machine Controller and other information. The system clears the registers from SW00000 to SW00049 to 0 at startup. They have a battery backup.
M	Data registers (M registers)	MBnnnnnnnh, MWnnnnnnn, MLnnnnnnn, MQnnnnnnn, MFnnnnnnn, MDnnnnnnn, MAnnnnnnn	MW0000000 to MW1048575	These registers are used as interfaces between programs. They have a battery backup.
G	G registers	GBnnnnnnnh, GWnnnnnnn, GLnnnnnnn, GQnnnnnnn, GFnnnnnnn, GDnnnnnnn, GAnnnnnnn	GW0000000 to GW2097151	These registers are used as interfaces between programs. They do not have a battery backup.

Continued on next page.

Continued from previous page.

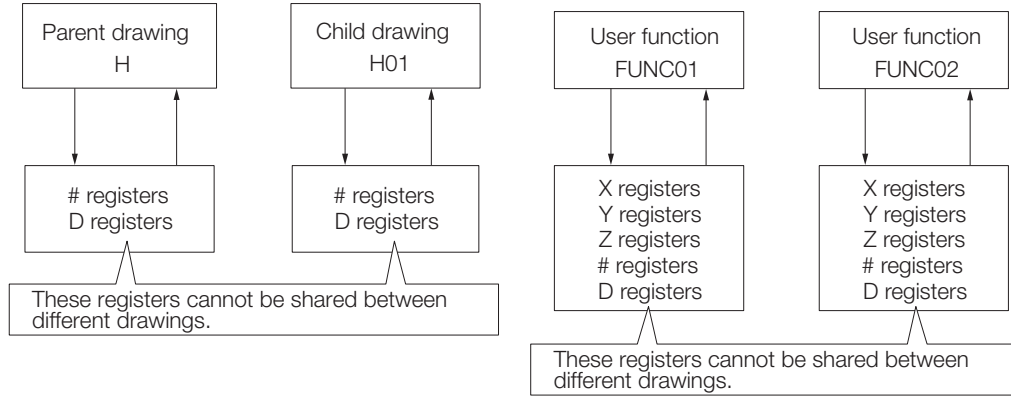
Type	Name	Designation Method	Usable Range	Description
I	Input registers (I registers)	IBhhhhhh, IWhhhhhh, ILhhhhhh, IQhhhhhh, IFhhhhhh, IDhhhhhh, IAhhhhhh,	IW00000 to IW07FFF, IW10000 to IW17FFF	These registers are used for input data.
			IW08000 to IW0FFFF, IW18000 to IW1FFFF	These registers store the motion monitor parameters. These registers are used for Motion Modules.
			IW20000 to IW23FFF	These registers are used for CPU interface input data.
O	Output registers (O registers)	OBhhhhhh, OWhhhhhh, OFhhhhhh, OQhhhhhh, OFhhhhhh, ODhhhhhh, OAhhhhhh,	OW00000 to OW07FFF, OW10000 to OW17FFF	These registers are used for output data.
			OW08000 to OW0FFFF, OW18000 to OW1FFFF	These registers store the motion setting parameters. These registers are used for Motion Modules.
			OW20000 to OW23FFF	These registers are used for CPU interface output data.
C	Constant registers (C registers)	CBnnnnnh, CWnnnnn, CLnnnnn, CQnnnnn, CFnnnnn, CDnnnnn, CAnnnnn	CW00000 to CW16383	These registers can be read in programs but they cannot be written. The values are set from the MPE720.

Note: n: decimal digit, h: hexadecimal digit


3.2 Local Registers

Local registers can be used within a specific drawing. They cannot be used in other drawings.

<Ladder Program Conceptual Diagram>



The following table gives details about local registers.

Type	Name	Designation Method	Usable Range	Description	Features
#	# registers	#Bnnnnnh, #Wnnnnn, #Lnnnnn, #Qnnnnn, #Fnnnnn, #Dnnnnn, #Annnnn	#W00000 to #W16383	These registers can be read in programs but they cannot be written. The values are set from the MPE720.	Program-specific
D	D registers	DBnnnnnh, DWnnnnn, DLnnnnn, DQnnnnn, DFnnnnn, DDnnnnn, DAnnnnn	DW00000 to DW16383	These registers can be used for general purposes within a program. By default, 32 words are reserved for each program. The default value after startup depends on the setting of the D Register Clear when Start Option. Refer to the following section for details.  3.3.2 Data Types on page 3-8	
X	Function input registers	XBnnnnnh, XWnnnnn, XLnnnnn, XQnnnnn, XFnnnnn, XDnnnnn	XW00000 to XW00016	These registers are used for inputs to functions. <ul style="list-style-type: none"> • Bit inputs: XB000000 to XB00000F • Integer inputs: XW00001 to XW00016 • Double-length integers: XL00001 to XL00015 • Quadruple-length integers: XQ00001 to XQ00013 • Real numbers: XF00001 to XF00015 • Double-precision real numbers: XD00001 to XD00013 	Function-specific
Y	Function output registers	YBnnnnnh, YWnnnnn, YLnnnnn, YQnnnnn, YFnnnnn, YDnnnnn	YW00000 to YW00016	These registers are used for outputs from functions. <ul style="list-style-type: none"> • Bit outputs: YB000000 to YB00000F • Integer outputs: YW00001 to YW00016 • Double-length integers: YL00001 to YL00015 • Quadruple-length integers: YQ00001 to YQ00013 • Real numbers: YF00001 to YF00015 • Double-precision real numbers: YD00001 to YD00013 	

Continued on next page.

Continued from previous page.

Type	Name	Designation Method	Usable Range	Description	Features
Z	Function internal registers	ZBnnnnnh, ZWnnnnn, ZLnnnnn, ZQnnnnn, ZFnnnnn, ZDnnnnn	ZW00000 to ZW00016	These are internal registers that are unique within each function. You can use them for internal processing in functions. <ul style="list-style-type: none"> • Bits: ZB000000 to ZB00063F • Integers: ZW00000 to ZW00063 • Double-length integers: ZL00000 to ZL00062 • Quadruple-length integers: ZQ00000 to ZQ00060 • Real numbers: ZF00000 to ZF00062 • Double-precision real numbers: ZD00000 to ZD00060 	Function-specific
A	Function external registers	ABnnnnnh, AWnnnnn, ALnnnnn, AQnnnnn, AFnnnnn, ADnnnnn	0 to 2097152	These are external registers that use the address input value as the base address. When the address input value of an M or D register is provided by the source of the function call, then the registers of the source of the function call can be accessed from inside the function by using that address as the base.	


Note: n: decimal digit, h: hexadecimal digit



User functions can be called from any programs, any number of times.

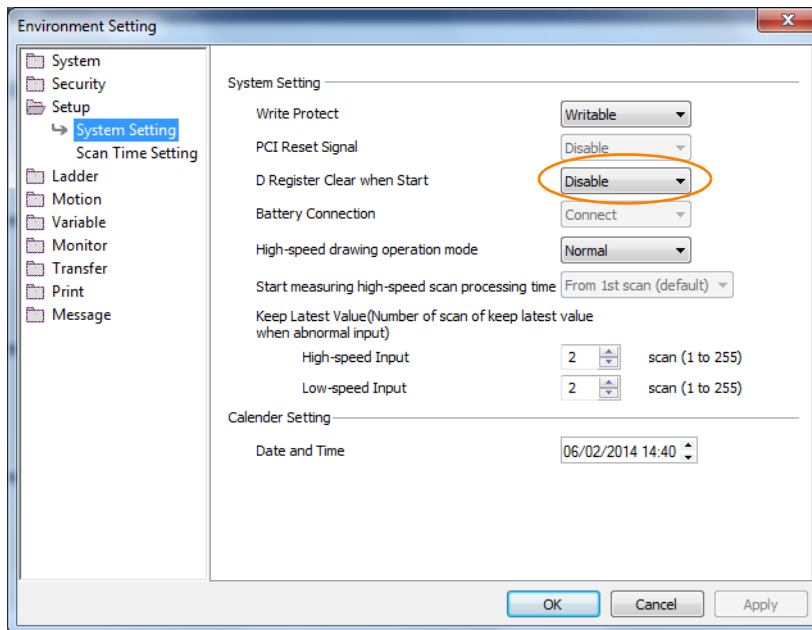
3.2.1 Precautions When Using Local Registers within a User Function

When you call a user function, consider what values could be in the local registers, and perform initialization as needed.

Name	Precautions
X registers (function input registers)	If input values are not set, the values will be uncertain. Do not use X registers that are outside of the range that is specified in the input definitions.
Y registers (function output registers)	If output values are not set, the values will be uncertain. Always set the values of the range of Y registers that is specified in the output definitions.
Z registers (function internal registers)	When the function is called, the previously set values will be lost and the values will be uncertain. These registers are not appropriate for instructions if the previous value must be retained. Use them only after initializing them within the function.
# registers	These are constant registers. Their values cannot be changed.
D registers	When the function is called, the previously set values are preserved. If a previous value is not necessary, initialize the value, or use a Z register instead. D registers retain the data until the power is turned OFF. The default value after startup depends on the setting of the D Register Clear when Start Option. Refer to the following section for details.  3.3.2 Data Types on page 3-8

3.2.2 Setting the D Register Clear When Start Option

1. Select *File – Environment Setting* from the MPE720 Version 7 Window.
2. Select *Setup – System Setting*.
3. Select **Enable** or **Disable** for the **D Register Clear when Start** Option.
Disable: The initial values will be uncertain.
Enable: The initial values will be 0.



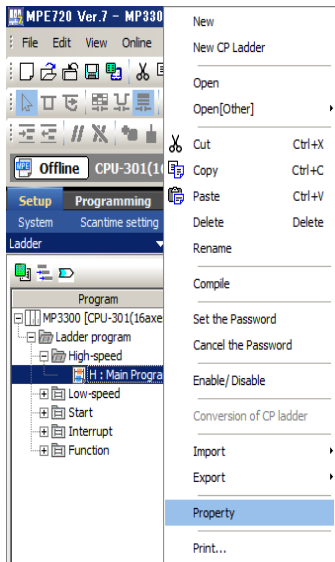
3.2.3 Setting for D Registers

Specify the range of registers that will be used on each drawing in the Program Property Dialog Box.

The default setting for D registers is 32 words for one drawing, but this can be extended to a maximum of 16,384 words.

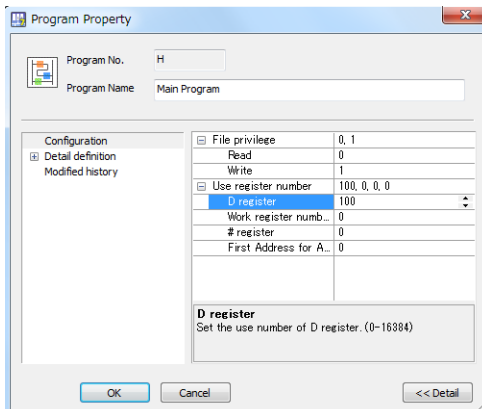
Use the following procedure to extend the range of D registers.

1. Right-click the drawing in the Ladder Pane and select **Property**.



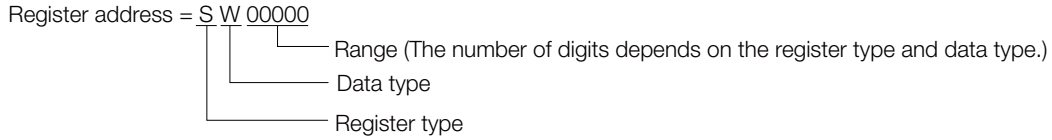
The Program Property Dialog Box will be displayed.

2. Change the range of D registers to 100 and click the **OK** Button.



This concludes extending the range of D registers.

3.3 Structure of Register Addresses



Information You can also use index registers or array registers as variables to address specific registers. Refer to the following section for details.

- 📖 3.4 Index Registers (i, j) on page 3-12
- 📖 3.5 Array Registers (I) on page 3-14

3.3.1 Register Types

Refer to the following section for the types of registers.

- 📖 3.1 Global Registers on page 3-2
- 📖 3.2 Local Registers on page 3-4


3.3.2 Data Types

There are various data types that you can use depending on the purpose of the application: bit, integer, double-length integer, quadruple-length integer, real number, double-precision real number, and address.

Symbol	Data Type	Range of Values	Data Size	Description
B	Bit	1 (ON) or 0 (OFF)	–	Used in relay circuits and to determine ON/OFF status.
W	Integer	-32,768 to 32,767 (8000 to 7FFF hex)	1 word	Used for numeric operations. The values in parenthesis on the left are for logical operations.
L	Double-length integer	-2,147,483,648 to 2,147,483,647 (80000000 to 7FFFFFFF hex)	2 words	Used for numeric operations. The values in parenthesis on the left are for logical operations.
Q	Quadruple-length integer*1	-9223372036854775808 to 9223372036854775807 (8000000000000000 to 7FFFFFFFFFFFFFFF hex)	4 words	Used for numeric operations. The values in parenthesis on the left are for logical operations.
F	Real number	± (1.175E-38 to 3.402E+38) or 0	2 words	Used for advanced numeric operations. *2
D	Double-precision real number*1	± (2.225E-308 to 1.798E+308) or 0	4 words	Used for advanced numeric operations. *2
A	Address	0 to 2,097,152	–	Used only as pointers for addressing.

*1. These data types cannot be used for indirect designation of motion programs.

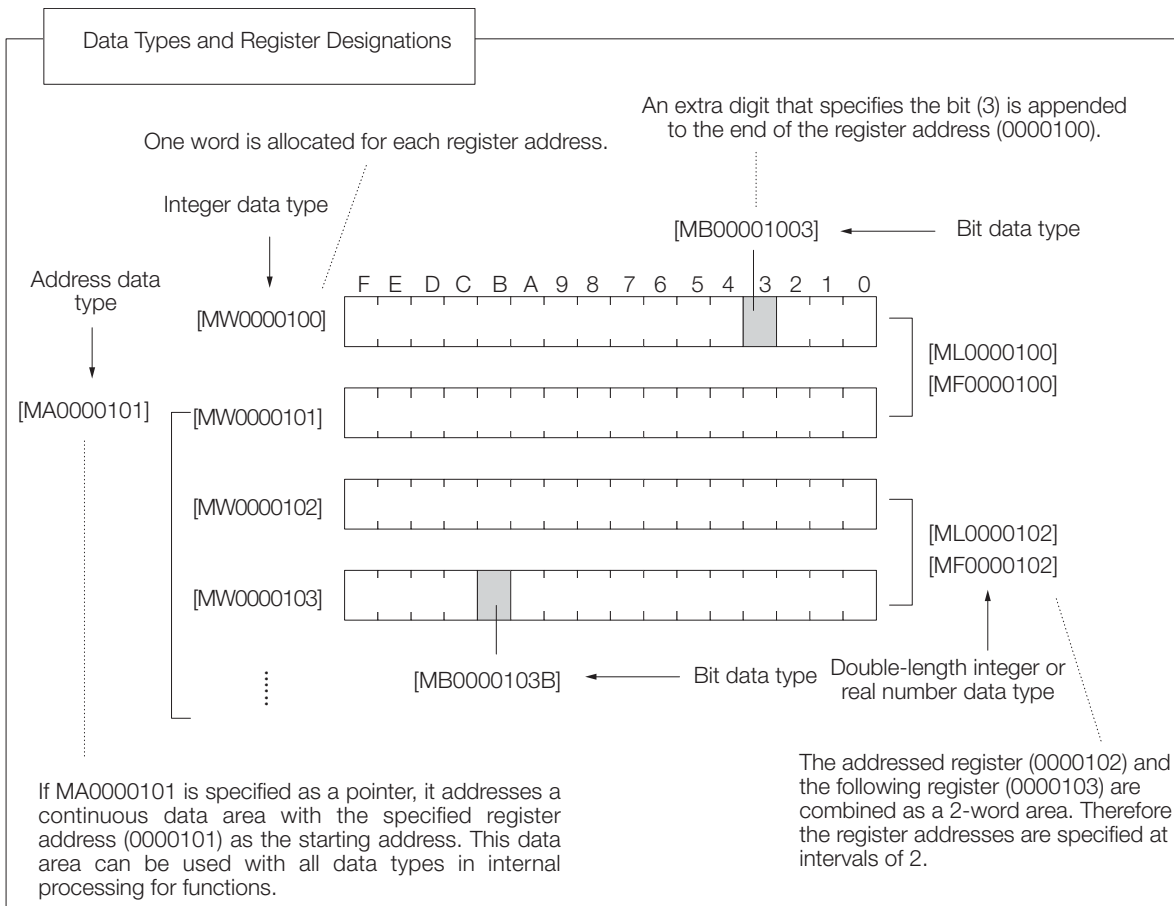
*2. Conforms to IEEE754 standards.



Important

The MP3000-series Machine Controller does not have separate registers for each data type. As shown in the following figure, the same address will access the same register even if the data type is different.

For example, MB00001003, a bit address, and the MW0000100, an integer address, have different data types, but they both access the same register, MW0000100.



Pointer Designation

When an address is passed to a function as a parameter, this is referred to as pointer designation.

Term

When pointer designation is used, the continuous data area starting from the address of the specified register address can be used in internal processing for functions with all data types.

Precautions for Operations Using Different Data Types

If you perform an operation using different data types, be aware that the results will be different depending on the data type of the storage register, as described below.

- **Storing Real Number Data in an Integer Register**

<When Numbers Are Truncated After the Decimal Point>


MW0000100 = MF0000200: The real number data is converted to integer data and stored in
(1) (1.5678) the destination register.

<When Numbers Are Rounded Off>

MW00100 = MF000200
(2) (1.5678)

MW100 = MF000200
(-2) (-1.5678)

Note: There may be rounding error due to storing a real number in an integer register.
Whether numbers are rounded or truncated when converting a real number to an integer can be set in the properties of the drawing.

 *Setting for Real Number Casting on page 3-11*

MW0000100 = MF0000200 + MF0000202: The result of the operation may be different
(0124) (123.48) (0.02) depending on the value of the variable.
(0123) (123.49) (0.01)

- **Storing Real Number Data in a Double-length Integer Register**

ML0000100 = MF0000200: The real number data is converted to integer data and stored in
(65432) (65432.1) the destination register.

- **Storing Double-length Integer Data in an Integer Register**

MW0000100 = ML0000200: The lower 16 bits of the double-length integer data are stored
(-00001) (65535) without change.

- **Storing Integer Data in a Double-length Integer Register**

ML0000100 = MW0000200: The integer data is converted to double-length integer data and
(0001234) (1234) stored in the destination register.

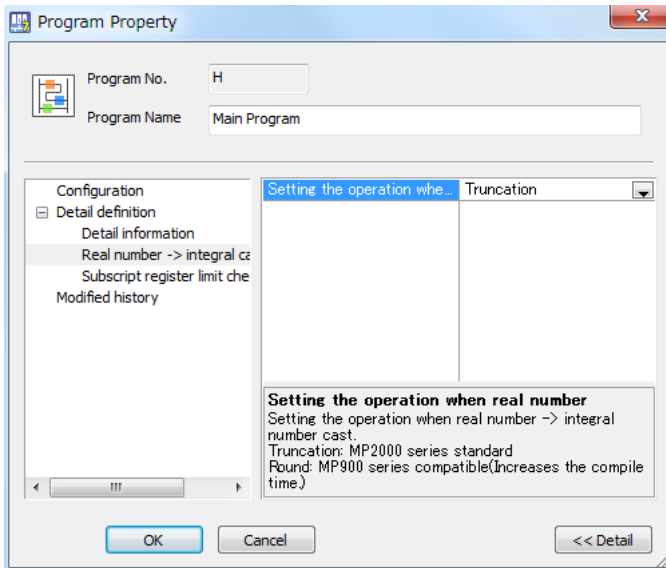
Setting for Real Number Casting

The casting method (truncating or rounding) can be set in the detailed definitions in the Program Property Dialog Box.

The method to use for real number casting is set for each drawing.

Use the following procedure to display the Program Property Dialog Box.

1. In the Ladder Pane, select the ladder program for which to view the properties.
2. Right-click the selected program and select **Property** from the pop-up menu. The Program Property Dialog Box will be displayed.

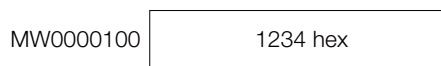


Information The data is little endian, as shown in the following example.

- MB00001006



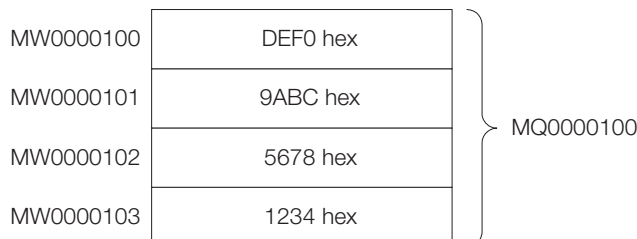
- MW0000100 = 1234 hex



- ML0000100 = 12345678 hex

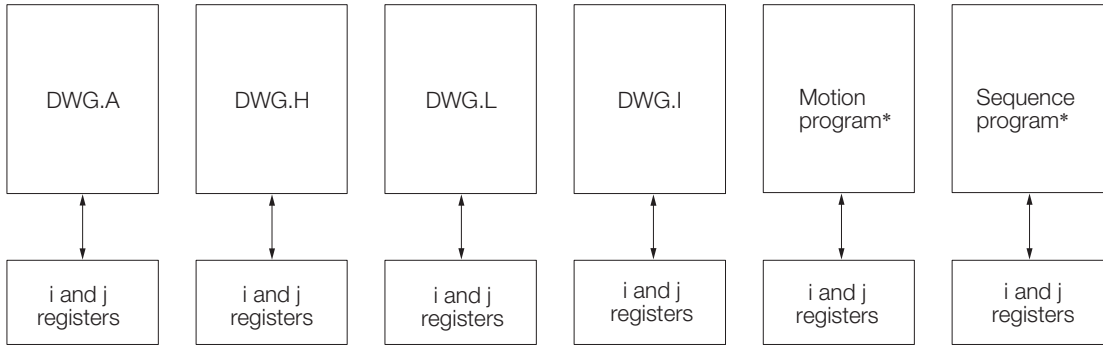


- MQ0000100 = 123456789ABCDEF0 hex



3.4 Index Registers (i, j)

There are two special registers, i and j, that are used to modify relay and register addresses. The functions of i and j are identical. They are used to handle register addresses like variables. There are index registers for each program type, as shown in the following figure.



* Motion programs and sequence programs have separate i and j registers for each task.

Note: Functions reference the i and j registers that belong to the calling drawing. For example, a function called by DWG.H will reference the i and j registers for DWG.H.

We will now describe how an index register behaves using examples for each register data type.

- Attaching an Index to a Bit Register

Using an index is the same as adding the value of i or j to the register address. For example, if i = 2, MB00000000i is the same as MB00000002.

i = 2;
 DB000000 = MB00000000i; \longleftrightarrow DB000000 = MB00000002;

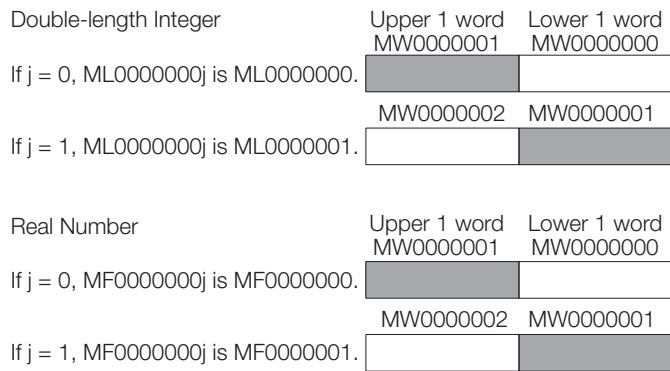
- Attaching an Index to an Integer Register

Using an index is the same as adding the value of i or j to the register address. For example, if j = 30, MW0000001j is the same as MW00000031.

j = 30;
 DW000000 = MW0000001j; \longleftrightarrow DW000000 = MW00000031;

- Attaching an Index to a Double-length Integer or a Real Number Register

Using an index is the same as adding the value of i or j to the register address. For example, if j = 1, ML0000000j is the same as ML00000001. Similarly, if j = 1, MF0000000j is the same as MF00000001.



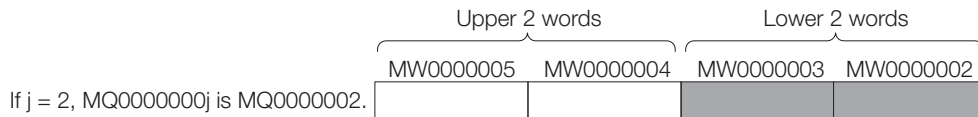
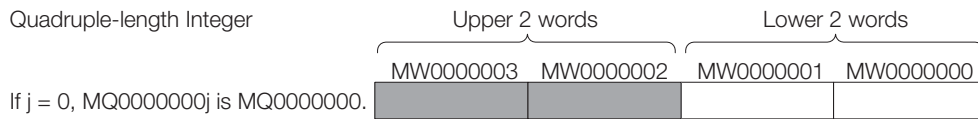
Information Double-length integers and real numbers use a region that is 2 words in size. For example, when using ML0000000j with both j = 0 and j = 1, the one-word area of MW0000001 will overlap. Be careful of overlapping areas when indexing double-length integer or real number register addresses.

- Attaching an Index to a Quadruple-length Integer or a Double-precision Real Number Register

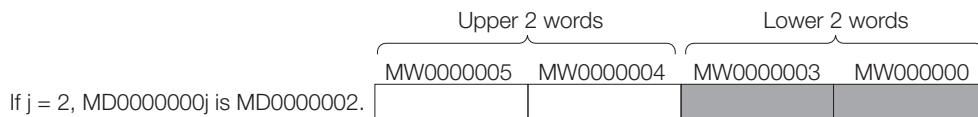
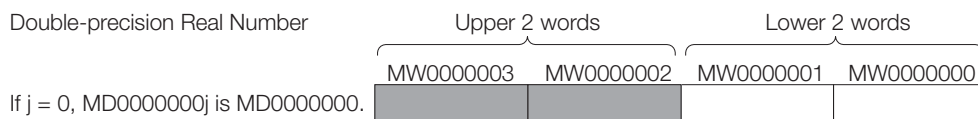
Using an index is the same as adding the value of i or j to the register address.

For example, if $j = 2$, MQ0000000j is the same as MQ0000002. Similarly, if $j = 2$, MD0000000j is the same as MD0000002.

Quadruple-length Integer



Double-precision Real Number



Information Quadruple-length integers and double-precision real numbers use a region that is 4 words in size. For example, when using MQ0000000j with both $j = 0$ and $j = 2$, the two-word area of MW00000002 and MW00000003 will overlap. Be careful of overlapping areas when indexing quadruple-length integer or double-precision real number register addresses.

3.5 Array Registers ([])

Array registers are used to modify register addresses, and are denoted by square brackets []. They are used to handle register addresses as variables.

Similarly to index registers, an offset is added to the register address.

- Attaching an Array Register to a Bit Register

Using an array register is the same as adding the value of the array register to the register address.

For example, if $DW00000 = 2$, $MB00000000[DW00000]$ is the same as $MB00000002$.

$DW00000 = 2;$

$DB000020 = MB00000000[DW00000];$ Equivalent \longleftrightarrow $DB000020 = MB00000002;$

- Attaching an Array Register to a Register Other Than a Bit Register

Using an array register is the same as adding the word size of the data type of the array register times the value of the array register to the register address.

For example, if $DW00000 = 30$, $ML00000002[DW00000]$ is the same as $ML0000062$.

$DL00002 = ML00000 (30 \times 2 + 2) = ML0000062$

$DW00000 = 30;$

$DL00002 = ML00000002[DW00000];$ Equivalent \longleftrightarrow $DL00002 = ML0000062;$

Ladder Language Instructions

4

This chapter describes the ladder language instructions in detail.

4.1 Introduction 4-6

- 4.1.1 Ladder Language Instructions 4-6
- 4.1.2 How to Read the Ladder Language Instructions 4-10

4.2 Relay Circuit Instructions 4-11

- 4.2.1 NO Contact (NOC) 4-11
- 4.2.2 Rising-edge NO Contact (ONP-NOC) 4-12
- 4.2.3 Falling-edge NO Contact (OFFP-NOC) 4-13
- 4.2.4 NC Contact (NCC) 4-14
- 4.2.5 Rising-edge NC Contact (ONP-NCC) 4-14
- 4.2.6 Falling-edge NC Contact (OFFP-NCC) 4-15
- 4.2.7 1-ms ON-Delay Timer (TON(1ms)) 4-16
- 4.2.8 1-ms OFF-Delay Timer (TOFF(1 ms)) 4-18
- 4.2.9 10-ms ON-Delay Timer (TON(10ms)) 4-19
- 4.2.10 10-ms OFF-Delay Timer (TOFF(10ms)) 4-21
- 4.2.11 1-s ON-Delay Timer (TON(1s)) 4-22
- 4.2.12 1-s OFF-Delay Timer (TOFF(1s)) 4-24
- 4.2.13 Rising-edge Pulses (ON-PLS) 4-25
- 4.2.14 Falling-edge Pulses (OFF-PLS) 4-27
- 4.2.15 Coil (COIL) 4-29
- 4.2.16 Reverse Coil (REV-COIL) 4-30
- 4.2.17 Rising-edge Detection Coil (ONP-COIL) 4-31
- 4.2.18 Falling-edge Detection Coil (OFFP-COIL) 4-31
- 4.2.19 Set Coil (S-COIL) 4-32
- 4.2.20 Reset Coil (R-COIL) 4-33

4.3 Numeric Operation Instructions 4-34

4.3.1	Store (STORE)	4-34
4.3.2	Add (ADD (+))	4-35
4.3.3	Extended Add (ADDX (++))	4-36
4.3.4	Subtract (SUB (-))	4-38
4.3.5	Extended Subtract (SUBX (--))	4-39
4.3.6	Multiply (MUL (x))	4-41
4.3.7	Divide (DIV (÷))	4-42
4.3.8	Integer Remainder (MOD)	4-43
4.3.9	Real Remainder (REM)	4-45
4.3.10	Increment (INC)	4-46
4.3.11	Decrement (DEC)	4-47
4.3.12	Add Time (TMADD)	4-48
4.3.13	Subtract Time (TMSUB)	4-50
4.3.14	Spend Time (SPEND)	4-52
4.3.15	Invert Sign (INV)	4-54
4.3.16	One's Complement (COM)	4-55
4.3.17	Absolute Value (ABS)	4-56
4.3.18	Binary Conversion (BIN)	4-57
4.3.19	BCD Conversion (BCD)	4-58
4.3.20	Parity Conversion (PARITY)	4-59
4.3.21	ASCII Conversion 1 (ASCII)	4-60
4.3.22	ASCII Conversion 2 (BINASC)	4-61
4.3.23	ASCII Conversion 3 (ASCBIN)	4-62

4.4 Logic Operations and Comparison Instructions . . 4-64

4.4.1	Inclusive AND (AND)	4-64
4.4.2	Inclusive OR (OR)	4-65
4.4.3	Exclusive OR (XOR)	4-66
4.4.4	Less Than (<)	4-67
4.4.5	Less Than or Equal (≤)	4-68
4.4.6	Equal (=)	4-69
4.4.7	Not Equal (≠)	4-70
4.4.8	Greater Than or Equal (≥)	4-71
4.4.9	Greater Than (>)	4-72
4.4.10	Range Check (RCHK)	4-73

4.5 Program Control Instructions 4-75

4.5.1	Call Sequence Program (SEE)	4-75
4.5.2	Call Motion Program (MSEE)	4-76
4.5.3	Call User Function (FUNC)	4-78
4.5.4	Direct Input String (INS)	4-79
4.5.5	Direct Output String (OUTS)	4-81
4.5.6	Call Extended Program (XCALL)	4-83
4.5.7	WHILE Construct (WHILE, END_WHILE)	4-84
4.5.8	FOR Construct (FOR, END_FOR)	4-86
4.5.9	IF Construct (IF, END_IF)	4-88
4.5.10	IF-ELSE Construct (IF, ELSE, END_IF)	4-90
4.5.11	Expression (EXPRESSION)	4-91

4.6 Basic Function Instructions 4-93

4.6.1	Square Root (SQRT)	4-93
4.6.2	Sine (SIN)	4-94
4.6.3	Cosine (COS)	4-95
4.6.4	Tangent (TAN.)	4-97
4.6.5	Arc Sine (ASIN)	4-98
4.6.6	Arc Cosine (ACOS)	4-99
4.6.7	Arc Tangent (ATAN)	4-100
4.6.8	Exponential (EXP)	4-101
4.6.9	Natural Logarithm (LN)	4-102
4.6.10	Common Logarithm (LOG)	4-103

4.7 Data Shift Instructions 4-104

4.7.1	Bit Rotate Left (ROTL)	4-104
4.7.2	Bit Rotate Right (ROTR)	4-105
4.7.3	Move Bit (MOVB)	4-106
4.7.4	Move Word (MOVW)	4-108
4.7.5	Exchange (XCHG)	4-110
4.7.6	Table Initialization (SETW)	4-111
4.7.7	Byte-to-word Expansion (BEXTD)	4-113
4.7.8	Word-to-byte Compression (BPRESS)	4-114
4.7.9	Binary Search (BSRCH)	4-116
4.7.10	Sort (SORT)	4-117
4.7.11	Bit Shift Left (SHFTL)	4-118
4.7.12	Bit Shift Right (SHFTR)	4-120
4.7.13	Copy Word (COPYW)	4-121
4.7.14	Byte Swap (BSWAP)	4-122

4.8 DDC Instructions 4-123

4.8.1	Dead Zone A (DZA)	4-123
4.8.2	Dead Zone B (DZB)	4-124
4.8.3	Upper/Lower Limit (LIMIT)	4-126
4.8.4	PI Control (PI)	4-128
4.8.5	PD Control (PD)	4-133
4.8.6	PID Control (PID)	4-137
4.8.7	First-order Lag (LAG)	4-142
4.8.8	Phase Lead Lag (LLAG)	4-144
4.8.9	Function Generator (FGN)	4-147
4.8.10	Inverse Function Generator (IFGN)	4-151
4.8.11	Linear Accelerator/Decelerator 1 (LAU)	4-155
4.8.12	Linear Accelerator/Decelerator 2 (SLAU)	4-161
4.8.13	Pulse Width Modulation (PWM)	4-170

4.9 Table Manipulation Instructions 4-173

4.9.1	Read Table Block (TBLBR/TBLBRE)	4-173
4.9.2	Write Table Block (TBLBW/TBLBWE)	4-177
4.9.3	Search for Table Row (TBL SRL/TBL SRLE)	4-181
4.9.4	Search for Table Column (TBL SRC/ TBL SRCE)	4-184
4.9.5	Clear Table Block (TBLCL/TBLCLE)	4-187
4.9.6	Move Table Block (TBLMV/TBLMVE)	4-190
4.9.7	Read Queue Table (QTBLR/QTBLRE and QTBLRI/QTBLRIE)	4-194
4.9.8	Write Queue Table (QTBLW/QTBLWE and QTBLWI/QTBLWIE)	4-198
4.9.9	Clear Queue Table Pointer (QTBLCL/ QTBLCLE)	4-202

4.10 System Function Instructions 4-204

4.10.1	Counter (COUNTER)	4-204
4.10.2	First-in First-out (FINFOUT)	4-207
4.10.3	Trace (TRACE)	4-210
4.10.4	Read Data Trace (DTRC-RD/DTRC-RDE)	4-212
4.10.5	Send Message (MSG-SND)	4-216
4.10.6	Send Message Extended (MSG-SNDE)	4-218
4.10.7	Receive Message (MSG-RCV)	4-220
4.10.8	Receive Message Extended (MSG-RCVE)	4-221
4.10.9	Write SERVOPACK Parameter (MLNK-SVW)	4-223
4.10.10	Read SERVOPACK Parameter (MLNK-SVR)	4-228
4.10.11	Flash Operation (FLASH-OP)	4-233
4.10.12	Write Motion Register (MOTREG-W)	4-236
4.10.13	Read Motion Register (MOTREG-R)	4-238
4.10.14	Import (IMPORT/IMPORTL/IMPORTLE)	4-240
4.10.15	Export (EXPORT/EXPORTL/EXPORTLE)	4-248

4.11 Storage Operation Instructions 4-254

4.11.1	Open File (FOPEN)	4-254
4.11.2	Close File (FCLOSE)	4-257
4.11.3	Read Data from File (FREAD)	4-258
4.11.4	Write Data to File (FWRITE)	4-260
4.11.5	Set File Position Indicator (FSEEK)	4-262
4.11.6	Read Line from File to String (FGETS)	4-264
4.11.7	Write String to File (FPUTS)	4-266
4.11.8	Copy File (FCOPY)	4-268
4.11.9	Delete File (FREMOVE)	4-270
4.11.10	Rename File (FRENAME)	4-271
4.11.11	Create Directory (DCREATE)	4-274
4.11.12	Delete Directory (DREMOVE)	4-276
4.11.13	Send File to FTP Server (FTPPUT)	4-277

4.12 String Operation Instructions 4-280

- 4.12.1 Convert Integer to String (INT2STR) 4-280
- 4.12.2 Convert Real Number to String (REAL2STR) . . 4-282
- 4.12.3 Convert String to Integer (STR2INT) 4-283
- 4.12.4 Convert String to Real Number (STR2REAL) . . 4-284
- 4.12.5 Store String (STRSET) 4-286
- 4.12.6 Partially Delete String (STRDEL) 4-287
- 4.12.7 Copy String (STRCPY) 4-288
- 4.12.8 Get String Length (STRLEN) 4-290
- 4.12.9 Concatenate Strings (STRCAT) 4-291
- 4.12.10 Compare Strings (STRCMP) 4-293
- 4.12.11 Insert String (STRINS) 4-294
- 4.12.12 Find String (STRFIND) 4-296
- 4.12.13 Extract String (STREXTR) 4-297
- 4.12.14 Extract String from End (STREXTRE) 4-299
- 4.12.15 Delete Spaces at String Ends (STRTRIM) 4-300

4.1 Introduction

This section describes the types of ladder language instructions and their functionality. It also shows how to interpret the rest of this chapter.

4.1.1 Ladder Language Instructions

The following table lists the ladder language instructions.

Type	Instruction	Meaning	GUI Name
Relay Circuit Instructions	NOC	NO Contact	NO Contact
	ONP-NOC	Rising-edge NO Contact	Rising-edge NO Contact
	OFFP-NOC	Falling-edge NO Contact	Falling-edge NO Contact
	NCC	NC Contact	NC Contact
	ONP-NCC	Rising-edge NC Contact	Rising-edge NC Contact
	OFFP-NCC	Falling-edge NC Contact	Falling-edge NC Contact
	TON (1 ms)	1-ms ON-Delay Timer	On-Delay Timer (1ms)
	TOFF (1 ms)	1-ms OFF-Delay Timer	Off-Delay Timer (1ms)
	TON (10 ms)	10-ms ON-Delay Timer	On-Delay Timer (10ms)
	TOFF (10 ms)	10-ms OFF-Delay Timer	Off-Delay Timer (10ms)
	TON (1 s)	1-s ON-Delay Timer	On-Delay Timer (1s)
	TOFF (1 s)	1-s OFF-Delay Timer	Off-Delay Timer (1s)
	ON-PLS	Rising-edge Pulses	Rising Edge Pulses
	OFF-PLS	Falling-edge Pulses	Falling Edge Pulses
	COIL	Coil	Coil
	REV-COIL	Reverse Coil	Reverse Coil
	ONP-COIL	Rising-edge Detection Coil	Rising-edge Detection Coil
	OFFP-COIL	Falling-edge Detection Coil	Falling-edge Detection Coil
S-COIL	Set Coil	Set Coil	
R-COIL	Reset Coil	Reset Coil	

Continued on next page.

Continued from previous page.

Type	Instruction	Meaning	GUI Name
Numeric Operation Instructions	STORE	Store	Store
	ADD(+)	Add	Addition
	ADDX(+ +)	Extended Add	Extended Addition
	SUB(-)	Subtract	Subtraction
	SUBX(- -)	Extended Subtract	Extended Subtraction
	MUL(x)	Multiply	Multiplication
	DIV(+)	Divide	Division
	MOD	Integer Remainder	Integer Remainder
	REM	Real Remainder	Real Remainder
	INC	Increment	Increment
	DEC	Decrement	Decrement
	TMADD	Add Time	Add Time
	TMSUB	Subtract Time	Subtract Time
	SPEND	Spend Time	Spend Time
	INV	Invert Sign	Sign Inversion
	COM	One's Complement	1's Complement
	ABS	Absolute Value	Absolute Value
	BIN	Binary Conversion	Binary Conversion
	BCD	BCD Conversion	BCD Conversion
PARITY	Parity Conversion	Parity Conversion	
ASCII	ASCII Conversion 1	ASCII Conversion 1	
BINASC	ASCII Conversion 2	ASCII Conversion 2	
ASCBIN	ASCII Conversion 3	ASCII Conversion 3	
Logic Operation Instructions	AND	Inclusive AND	Inclusive AND
	OR	Inclusive OR	Inclusive OR
	XOR	Exclusive OR	Exclusive OR
	<	Less Than	Less Than (A<B)
	≤	Less Than or Equal	Less Than or Equal (A<=B)
	=	Equal	Equal (A==B)
	≠	Not Equal	Not Equal (A!=B)
	≥	Greater Than or Equal	Greater Than or Equal (A>=B)
	>	Greater Than	Greater Than (A>B)
RCHK	Range Check	Range Check	
Program Control Instructions	SEE	Call Sequence Subprogram	Call Program
	MSEE	Call Motion Program	Call Motion Program
	FUNC	Call User Function	User Function
	INS	Direct Input String	Direct Input String
	OUTS	Direct Output String	Direct Output String
	XCALL	Call Extended Program	Call Extended Program
	WHILE END_WHILE	WHILE construct	While/Do While End
	FOR END_FOR	FOR construct	For For End
	IF END_IF	IF construct	If/Then If End
	IF ELSE END_IF	IF-ELSE construct	If/Then Else If End
	EXPRESSION	Expression	Expression

Continued on next page.

Continued from previous page.

Type	Instruction	Meaning	GUI Name
Basic Function Instructions	SQRT	Square Root	Square Root
	SIN	Sine	Sine
	COS	Cosine	Cosine
	TAN	Tangent	Tangent
	ASIN	Arc Sine	Arc Sine
	ACOS	Arc Cosine	Arc Cosine
	ATAN	Arc Tangent	Arc Tangent
	EXP	Exponential	Exponential
	LN	Natural Logarithm	Natural Logarithm
LOG	Common Logarithm	Common Logarithm	
Data Manipulation Instructions	ROTL	Bit Rotate Left	Bit Rotate Left
	ROTR	Bit Rotate Right	Bit Rotate Right
	MOVB	Move Bit	Move Bit
	MOVW	Move Word	Move Word
	XCHG	Exchange	Exchange
	SETW	Table Initialization	Set Word
	BEXTD	Byte-to-word Expansion	Byte to Word Expansion
	BPRESS	Word-to-byte Compression	Word to Byte Compression
	BSRCH	Binary Search	Binary Search
	SORT	Sort	Sort
	SHFTL	Bit Shift Left	Bit Shift Left
	SHFTR	Bit Shift Right	Bit Shift Right
	COPYW	Copy Word	Copy Word
	BSWAP	Byte Swap	Byte Swap
DDC Instructions	DZA	Dead Zone A	Dead Zone A
	DZB	Dead Zone B	Dead Zone B
	LIMIT	Upper/Lower Limit	Upper/Lower Limit
	PI	PI Control	PI Control
	PD	PD Control	PD Control
	PID	PID Control	PID Control
	LAG	First-order Lag	First Order Lag
	LLAG	Phase Lead Lag	Phase Lead Lag
	FGN	Function Generator	Function Generator
	IFGN	Inverse Function Generator	Inverse Function Generator
	LAU	Linear Accelerator/Decelerator 1	Linear Accelerator/Decelerator1
SLAU	Linear Accelerator/Decelerator 2	Linear Accelerator/Decelerator2	
PWM	Pulse Width Modulation	Pulse Width Modulation	
Table Manipulation Instructions	TBLBR/TBLBRE	Read Table Block	Table Block Read
	TBLBW/TBLBWE	Write Table Block	Table Block Write
	TBLSRL/TBLSRLE	Search for Table Row	Table Row Search
	TBLSRC/TBLSRCE	Search for Table Column	Table Column Search
	TBLCL/TBLCLE	Clear Table Block	Table Block Clear
	TBLMV/TBLMVE	Move Table Block	Table Block Move
	QTBLR/QTBLRE	Read Queue Table	Queue Table Read
	QTBLRI/QTBLRIE	Read Queue Table with Pointer Increment	Queue Table Read with Pointer Increment
	QTBLW/QTBLWE	Write Queue Table	Queue Table Write
	QTBLWI/QTBLWIE	Write Queue Table with Pointer Increment	Queue Table Write with Pointer Increment
QTBLCL/QTBLCLE	Clear Queue Table Pointer	Queue Table Pointer Clear	

Continued on next page.

Continued from previous page.

Type	Instruction	Meaning	GUI Name
Standard System Function Instructions	COUNTER	Counter	Counter
	FINFOUT	First-in First-out	First-in First-out
	TRACE	Trace	Trace
	DTRC-RD/DTRC-RDE	Read Data Trace	Data-Trace Read Extend
	MSG-SND	Send Message	Send Message
	MSG-SNDE	Send Message Extended	Send Message Extend
	MSG-RCV	Receive Message	Receive Message
	MSG-RCVE	Receive Message Extended	Receive Message Extend
	MLNK-SVW	Write SERVOPACK Parameter	Write Servo Pack Parameter with MECHATROLINK
	MLNK-SVR	Read SERVOPACK Parameter	Read Servo Pack Parameter with MECHATROLINK
	FLASH-OP	Flash Operation	Operate Flash Memory
	MOTREG-W	Write Motion Register	Write the Motion Parameter to Motion Register
	MOTREG-R	Read Motion Register	Read the Motion Parameter from Motion Register
	IMPORT/IMPORTL/ IMPORTLE	Import	Import
	EXPORT/EXPORTL/ EXPORTLE	Export	Export
Storage Operation Instructions	FOPEN	Open File	Open File
	FCLOSE	Close File	Close File
	FREAD	Read Data from File	Read Data from File
	FWRITE	Write Data to File	Write Data to File
	FSEEK	Set File Position Indicator	Set File Position Indicator
	FGETS	Read Line from File to String	Read Line from File to String
	FPUTS	Write String to File	Write String to File
	FCOPY	Copy File	Copy File
	FREMOVE	Delete File	Delete File
	FRENAME	Rename File	Rename File
	DCREATE	Create Directory	Create Directory
	DREMOVE	Delete Directory	Delete Directory
String Operation Instructions	FTPPUT	Send File to FTP Server	Send File to FTP Server
	INT2STR	Convert Integer to String	Convert Integer to String
	REAL2STR	Convert Real Number to String	Convert Real Number to String
	STR2INT	Convert String to Integer	Convert String to Integer
	STR2REAL	Convert String to Real Number	Convert String to Real Number
	STRSET	Store String	Store String
	STRDEL	Partially Delete String	Partially Delete String
	STRCPY	Copy String	Copy String
	STRLEN	Get String Length	Get String Length
	STRCAT	Concatenate Strings	Concatenate Strings
	STRCMP	Compare Strings	Compare Strings
	STRINS	Insert String	Insert String
	STRFIND	Find String	Find String
	STREXTR	Extract String	Extract String
	STREXTRE	Extract String from End	Extract String from End
STRTRIM	Delete Spaces at String Ends	Delete Spaces at String Ends	

4.1.2 How to Read the Ladder Language Instructions

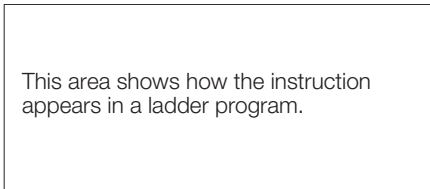
This chapter describes each instruction using the following format.

The operation performed by the instruction is described.

Where necessary, a diagram is used to show the operation performed by the instruction.

Format

This section describes the format for programming the instruction.



Icon: Shows the icon used in the MPE720.


Key entry: Shows the shortcut key combination used in the Ladder Editor.

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Label used in the ladder diagram	×	○	○	×	○	×	×	○	○

Note: 1. x: This data type cannot be used.

○: All registers with this data type can be used.

2. Refer to the following chapter for details on data types.

 *Chapter 3 Registers*

Programming Example

This section gives a ladder programming example that uses the instruction.

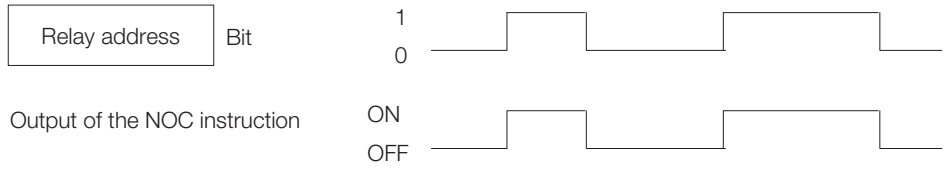
Additional Information

This section may contain additional information about the instruction. It is omitted if there is no additional information that is required for the instruction.

4.2 Relay Circuit Instructions

4.2.1 NO Contact (NOC)

The relay outputs ON whenever the bit with the specified relay address is 1.
 The relay outputs OFF when the bit is 0.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Relay address	○	×	×	×	×	×	×	×	×

Programming Example

The DB000001 output coil is ON whenever the DB000000 relay in the NOC instruction is ON.



4.2.2 Rising-edge NO Contact (ONP-NOC)

ON is output for only one scan when the bit input changes from 0 to 1.

The resulting operation is the combination of the NOC and ON-PLS instructions.

- Information** • This is the same operation as that of the OFFP-NCC instruction.
- The ONP-NOC instruction cannot be used in user functions. Use an NO contact with the Rising-edge Pulses (ON-PLS) instruction.

Format

The format of this instruction is shown below.

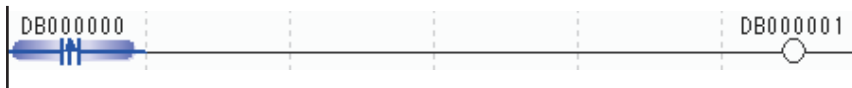


I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Relay address	O*	x	x	x	x	x	x	x	x

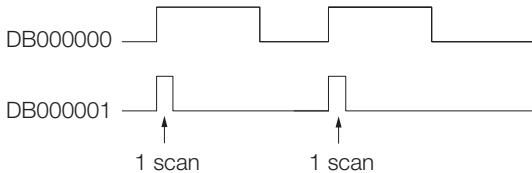
* The # and C registers will not produce the desired result because they are constant registers that do not undergo value changes.

Programming Example

The DB000001 output coil turns ON when the DB000000 relay in the NOC instruction changes from OFF to ON.



The timing chart is shown below.



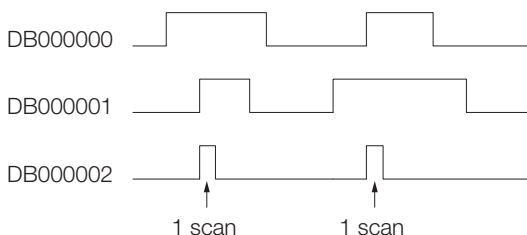
- Information** If you program another instruction before the ONP-NOC instruction, the result of the operation with the other instruction will be output.
- In the following example, the DB000002 output coil turns ON when the AND condition of the DB000000 and DB000001 relays changes from OFF to ON.



The following circuit is equivalent to the above circuit.



The timing chart is shown below.



4.2.3 Falling-edge NO Contact (OFFP-NOC)

ON is output for only one scan when the bit input changes from 1 to 0.

The resulting operation is the combination of the NOC and OFF-PLS instructions.

- Information**
- This is the same operation as that of the ONP-NCC instruction.
 - The OFFP-NOC instruction cannot be used in user functions. Use an NO contact with the Falling-edge Pulses (OFF-PLS) instruction.

Format

The format of this instruction is shown below.

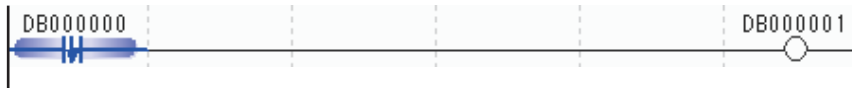


I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Relay address	○*	×	×	×	×	×	×	×	×

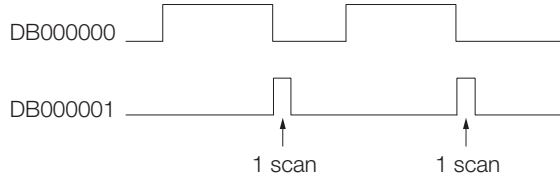
* The # and C registers will not produce the desired result because they are constant registers that do not undergo value changes.

Programming Example

The DB000001 output coil turns ON when the DB000000 relay in the NOC instruction changes from ON to OFF.



The timing chart is shown below.



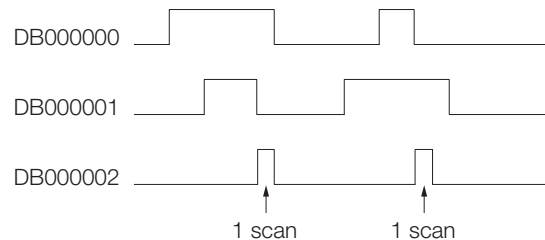
- Information**
- If you program another instruction before the OFFP-NOC instruction, the result of the operation with the other instruction will be output.
- In the following example, the DB000002 output coil turns ON when the AND condition of the DB000000 and DB000001 relays changes from ON to OFF.



The following circuit is equivalent to the above circuit.

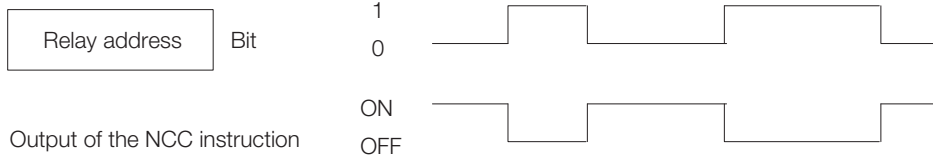


The timing chart is shown below.



4.2.4 NC Contact (NCC)

The relay outputs OFF whenever the bit with the specified relay address is 1.
 The relay outputs ON when the bit is 0.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Relay address	O	x	x	x	x	x	x	x	x

Programming Example

The DB000001 coil is ON whenever the DB000000 relay in the NCC instruction is OFF.



4.2.5 Rising-edge NC Contact (ONP-NCC)

ON is output for only one scan when the bit input changes from 1 to 0.
 The resulting operation is the combination of the NCC and ON-PLS instructions.

- Information** • This is the same operation as that of the OFFP-NOC instruction.
- The ONP-NCC instruction cannot be used in user functions. Use an NC contact with the Rising-edge Pulses (ON-PLS) instruction.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Relay address	O*	x	x	x	x	x	x	x	x

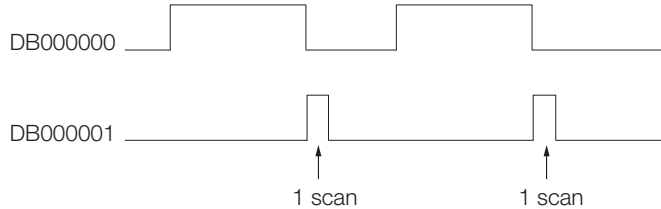
* The # and C registers will not produce the desired result because they are constant registers that do not undergo value changes.

Programming Example

The DB000001 output coil turns ON when the DB000000 relay in the NCC instruction changes from ON to OFF.



The timing chart is shown below.



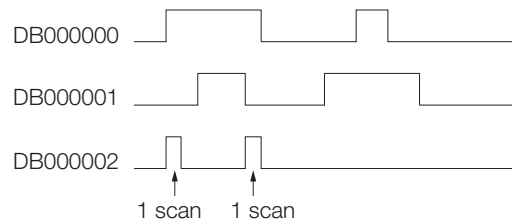
Information If you program another instruction before the ONP-NCC instruction, the result of the operation with the other instruction will be output.
 In the following example, the DB000002 output coil turns ON when the AND condition of the DB000000 relay and the inverted status of the DB000001 relay changes from OFF to ON.



The following circuit is equivalent to the above circuit.



The timing chart is shown below.



4.2.6 Falling-edge NC Contact (OFFP-NCC)

ON is output for only one scan when the bit input changes from 0 to 1.

The resulting operation is the combination of the NCC and OFF-PLS instructions.

- Information** • This is the same operation as that of the ONP-NOC instruction.
- The OFFP-NCC instruction cannot be used in user functions. Use an NC contact with the Falling-edge Pulses (OFF-PLS) instruction.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Relay address	O*	x	x	x	x	x	x	x	x

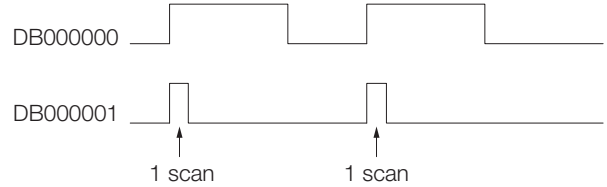
* The # and C registers will not produce the desired result because they are constant registers that do not undergo value changes.

Programming Example

The DB000001 output coil turns ON when the DB000000 relay in the NCC instruction changes from OFF to ON.



The timing chart is shown below.



Information

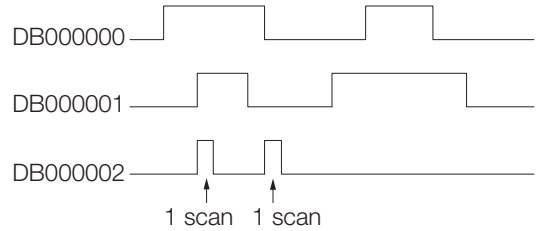
If you program another instruction before the OFFP-NCC instruction, the result of the operation with the other instruction will be output. In the following example, the DB000002 output coil turns ON when the AND condition of the DB000000 relay and the inverted status of the DB000001 relay changes from ON to OFF.



The following circuit is equivalent to the above circuit.



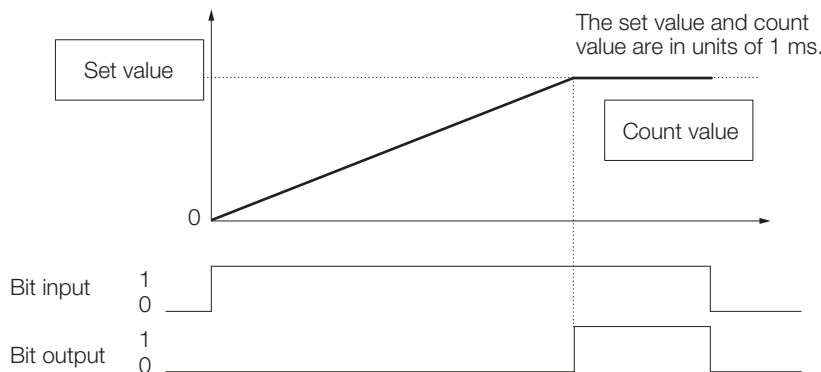
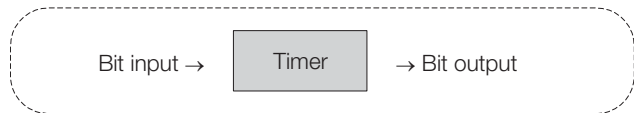
The timing chart is shown below.



4.2.7 1-ms ON-Delay Timer (TON(1ms))

The timer counts the time whenever the timer bit input is 1. The bit output is set to 1 when the count value equals the set value.

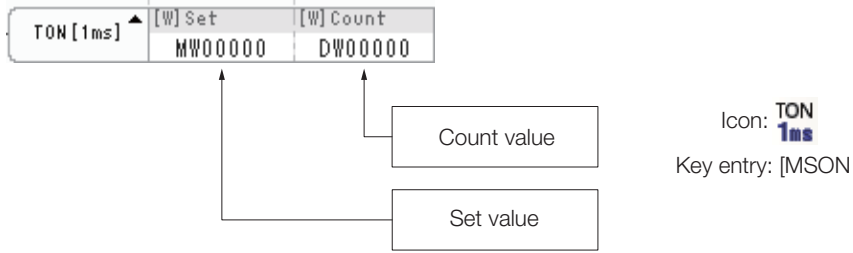
If the bit input changes to 0 during counting, the timer will stop counting. If the bit input changes to 1 again, the timer starts counting again from the beginning (i.e., from 0). The actual counted time (in units of 1 ms) is stored in the Count register.



Note: The counting error is 1 ms or less.

Format

The format of this instruction is shown below.



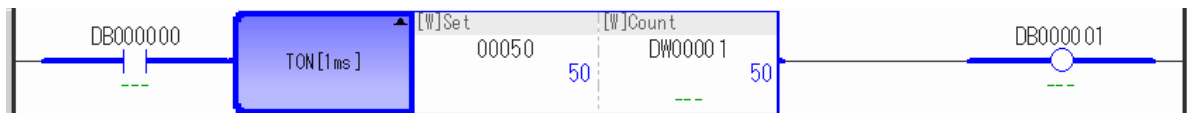
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Set (Set value)	×	○	×	×	×	×	×	×	○
Count (Count value)	×	○*	×	×	×	×	×	×	×

* C and # registers cannot be used.

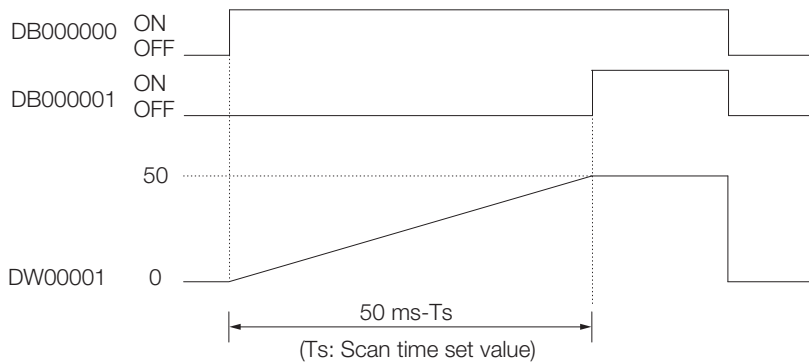
Programming Example

In the following programming example, the set value of the TON instruction is 50, and the count value is stored in the DW00001 register.

The DB000001 coil will turn ON after the DB000000 relay stays ON for 50 ms.



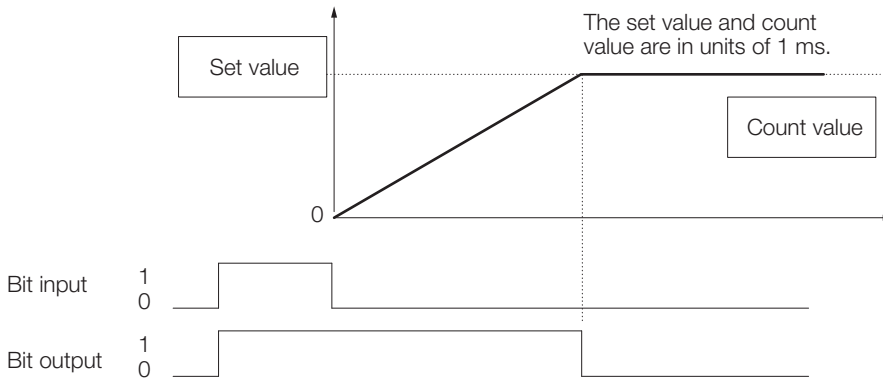
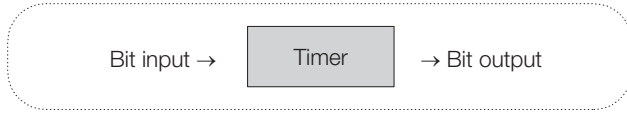
The timing chart is shown below.



4.2.8 1-ms OFF-Delay Timer (TOFF(1 ms))

The timer counts the time whenever the timer bit input is 0. The bit output is set to 0 when the count value equals the set value.

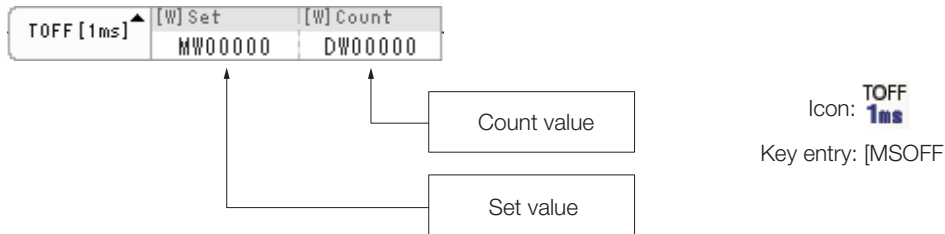
If the bit input changes to 1 during counting, the timer will stop counting. If the bit input changes to 0 again, the timer starts counting again from the beginning (i.e., from 0). The actual counted time (in units of 1 ms) is stored in the Count register.



Note: The counting error is 1 ms or less.

Format

The format of this instruction is shown below.



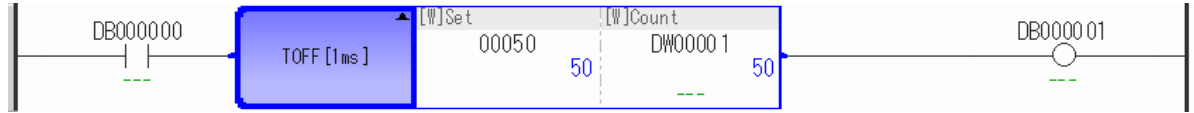
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Set (Set value)	×	○	×	×	×	×	×	×	○
Count (Count value)	×	○*	×	×	×	×	×	×	×

* C and # registers cannot be used.

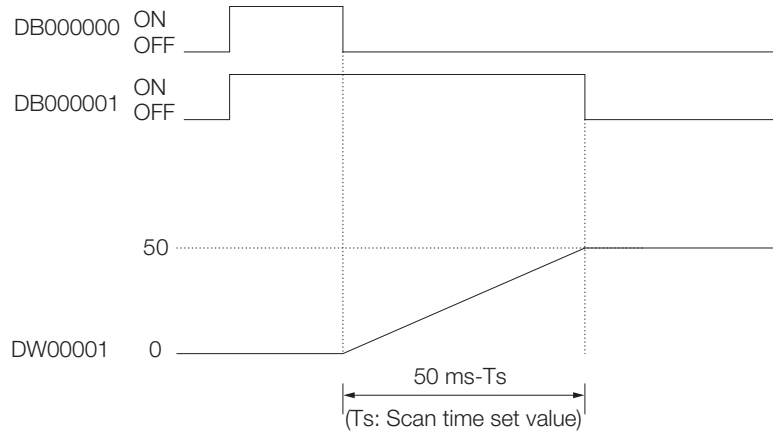
Programming Example

In the following programming example, the set value of the TOFF instruction is 50, and the count value is stored in the DW00001 register.

The DB000001 coil will turn OFF after the DB000000 relay stays OFF for 50 ms.



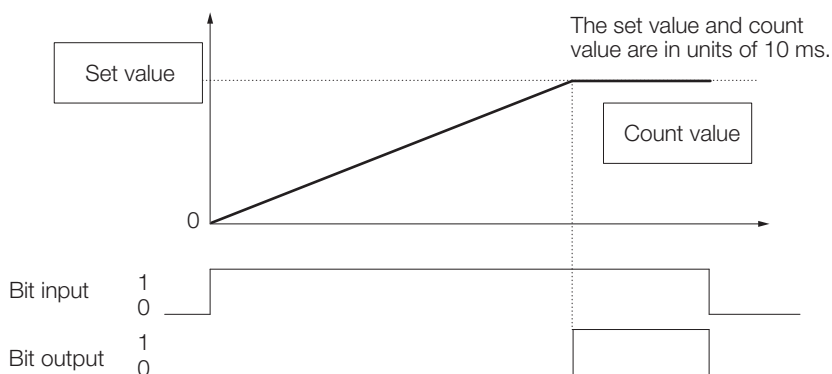
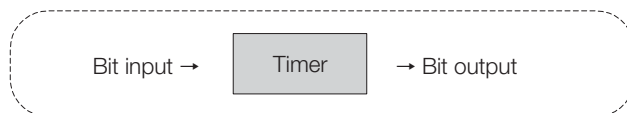
The timing chart is shown below.



4.2.9 10-ms ON-Delay Timer (TON(10ms))

The timer counts the time whenever the timer bit input is 1. The bit output is set to 1 when the count value equals the set value.

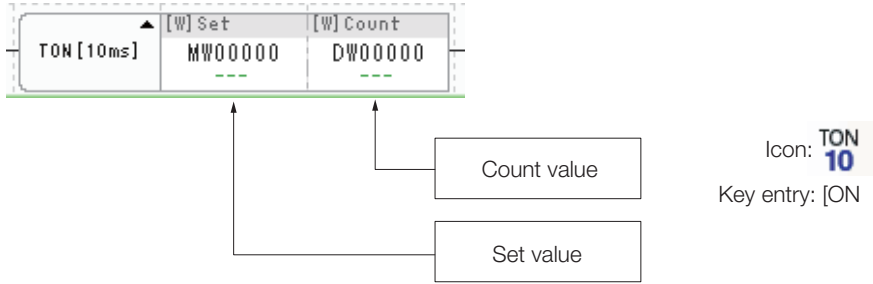
If the bit input changes to 0 during counting, the timer will stop counting. If the bit input changes to 1 again, the timer starts counting again from the beginning (i.e., from 0). The actual counted time (in units of 10 ms) is stored in the Count register.



Note: The counting error is 10 ms or less.

Format

The format of this instruction is shown below.



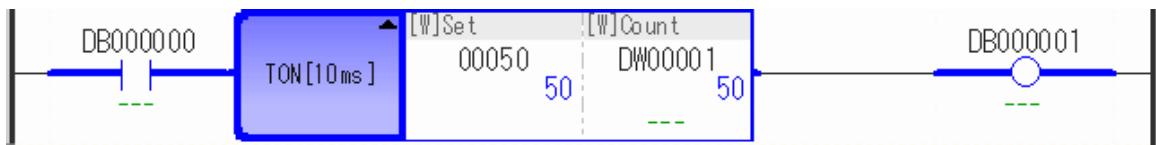
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Set (Set value)	×	○	×	×	×	×	×	×	○
Count (Count value)	×	○*	×	×	×	×	×	×	×

* C and # registers cannot be used.

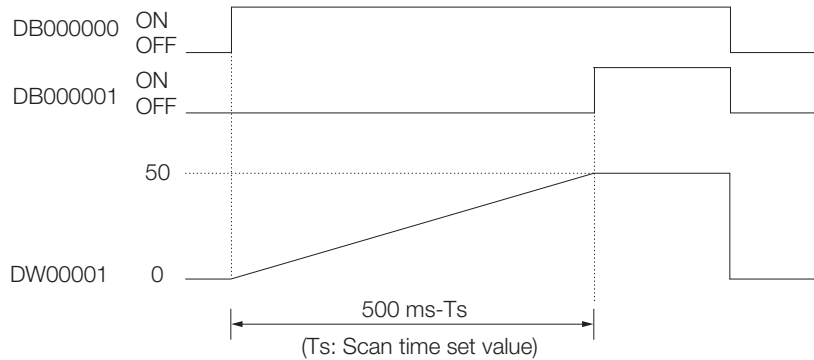
Programming Example

In the following programming example, the set value of the TON instruction is 50, and the count value is stored in the DW00001 register.

The DB000001 coil will turn ON after the DB000000 relay stays ON for 500 ms.



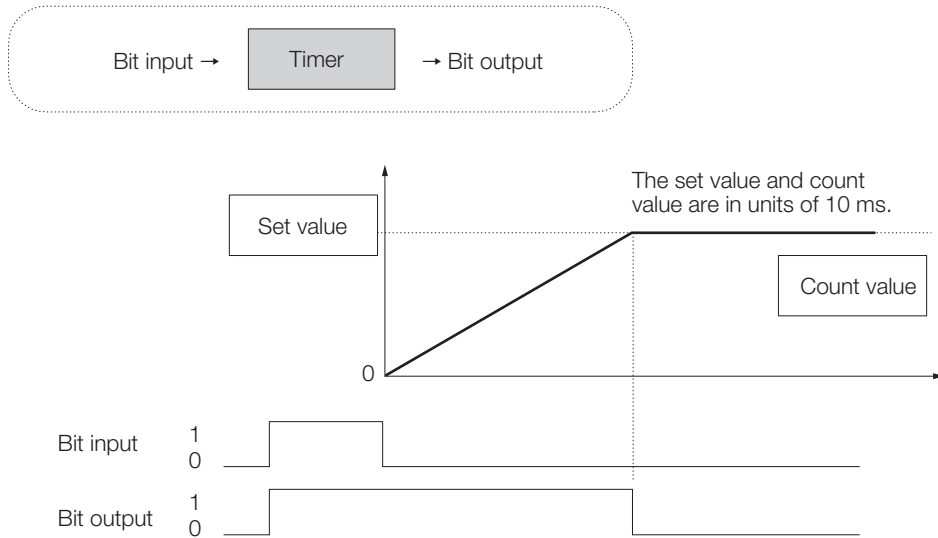
The timing chart is shown below.



4.2.10 10-ms OFF-Delay Timer (TOFF(10ms))

The timer counts the time whenever the timer bit input is 0. The bit output is set to 0 when the count value equals the set value.

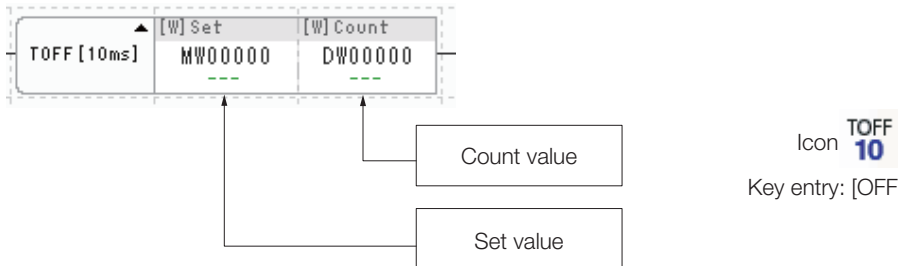
If the bit input changes to 1 during counting, the timer will stop counting. If the bit input changes to 0 again, the timer starts counting again from the beginning (i.e., from 0). The actual counted time (in units of 10 ms) is stored in the Count register.



Note: The counting error is 10 ms or less.

Format

The format of this instruction is shown below.



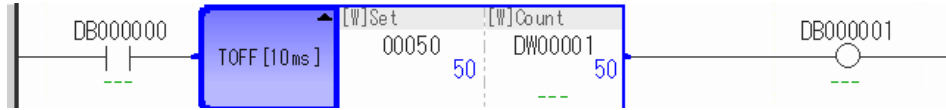
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Set (Set value)	×	○	×	×	×	×	×	×	○
Count (Count value)	×	○*	×	×	×	×	×	×	×

* C and # registers cannot be used.

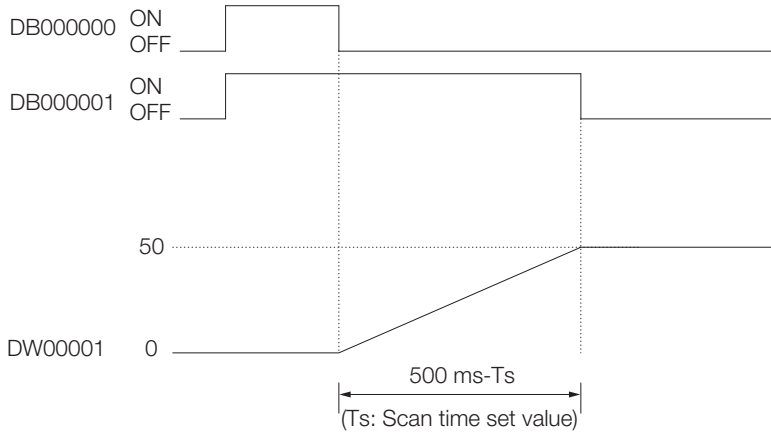
Programming Example

In the following programming example, the set value of the TOFF instruction is 50, and the count value is stored in the DW00001 register.

The DB000001 coil will turn OFF after the DB000000 relay stays OFF for 500 ms.



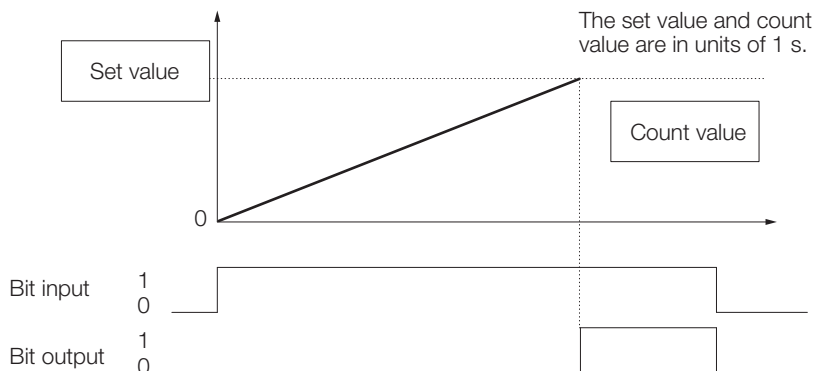
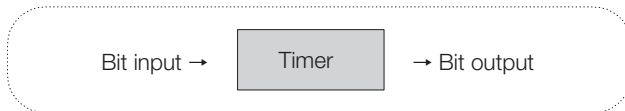
The timing chart is shown below.



4.2.11 1-s ON-Delay Timer (TON(1s))

The timer counts the time whenever the timer bit input is 1. The bit output is set to 1 when the count value equals the set value.

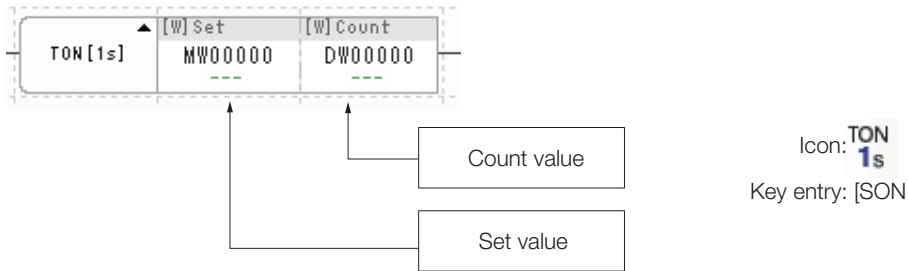
If the bit input changes to 0 during counting, the timer will stop counting. If the bit input changes to 1 again, the timer starts counting again from the beginning (i.e., from 0). The actual counted time (in units of 1 s) is stored in the Count register.



Note: The counting error is 1 s or less.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Set (Set value)	×	○	×	×	×	×	×	×	○
Count (Count value)	×	○*	×	×	×	×	×	×	×

* C and # registers cannot be used.

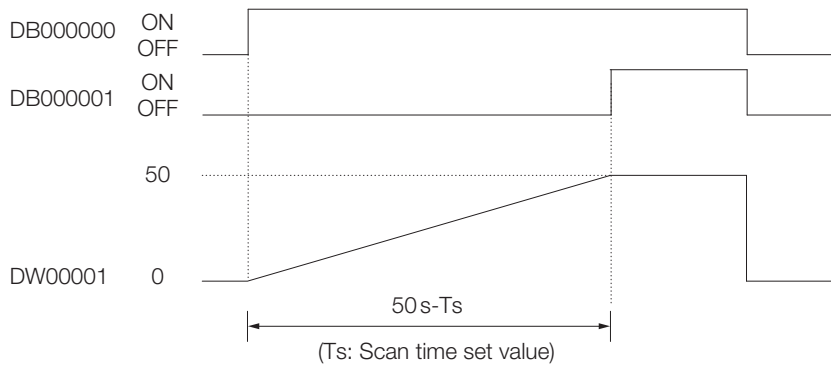
Programming Example

In the following programming example, the set value of the TON instruction is 50, and the count value is stored in the DW00001 register.

The DB000001 coil will turn ON after the DB000000 relay stays ON for 50 s.



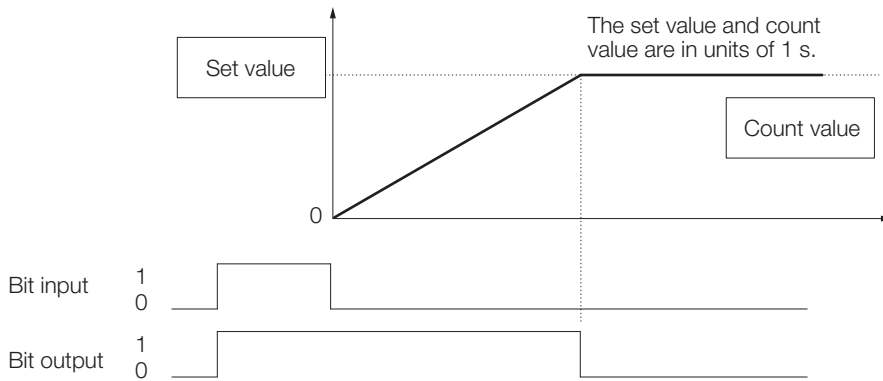
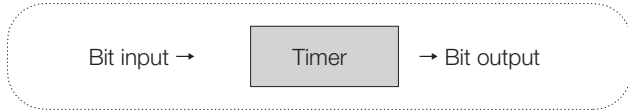
The timing chart is shown below.



4.2.12 1-s OFF-Delay Timer (TOFF(1s))

The timer counts the time whenever the timer bit input is 0. The bit output is set to 1 when the count value equals the set value.

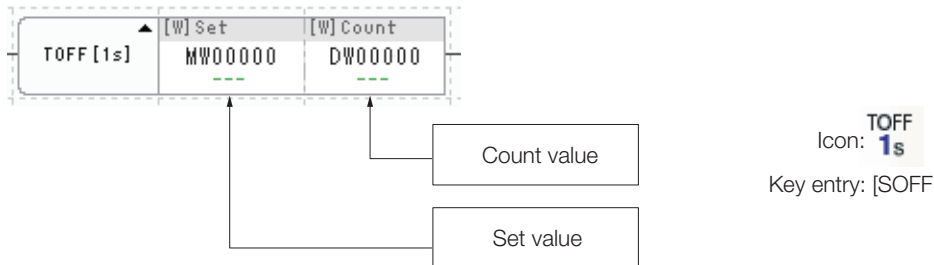
If the bit input changes to 1 during counting, the timer will stop counting. If the bit input changes to 0 again, the timer starts counting again from the beginning (i.e., from 0). The actual counted time (in units of 1 s) is stored in the Count register.



Note: The counting error is 1 s or less.

Format

The format of this instruction is shown below.



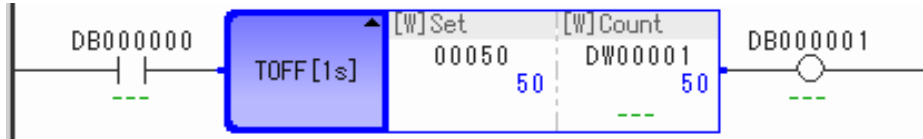
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Set (Set value)	×	○	×	×	×	×	×	×	○
Count (Count value)	×	○*	×	×	×	×	×	×	×

* C and # registers cannot be used.

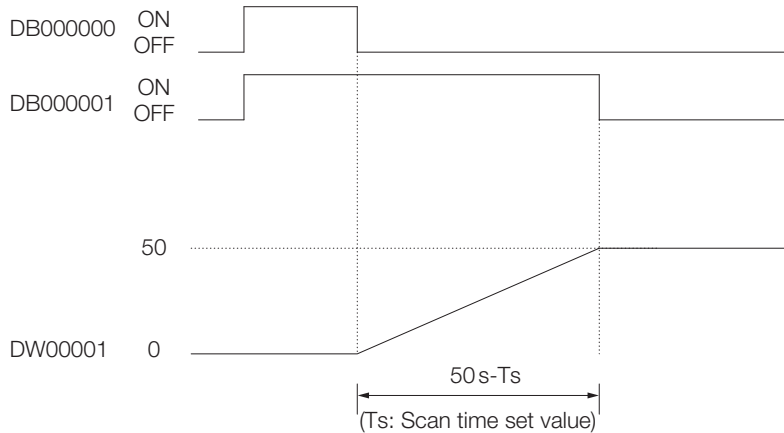
Programming Example

In the following programming example, the set value of the TOFF instruction is 50, and the count value is stored in the DW00001 register.

The DB000001 coil will turn OFF after the DB000000 relay stays OFF for 50 s.

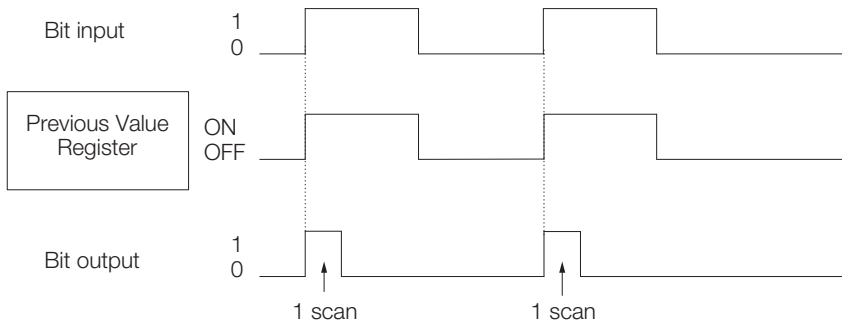


The timing chart is shown below.



4.2.13 Rising-edge Pulses (ON-PLS)

The bit output is set to 1 for only one scan when the bit input changes from 0 to 1. The previous value of the bit input is saved in the Previous Value Register of the ON-PLS instruction.



The following truth table shows the relationship between the bit input, the Previous Value Register, and the bit output of the ON-PLS instruction.

Bit Output	Previous Value Register	ON-PLS Instruction	Bit Input
0	OFF	→	0
0	ON	→	0
1	OFF	→	1
1	ON	→	0

In the third row of the table, notice how the bit input changes from 0 in the Previous Value Register to 1, causing the ON-PLS instruction to set the bit output to 1.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Previous Value Register	O*	×	×	×	×	×	×	×	×

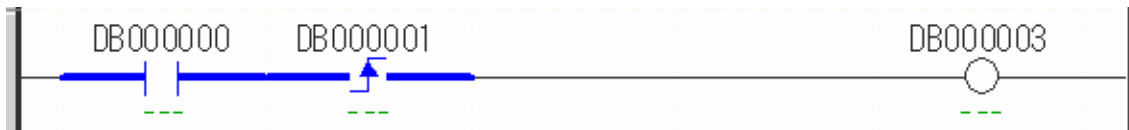
* C and # registers cannot be used.

Note: The Previous Value Register holds the previous value of the bit input. Do not use other instructions to set the value of this register.

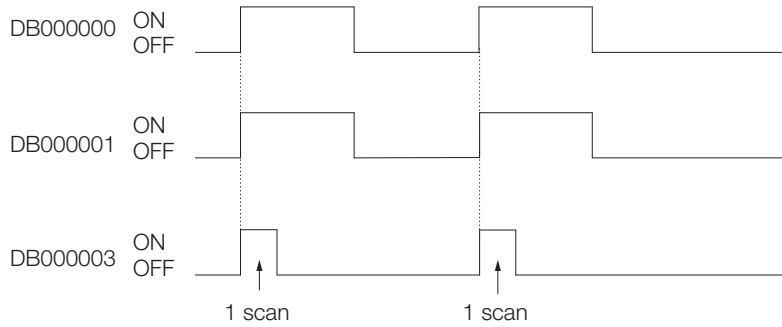
Programming Example

The DB000003 output coil turns ON for only one scan when the DB000000 relay changes from OFF to ON. The DB000001 register is used to store the previous value of DB000000.

Information Do not use more than one previous value register for the same drawing.

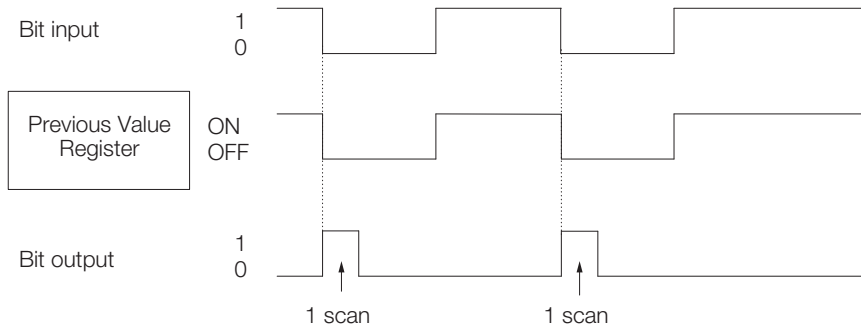


The timing chart is shown below.



4.2.14 Falling-edge Pulses (OFF-PLS)

The bit output is set to 1 for only one scan when the bit input changes from 1 to 0. The previous value of the bit input is saved in the Previous Value Register of the OFF-PLS instruction.



The following truth table shows the relationship between the bit input, the Previous Value Register, and the bit output of the OFF-PLS instruction.

Bit Output	Previous Value Register	OFF-PLS Instruction	Bit Input
0	OFF	→	0
0	ON	→	1
1	OFF	→	0
1	ON	→	0

In the second row of the table, notice how the bit input changes from 1 in the Previous Value Register to 0, causing the OFF-PLS instruction to set the bit output to 1.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Previous Value Register	O*	x	x	x	x	x	x	x	x

* C and # registers cannot be used.

Note: The Previous Value Register holds the previous value of the bit input. Do not use other instructions to set the value of this register.

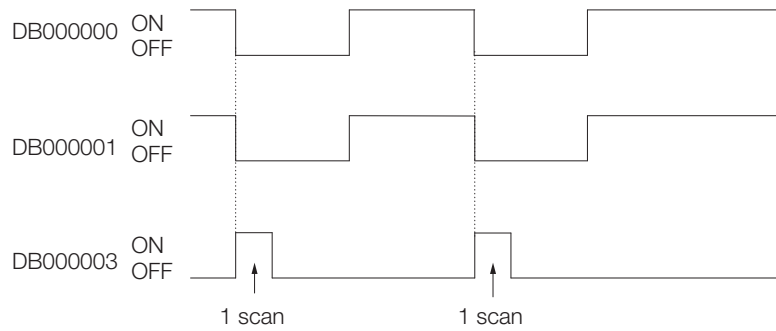
Programming Example

The DB000003 output coil turns ON for only one scan when the DB000000 relay changes from ON to OFF. The DB000001 register is used to store the previous value of DB000000.

Information Do not use more than one previous value register for the same drawing.

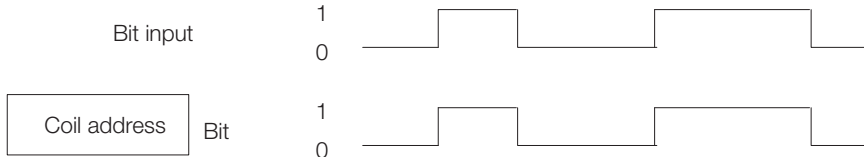


The timing chart is shown below.



4.2.15 Coil (COIL)

The bit at the coil address is set to 1 whenever the bit input is 1. The bit at the coil address is set to 0 whenever the bit input is 0.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Coil address	○*	×	×	×	×	×	×	×	×

* C and # registers cannot be used.

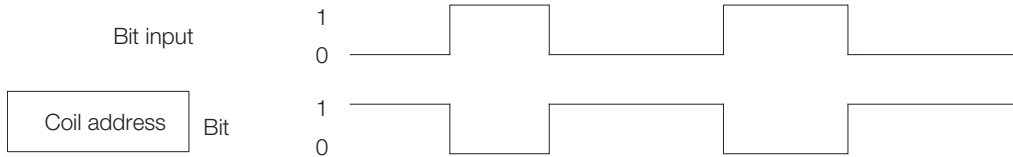
Programming Example

The DB000000 coil turns ON when the DB000001 relay turns ON.



4.2.16 Reverse Coil (REV-COIL)

The bit at the coil address is set to 1 whenever the bit input is 0. The bit at the coil address is set to 0 whenever the bit input is 1.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Coil address	O*	x	x	x	x	x	x	x	x

* # and C registers cannot be used.

Programming Example

The DB000000 coil turns OFF when the DB000001 relay turns ON.



4.2.17 Rising-edge Detection Coil (ONP-COIL)

The bit at the coil address is set to 1 for only one scan when the bit input changes from 0 to 1. The resulting operation is the same as the combination of the ON-PLS and COIL instructions.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Coil address	O*	x	x	x	x	x	x	x	x

* # and C registers cannot be used.

Programming Example

The DB000001 rising-edge detection coil turns ON when the DB000000 relay in the NOC instruction changes from OFF to ON.



4.2.18 Falling-edge Detection Coil (OFFP-COIL)

The bit at the coil address is set to 1 for only one scan when the bit input changes from 1 to 0. The resulting operation is the same as the combination of the OFF-PLS and COIL instructions.

Format

The format of this instruction is shown below.

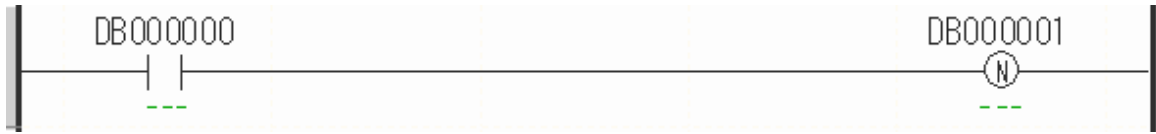


I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Coil address	O*	x	x	x	x	x	x	x	x

* # and C registers cannot be used.

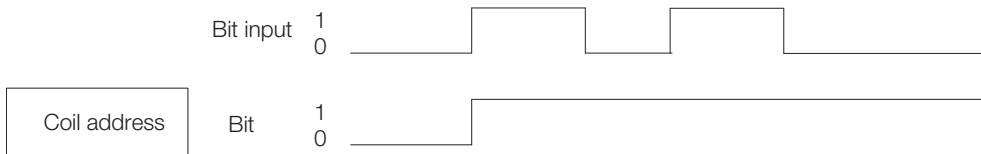
Programming Example

The DB000001 falling-edge detection coil turns ON when the DB000000 relay in the NOC instruction changes from ON to OFF.



4.2.19 Set Coil (S-COIL)

The bit at the coil address is set to 1 when the bit input is 1. The set coil stays in the ON state.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Coil address	O*	x	x	x	x	x	x	x	x

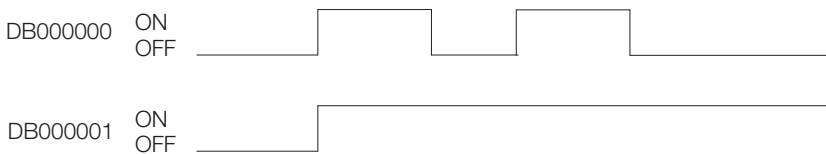
* C and # registers cannot be used.

Programming Example

The DB000001 set coil stays in the ON state when the DB000000 relay turns ON.

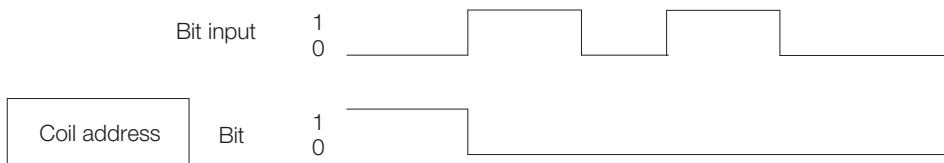


The timing chart is shown below.



4.2.20 Reset Coil (R-COIL)

The bit at the reset coil address is set to 0 when the bit input is 1. The reset coil stays in the OFF state.



Format

The format of this instruction is shown below.



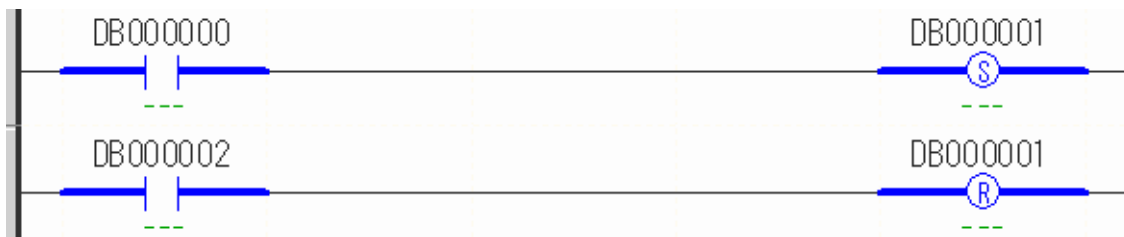
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Coil address	O*	x	x	x	x	x	x	x	x

* C and # registers cannot be used.

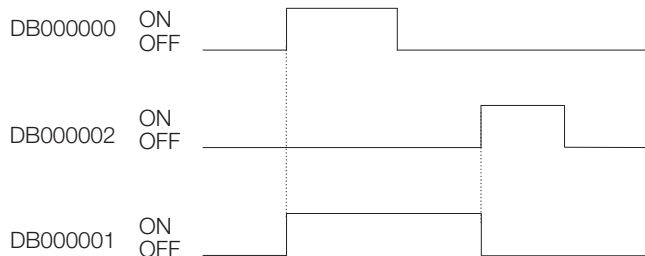
Programming Example

In the following programming example, the reset coil is used to turn OFF the set coil that was turned ON in the first line.

The DB000001 reset coil in the second line turns ON if the DB000002 relay turns ON while the DB000001 set coil is ON, therefore turning OFF the DB000001 set coil.



The timing chart is shown below.



4.3 Numeric Operation Instructions

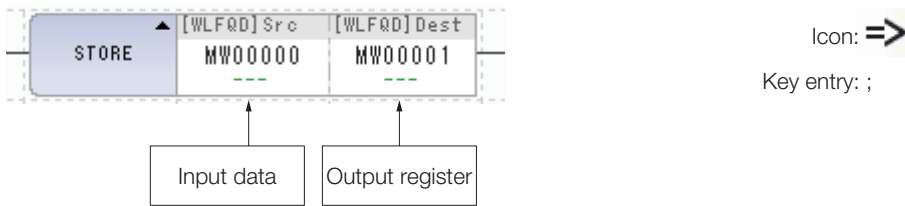
4.3.1 Store (STORE)

The input data is stored in the output register.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	x	○	○	○	○	○	○	○	○
Dest (Output register)	x	○*	○*	○*	○*	○*	○*	○	x

* C and # registers cannot be used.

Programming Example

In the following programming example, the input data is stored in the output register.

- Storing the Input Data, an Integer Value of 12345, in the MW00000 Output Register



- Storing the Input Data, a Real Value of 123.45, in the MW00000 Output Register



- Storing the Input Data, a Double-length Integer Value of 89ABCDEF Hex, in the MW00000 Output Register

The lower word of the double-length integer, -12,817 (CDEF hex) is stored in MW0000.



- Storing the Input Data, an Integer Value of 1234, in the MF00000 Output Register



Information When performing operations with different data types, the result of the operation will depend on the data type of the output register.

Chapter 3 Registers – Precautions for Operations Using Different Data Types on page 3-10

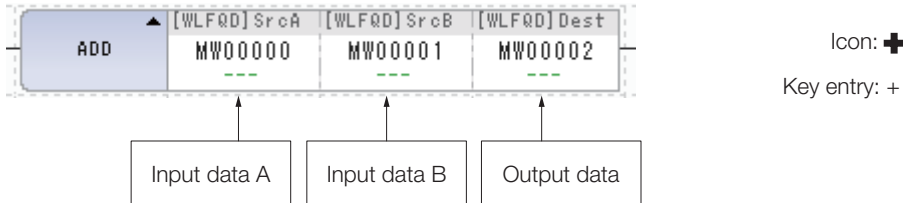
4.3.2 Add (ADD (+))

Input data A and input data B are added and the result is stored in the output data.
An operation error occurs if the result produces an overflow or underflow.



Format

The format of this instruction is shown below.



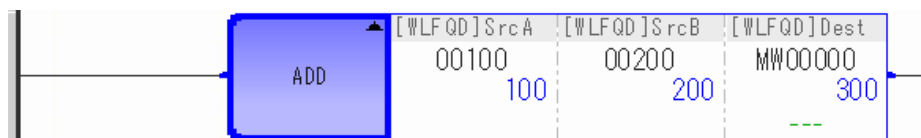
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
SrcA (Input data A)	×	○	○	○	○	○	×	○	○
SrcB (Input data B)	×	○	○	○	○	○	×	○	○
Dest (Output data)	×	○*	○*	○*	○*	○*	×	○	×

* C and # registers cannot be used.

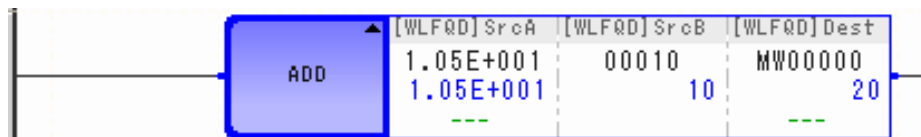
Programming Example

In the following programming example, input data A and input data B are added and the result is stored in the output data.

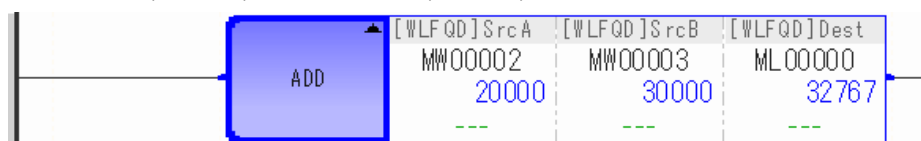
- Storing the Output Data in MW00000 When Input Data A Is 100 and Input Data B Is 200
 $100 + 200 \rightarrow \text{MW00000} = 300$



- Storing the Output Data in MW00000 When Input Data A Is 10.5 and Input Data B Is 10
 $10.5 + 10 \rightarrow \text{MW00000} = 20$ (when truncating below the decimal point is set)



- Storing the Output Data in ML00000 When Input Data A in MW00002 Is 20,000 and Input Data B in MW00003 Is 30,000
 $\text{MW00002} (20,000) + \text{MW00003} (30,000) \rightarrow \text{ML00000} = 32,767^*$



* In the example given above, an overflow error occurs because both input data A and B are integers, which limits the result to a number within the range for integers.

Additional Information

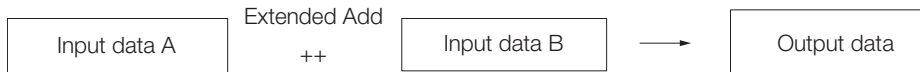
With integer operations, an overflow operation error occurs if the result exceeds 32,767 and an underflow operation error occurs if the result is less than -32,768.

With double-length integer operations, an overflow operation error occurs if the result exceeds 2,147,483,647 and an underflow operation error occurs if the result is less than -2,147,483,648.

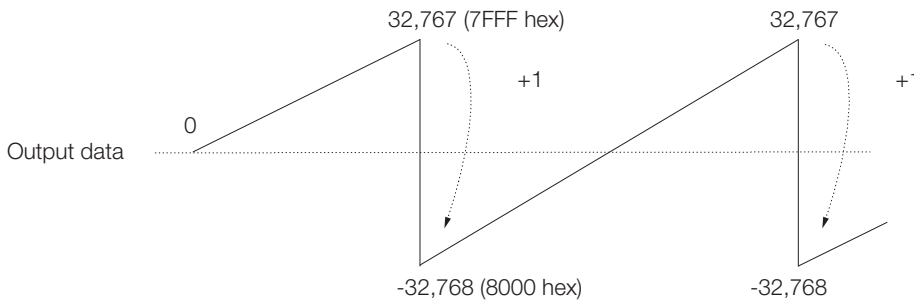
Information When performing operations with different data types, the result of the operation will depend on the data type of the output register.
 Chapter 3 Registers – Precautions for Operations Using Different Data Types on page 3-10
 Normally, addition and subtraction instructions (+, -, ++, and --) involving double-length integers are performed as 32-bit operations.
 However, these instructions are performed as 64-bit operations if they are used to correct the remainder produced by an immediately preceding MUL (x) instruction and are immediately followed by a DIV (+) instruction.

4.3.3 Extended Add (ADDX (++))

Input data A and input data B are added and the result is stored in the output data.
 Overflows are not treated as operation errors. Operation continues from the maximum value in the negative direction.
 Underflows are not treated as operation errors. Operation continues from the maximum value in the positive direction.



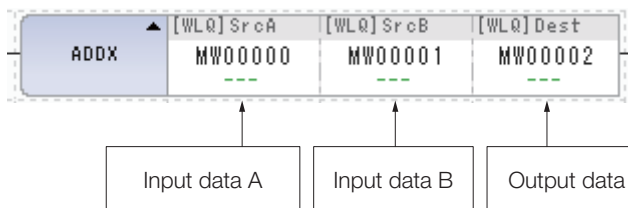
The following figure shows how the output data changes.



- Note: 1. In example shown above, the output data is integer data. With double-length integers, adding 1 to 2,147,483,647 (7FFFFFFF hex) results in -2,147,483,648 (80000000 hex).
- 2. Unlike operations for the ADD (+), SUB (-), or EXPRESSION instructions, overflows and underflows do not occur.

Format

The format of this instruction is shown below.



Icon: ++
 Key entry: ++

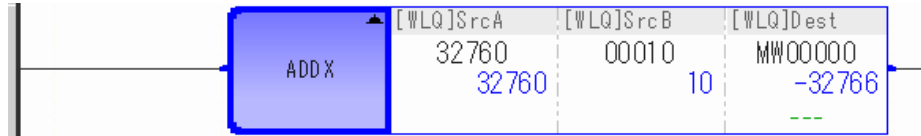
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
SrcA (Input data A)	x	o	o	o	x	x	x	o	o
SrcB (Input data B)	x	o	o	o	x	x	x	o	o
Dest (Output data)	x	o*	o*	o*	x	x	x	o	x

* C and # registers cannot be used.

Programming Example

In the following programming example, input data A and input data B are extended-added, and the result is stored in the output data.

- Storing the Output Data in MW00000 When Input Data A Is 32,760 and Input Data B Is 10
 $32,760 ++ 10 \rightarrow MW00000 = -32,766$

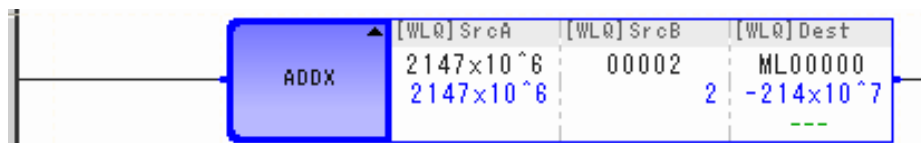


- Storing the Output Data in ML00000 When Input Data A in MW00002 Is 20,000 and Input Data B in MW00003 Is 30,000
 $MW00002 (20,000) ++ MW00003 (30,000) \rightarrow ML00000 = -15,536^*$

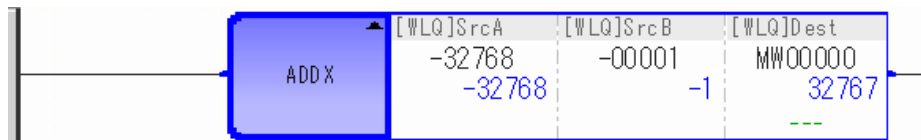


* In the example given above, ML00000 does not equal 50,000 because both input data A and B are integers, which limits the result to a number within the range for integers.

- Storing the Output Data in ML00000 When Input Data A Is 2,147,483,647 and Input Data B Is 2
 $2,147,483,647 ++ 2 \rightarrow ML00000 = -241,783,647$



- Storing the Output Data in MW00000 When Input Data A Is -32,768 and Input Data B Is -1
 $-32,768 ++ -1 \rightarrow MW00000 = 32,767$



Information When performing operations with different data types, the result of the operation will depend on the data type of the output register.

 Chapter 3 Registers – Precautions for Operations Using Different Data Types on page 3-10

Normally, addition and subtraction instructions (+, -, ++, and --) involving double-length integers are performed as 32-bit operations.

However, these instructions are performed as 64-bit operations if they are used to correct the remainder produced by an immediately preceding MUL (x) instruction and are immediately followed by a DIV (+) instruction.

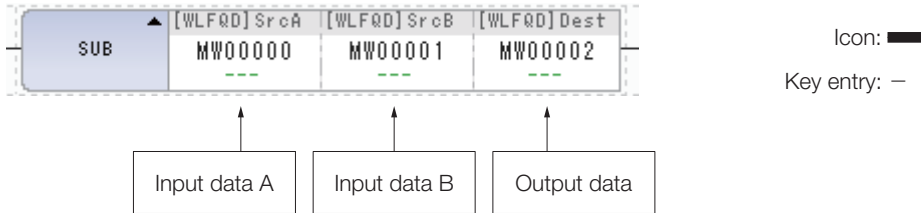
4.3.4 Subtract (SUB (-))

Input data B is subtracted from input data A and the result is stored in the output data. An operation error occurs if the result produces an overflow or underflow.



Format

The format of this instruction is shown below.



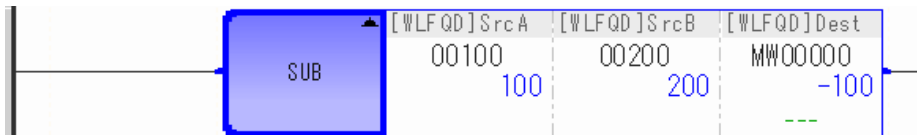
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
SrcA (Input data A)	x	○	○	○	○	○	x	○	○
SrcB (Input data B)	x	○	○	○	○	○	x	○	○
Dest (Output data)	x	○*	○*	○*	○*	○*	x	○	x

* C and # registers cannot be used.

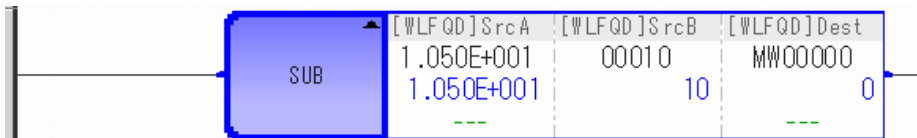
Programming Example

In the following programming example, input data B is subtracted from input data A and the result is stored in the output data.

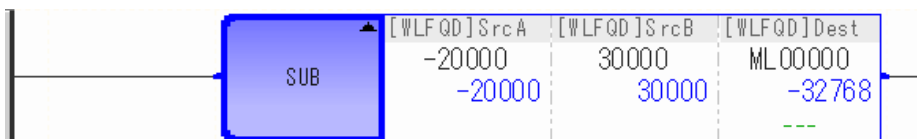
- Storing the Output Data in MW00000 When Input Data A Is 100 and Input Data B Is 200
 $100 - 200 \rightarrow MW00000 = -100$



- Storing the Output Data in MW00000 When Input Data A Is 10.5 and Input Data B Is 10
 $10.5 - 10 \rightarrow MW00000 = 0$ (when truncating below the decimal point is set)



- Storing the Output Data in ML00000 When Input Data A in MW00002 Is -20,000 and Input Data B in MW00003 Is 30,000
 $MW00002 (-20,000) - MW00003 (30,000) \rightarrow ML00000 = -32,768^*$



* In the example given above, an underflow error occurs because both input data A and B are integers, which limits the result to a number within the range for integers.

Additional Information

With integer operations, an overflow operation error occurs if the result exceeds 32,767 and an underflow operation error occurs if the result is less than -32,768.

With double-length integer operations, an overflow operation error occurs if the result exceeds 2,147,483,647 and an underflow operation error occurs if the result is less than -2,147,483,648.

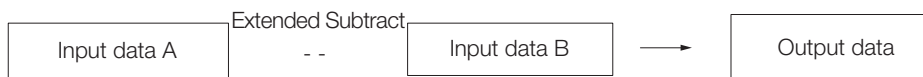
Information When performing operations with different data types, the result of the operation will depend on the data type of the output register.
 Chapter 3 Registers – Precautions for Operations Using Different Data Types on page 3-10
 Normally, addition and subtraction instructions (+, –, ++, and – –) involving double-length integers are performed as 32-bit operations.
 However, these instructions are performed as 64-bit operations if they are used to correct the remainder produced by an immediately preceding MUL (x) instruction and are immediately followed by a DIV (+) instruction.

4.3.5 Extended Subtract (SUBX (– –))

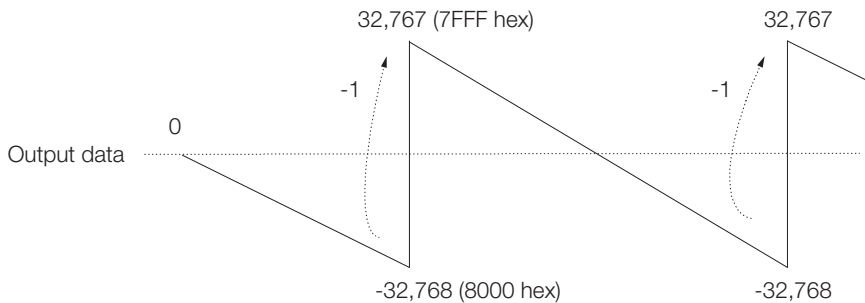
Input data B is subtracted from input data A and the result is stored in the output data.

Overflows are not treated as operation errors. Operation continues from the maximum value in the negative direction.

Underflows are not treated as operation errors. Operation continues from the maximum value in the positive direction.



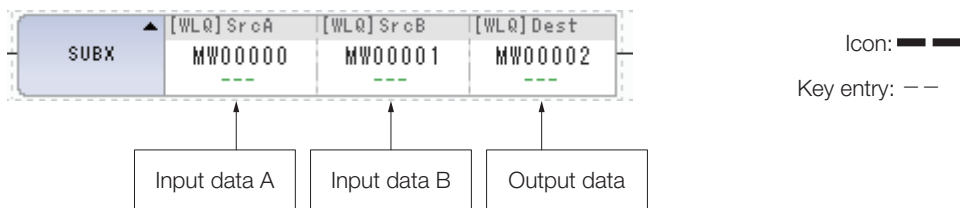
The following figure shows how the output data changes.



Note: 1. In example shown above, the output data is integer data. With double-length integers, subtracting 1 from -2,147,483,648 (80000000 hex) results in 2,147,483,647 (7FFFFFFF hex).
 2. Unlike operations for the ADD (+), SUB (-), or EXPRESSION instructions, overflows and underflows do not occur.

Format

The format of this instruction is shown below.



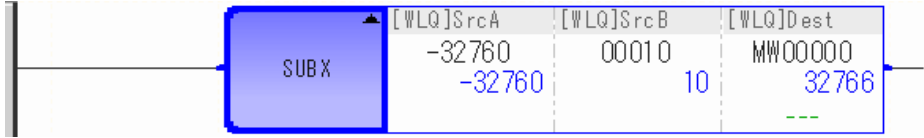
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
SrcA (Input data A)	×	○	○	○	×	×	×	○	○
SrcB (Input data B)	×	○	○	○	×	×	×	○	○
Dest (Output data)	×	○*	○*	○*	×	×	×	○	×

* C and # registers cannot be used.

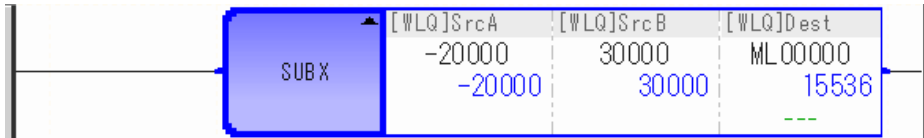
Programming Example

In the following programming example, input data B is extended-subtracted from input data A and the result is stored in the output data.

- Storing the Output Data in MW00000 When Input Data A Is -32,760 and Input Data B Is 10
 $-32,760 - - 10 \rightarrow MW00000 = 32,766$

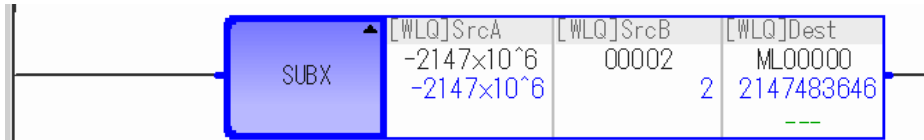


- Storing the Output Data in ML00000 When Input Data A in MW00002 Is -20,000 and Input Data B in MW00003 Is 30,000
 $MW00002 (-20,000) - - MW00003 (30,000) \rightarrow ML00000 = 15,536^*$

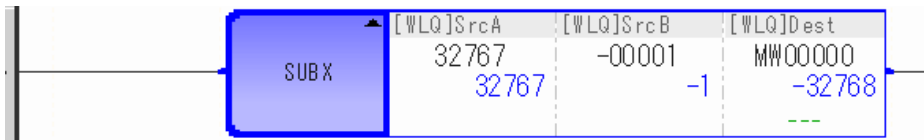


* In the example given above, ML00000 does not equal -50,000 because both input data A and B are integers, which limits the result to a number within the range for integers.

- Storing the Output Data in ML00000 When Input Data A Is -2,147,483,648 and Input Data B Is 2
 $-2,147,483,648 - - 2 \rightarrow ML00000 = 241,783,646$



- Storing the Output Data in MW00000 When Input Data A Is 32,767 and Input Data B Is -1
 $32,767 - - -1 \rightarrow MW00000 = -32,768$



Information

When performing operations with different data types, the result of the operation will depend on the data type of the output register.

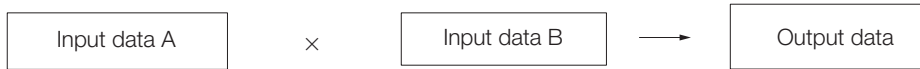
Chapter 3 Registers – Precautions for Operations Using Different Data Types on page 3-10

Normally, addition and subtraction instructions (+, -, ++, and --) involving double-length integers are performed as 32-bit operations.

However, these instructions are performed as 64-bit operations if they are used to correct the remainder produced by an immediately preceding MUL (x) instruction and are immediately followed by a DIV (÷) instruction.

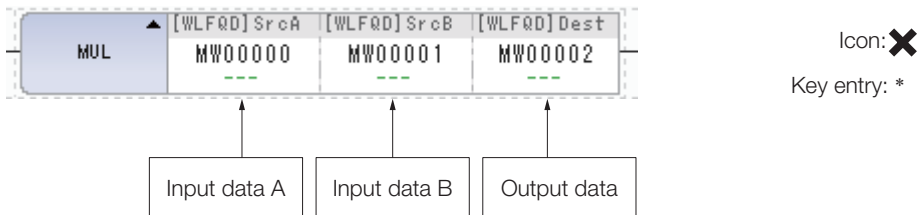
4.3.6 Multiply (MUL (x))

Input data A and input data B are multiplied and the result is stored in the output data.



Format

The format of this instruction is shown below.



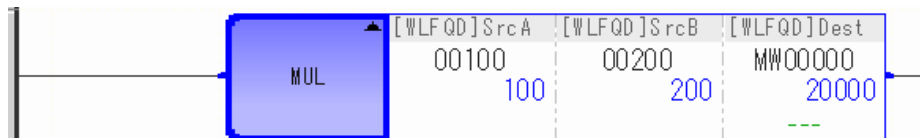
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
SrcA (Input data A)	×	○	○	○	○	○	×	○	○
SrcB (Input data B)	×	○	○	○	○	○	×	○	○
Dest (Output data)	×	○*	○*	○*	○*	○*	×	○	×

* C and # registers cannot be used.

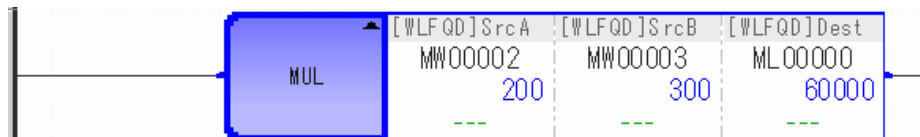
Programming Example

In the following programming example, input data A and input data B are multiplied and the result is stored in the output data.

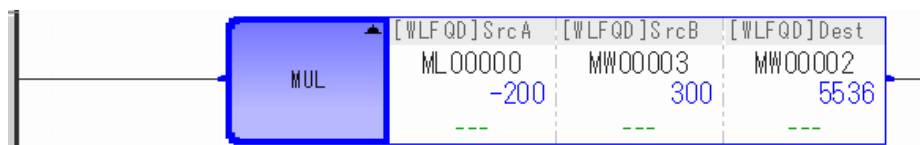
- Storing the Output Data in MW00000 When Input Data A Is 100 and Input Data B Is 200
 $100 \times 200 \rightarrow \text{MW00000} = 20,000$



- Storing the Output Data in ML00000 When Input Data A in MW00002 Is 200 and Input Data B in MW00003 Is 300
 $\text{MW00002} (200) \times \text{MW00003} (300) \rightarrow \text{ML00000} = 60,000$



- Storing the Output Data in MW00002 When Input Data A in ML00000 Is -200 and Input Data B in MW00003 Is 300
 $-200 \times 300 \rightarrow \text{MW00002} = 5,536^*$



* The input data contains a double-length integer, so this operation is performed as a double-length integer operation. However, the output data is integer data, so if the operation result exceeds the range for integers, the lower 16-bits of the original operation result will be stored in the output data.

Information When performing operations with different data types, the result of the operation will depend on the data type of the output register.

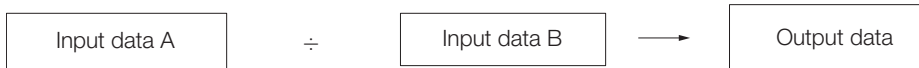
Chapter 3 Registers – Precautions for Operations Using Different Data Types on page 3-10

Normally, addition and subtraction instructions (+, −, ++, and −−) involving double-length integers are performed as 32-bit operations.

However, these instructions are performed as 64-bit operations if they are used to correct the remainder produced by an immediately preceding MUL (×) instruction and are immediately followed by a DIV (÷) instruction.

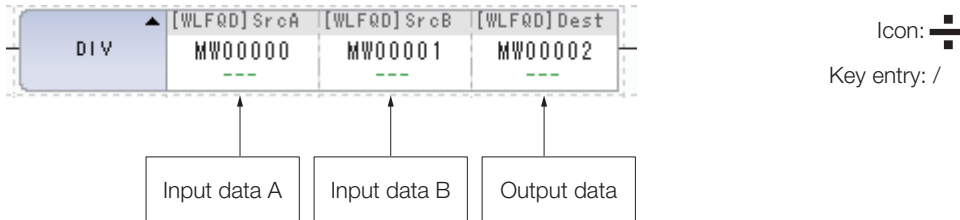
4.3.7 Divide (DIV (÷))

Input data A is divided by input data B and the result is stored in the output data.



Format

The format of this instruction is shown below.



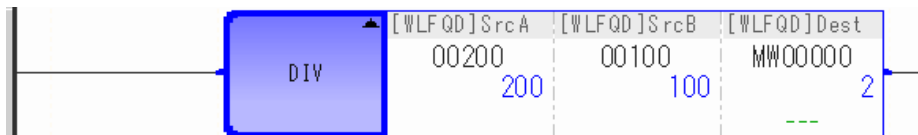
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
SrcA (Input data A)	×	○	○	○	○	○	×	○	○
SrcB (Input data B)	×	○	○	○	○	○	×	○	○
Dest (Output data)	×	○*	○*	○*	○*	○*	×	○	×

* C and # registers cannot be used.

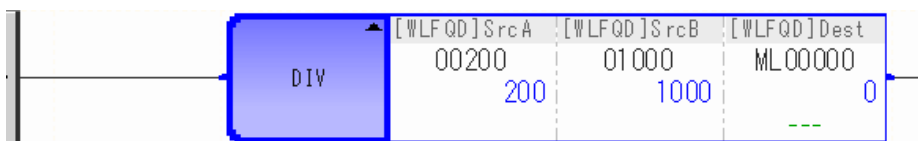
Programming Example

In the following programming example, input data A is divided by input data B and the result is stored in the output data.

- Storing the Output Data in MW00000 When Input Data A Is 200 and Input Data B Is 100
 $200 \div 100 \rightarrow MW00000 = 2$



- Storing the Output Data in ML00000 When Input Data A Is 200 and Input Data B Is 1,000
 $200 \div 1,000 \rightarrow ML00000 = 0$



- Storing the Output Data in MF00000 When Input Data A Is 200 and Input Data B Is 1,000
 $200 \div 1,000 \rightarrow MF00000 = 0.2$



Information When performing operations with different data types, the result of the operation will depend on the data type of the output register.

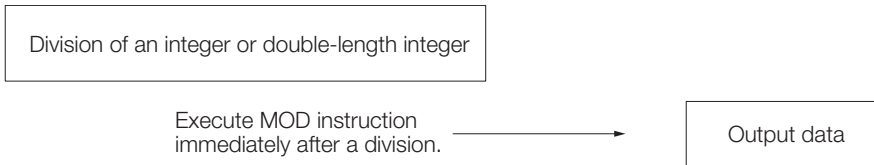
Chapter 3 Registers – Precautions for Operations Using Different Data Types on page 3-10

Normally, addition and subtraction instructions (+, -, ++, and --) involving double-length integers are performed as 32-bit operations.

However, these instructions are performed as 64-bit operations if they are used to correct the remainder produced by an immediately preceding MUL (x) instruction and are immediately followed by a DIV (+) instruction.

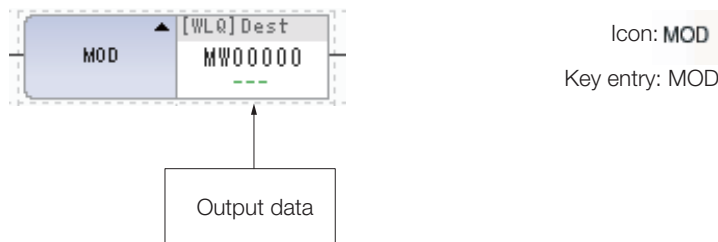
4.3.8 Integer Remainder (MOD)

The remainder of the immediately preceding integer or double-length integer division is stored in the output data. The MOD instruction must be executed immediately after the DIV (+) instruction. If the MOD instruction is executed at any other time, the operation result obtained before the next numeric operation instruction will be invalid.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Dest (Output data)	x	○*	○*	○*	x	x	x	○	x

* C and # registers cannot be used.

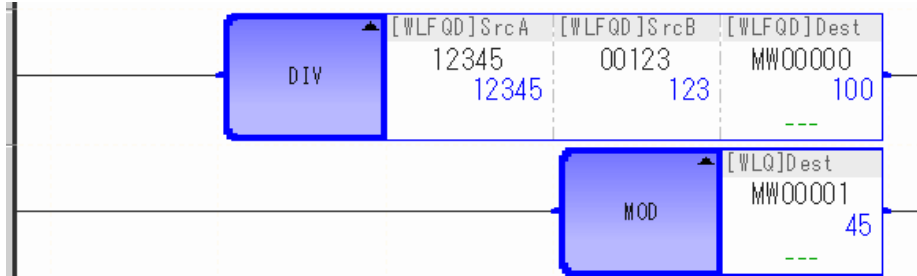
Programming Example

In the following programming example, input data A is divided by input data B and the remainder is stored in the output data.

- If the immediately preceding division is as follows:

$$12,345 \div 123 \rightarrow MW00000 = 100$$

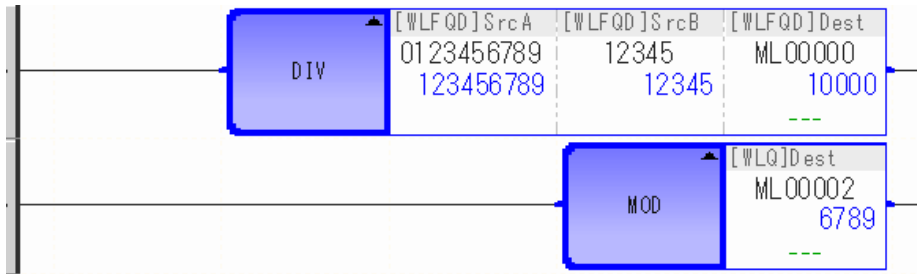
And then the MOD instruction is executed immediately afterward $\rightarrow MW00001 = 45$



- If the immediately preceding division is as follows:

$$123,456,789 \div 12,345 \rightarrow ML00000 = 10,000$$

And then the MOD instruction is executed immediately afterward $\rightarrow ML00002 = 6,789$



Information When performing operations with different data types, the result of the operation will depend on the data type of the output register.

Chapter 3 Registers – Precautions for Operations Using Different Data Types on page 3-10

4.3.9 Real Remainder (REM)

The remainder from a real number division is stored in the output data. Here, the remainder refers to the remainder obtained by repeatedly subtracting the base value from the input data.

Specifically, the value obtained by subtracting the base value from the input data n number of times (input data - base value $\times n$) is output when it becomes less than the base value.

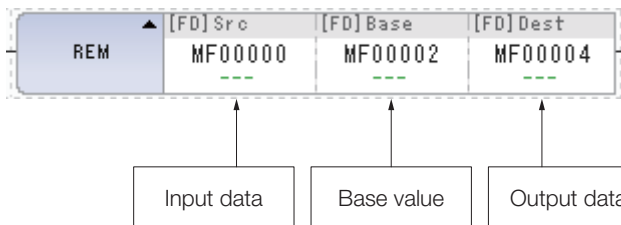


The output data is computed by using the first value of n that satisfies the following formula when the value of n is incremented from 0, 1, 2, 3, etc.

(Input data - Base value $\times n$) < Base value

Format

The format of this instruction is shown below.



Icon: **REM**

Key entry: REM

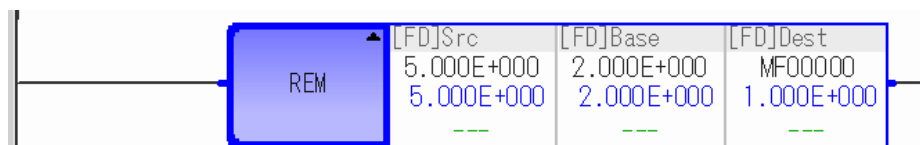
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	×	×	×	×	○	○	×	×	○
Base (Base value)	×	×	×	×	○	○	×	×	○
Dest (Output data)	×	×	×	×	○*	○*	×	○	×

* C and # registers cannot be used.

Programming Example

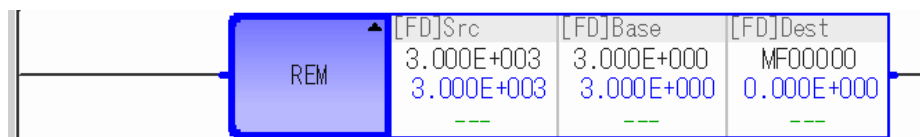
In the following programming example, the base value is subtracted from the input data n times and the remainder is stored in the output data.

- Storing the Output Data in MF00000 When the Input Data Is 5.0 and the Base Value Is 2.0.
 $5.0 - 2.0 - 2.0 = 1.0 < \text{Base (2.0)} \rightarrow \text{MF00000} = 1.0$



- Storing the Output Data in MF00000 When the Input Data Is 3,000.0 and the Base Value Is 3.0.

$3,000.0 - 3.0 - 3.0 \dots = 0.0 < \text{Base (3.0)} \rightarrow \text{MF00000} = 0.0$

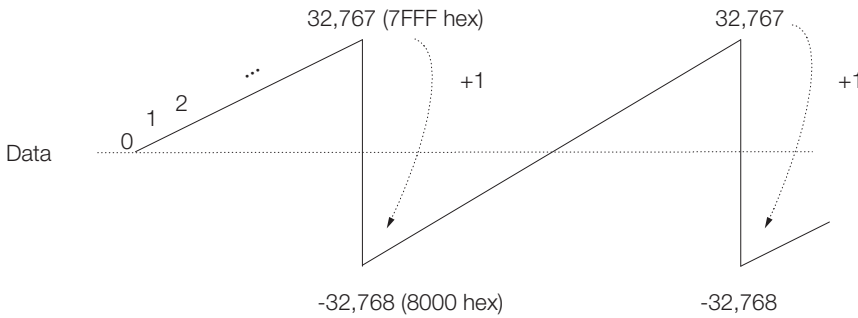


4.3.10 Increment (INC)

A value of 1 is added to the integer or double-length integer data. No overflow or underflow will occur for either an integer or double-length integer. This performs the same calculation as the ADDX (++) instruction.



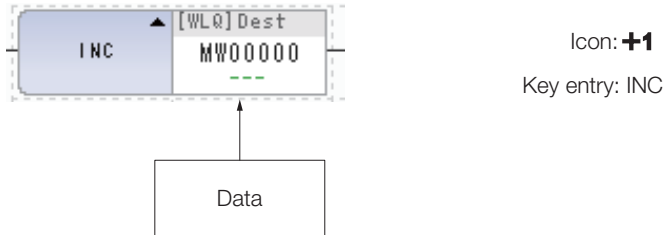
The following figure shows how the data changes when the INC instruction is executed.



Note: In example shown above, the data is an integer. With double-length integers, adding 1 to 2,147,483,647 (7FFFFFFF hex) results in -2,147,483,648 (80000000 hex).

Format

The format of this instruction is shown below.



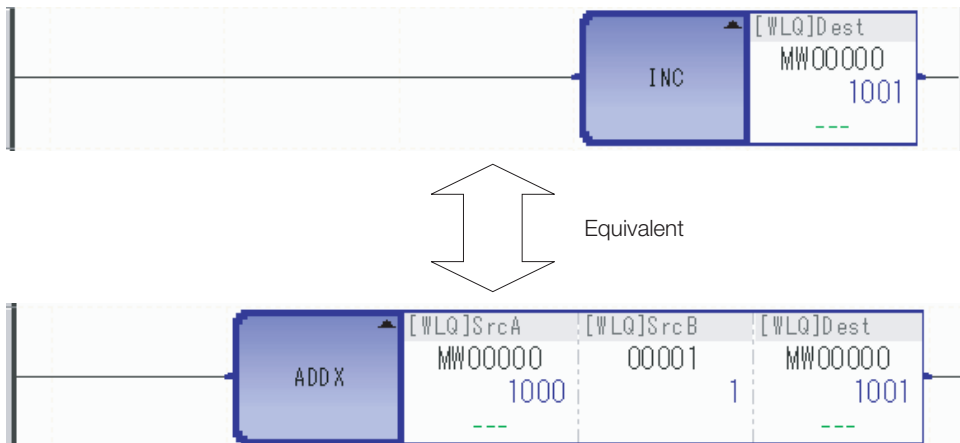
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Dest (Data)	x	o	o	o*	x	x	x	o	x

* C and # registers cannot be used.

Programming Example

The following programming examples demonstrate the usage of the INC instruction and the ADDX (++) instruction.

This is equivalent to adding 1 to the data 1,000 in MW00000 using the ADDX (++) instruction.

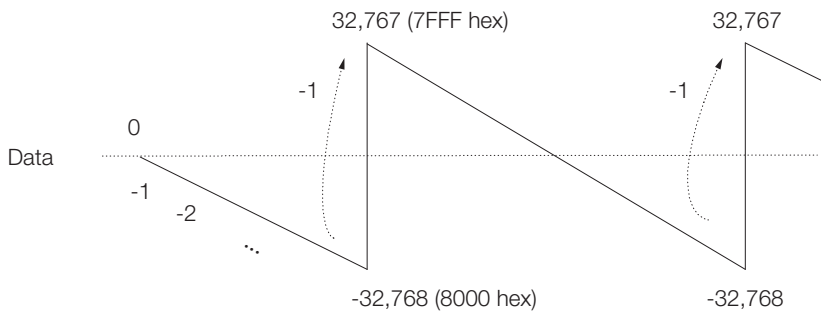


4.3.11 Decrement (DEC)

A value of 1 is subtracted from the integer or double-length integer data. No overflow or underflow will occur for either an integer or double-length integer. This performs the same calculation as the SUBX (– –) instruction.



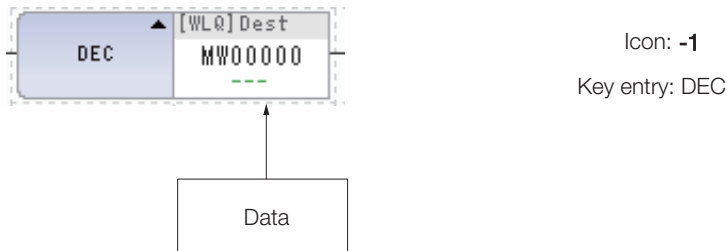
The following figure shows how the data changes when the DEC instruction is executed.



Note: In example shown above, the data is an integer. With double-length integers, subtracting 1 from -2,147,483,648 (80000000 hex) results in 2,147,483,647 (7FFFFFFF hex).

Format

The format of this instruction is shown below.



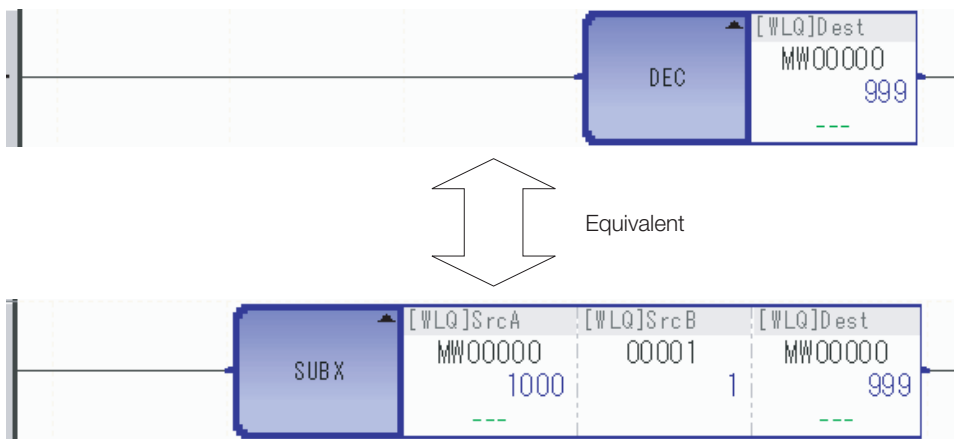
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Dest (Data)	x	o	o	o*	x	x	x	o	x

* C and # registers cannot be used.

Programming Example

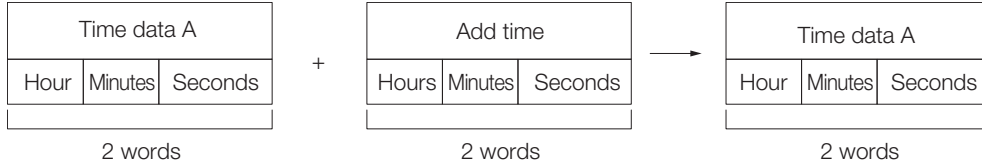
The following programming examples demonstrate the usage of the DEC instruction and the SUBX (– –) instruction.

This is equivalent to subtracting 1 from the data 1,000 in MW00000 using the SUBX (– –) instruction.



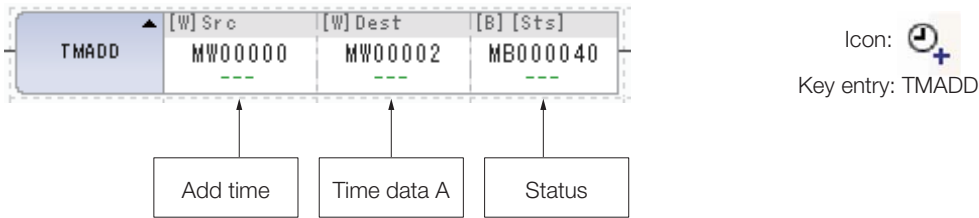
4.3.12 Add Time (TMADD)

A duration (hours/minutes/seconds) is added to a time (hour/minutes/seconds). The add time is added to time data A and the result is stored in time data A. Time data is two words long.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Add time)	×	○*2	×	×	×	×	×	×	×
Dest (Time data A)	×	○*2	×	×	×	×	×	×	×
Sts (Status)*1	○*2	×	×	×	×	×	×	×	×

*1. Optional.

*2. C and # registers cannot be used.

The time data is formatted as shown below.

Offset	Contents	Data Range (BCD)
0	Hour/minutes	Upper byte (hour): 00 to 23 Lower byte (minutes): 00 to 59
1	Seconds	0000 to 0059

If the operation result exceeds any of the data ranges given above, time data A is not updated and the seconds data will be set to 9999, and the status bit will set to 1.

If the operation result is within the ranges, the status bit is set to 0.

Programming Example

The following table gives typical conditions for creating ladder programming that uses the TMADD instruction. The examples show time data A before instruction execution, and the add time.

Time	Time Data A before Execution of Instruction	Add time
Hour/minutes	MW00000 = 0210 hex (2:10)	MW00002 = 0050 hex (0 hours 50 minutes)
Seconds	MW00001 = 0050 hex (50 seconds)	MW00003 = 0020 hex (20 seconds)

In the following programming example, the times are added according to the conditions given above, and the result is stored in time data A.

The screenshot shows a ladder logic program with the following components:

- EXPRESSION window (top):**

```
// time data A (before added)
MW00000=0x0210; // 2:10
528=528
MW00001=0x0050; // 50 second
80=80

// time to be added
MW00002=0x0050; // 0 hour 50min
80=80
MW00003=0x0020; // 20 second
32=32
```
- TMADD instruction:**

[W]Src	[W]Dest	[B] [Sts]
MW00002	MW00000	MB000040
---	---	---
		0
- EXPRESSION window (bottom):**

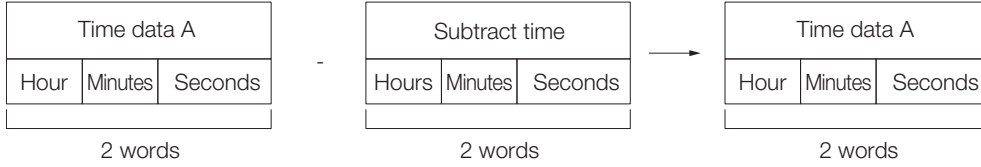
```
// time data (after added)
MW00000=MW00000; // :
769=769
MW00001=MW00001; // second
16=16
```

The result of adding the add time to the value of time data A before instruction execution is shown below.

Time	Time Data A after Execution of Instruction
Hour/minutes	MW00000 = 769 = 0301 hex (3:01)
Seconds	MW00001 = 16 = 0010 hex (10 seconds)

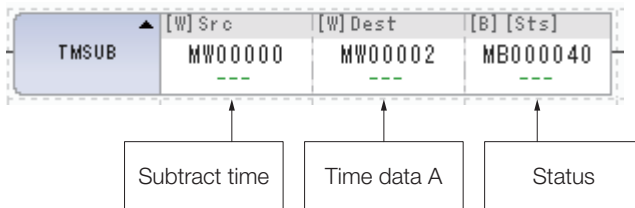
4.3.13 Subtract Time (TMSUB)

A duration (hours/minutes/seconds) is subtracted from a time (hour/minutes/seconds). The subtract time is subtracted from time data A and the result is stored in time data A. Time data is two words long.



Format

The format of this instruction is shown below.



Icon: Key entry: TMSUB

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Subtract time)	×	○*2	×	×	×	×	×	×	×
Dest (Time data A)	×	○*2	×	×	×	×	×	×	×
Sts (Status)*1	○*2	×	×	×	×	×	×	×	×

*1. Optional.

*2. C and # registers cannot be used.

The time data is formatted as shown below.

Offset	Contents	Data Range (BCD)
0	Hour/minutes	Upper byte (hour): 00 to 23 Lower byte (minutes): 00 to 59
1	Seconds	0000 to 0059

If the operation result exceeds any of the data ranges given above, time data A is not updated and the seconds data will be set to 9999, and the status bit will set to 1.

If the operation result is within the ranges, the status bit is set to 0.

Programming Example

The following table gives typical conditions for creating ladder programming that uses the TMSUB instruction. The examples show time data A before instruction execution, and the subtract time.

Time	Time Data A before Execution of Instruction	Subtract time
Hour/minutes	MW00000 = 0210 hex (2:10)	MW00002 = 0050 hex (0 hours 50 minutes)
Seconds	MW00001 = 0050 hex (50 seconds)	MW00003 = 0020 hex (20 seconds)

In the following programming example, the time is subtracted according to the conditions given above, and the result is stored in time data A.

The screenshot displays a ladder logic program with two 'EXPRESSION' windows and a TMSUB instruction block.

Top EXPRESSION window (before execution):

```
// time data A (before subtracted)
MW00000=0x0210; // 2:10
528=528
MW00001=0x0050; // 50 second
80=80

// time to be subtracted
MW00002=0x0050; // 0 hour 50min
80=80
MW00003=0x0020; // 20 second
32=32
```

TMSUB instruction block:

[W]Src	[W]Dest	[B] [Sts]
MW00002	MW00000	MB000040
---	---	0

Bottom EXPRESSION window (after execution):

```
// time data (after subtracted)
MW00000=MW00000; // :
288=288
MW00001=MW00001; // second
48=48
```

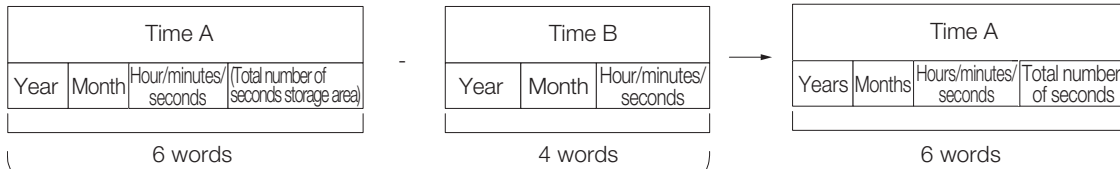
The result of subtracting the subtract time from the value of time data A before instruction execution is shown below.

Time	Time Data A after Execution of Instruction
Hour/minutes	MW00000 = 288 = 0120 hex (1:20)
Seconds	MW00001 = 48 = 0030 hex (30 seconds)

4.3.14 Spend Time (SPEND)

The elapsed time is calculated by subtracting two data items (year/month/day/hour/minutes/seconds). The instruction subtracts time B from time A, which gives the time elapsed from time B to time A and the result is stored in time A.

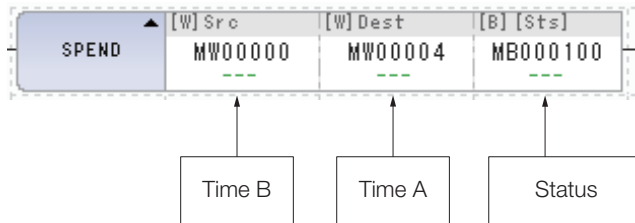
Time data is four words long.



The time elapsed from time B to time A is calculated.

Format

The format of this instruction is shown below.



Icon: 
Key entry: SPEND

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Time B)	×	○*2	×	×	×	×	×	×	×
Dest (Time A)	×	○*2	×	×	×	×	×	×	×
Sts (Status)*1	○*2	×	×	×	×	×	×	×	×

*1. Optional.

*2. C and # registers cannot be used.

Time B is formatted as shown below.

Offset	Contents	Data Range (BCD)	I/O
0	Year (BCD)	0000 to 0099	IN
1	Month/day (BCD)	Upper byte (month): 01 to 12 Lower byte (day): 01 to 31	IN
2	Hour/minutes (BCD)	Upper byte (hour): 00 to 23 Lower byte (minutes): 00 to 59	IN
3	Seconds (BCD)	0000 to 0059	IN

Time A is formatted as shown below.

Offset	Contents	Data Range (BCD)	I/O
0	Year (BCD)	0000 to 0099	IN/OUT
1	Month/day (BCD)	Upper byte (month): 01 to 12 Lower byte (day): 01 to 31	IN/OUT
2	Hour/minutes (BCD)	Upper byte (hour): 00 to 23 Lower byte (minutes): 00 to 59	IN/OUT
3	Seconds (BCD)	0000 to 0059	IN/OUT
4	Total number of seconds	Operation result of years, months, days, hours, minutes, and seconds converted into seconds (double-length integer).	IN/OUT
5			

If the operation result exceeds any of the data ranges given above, time A is not updated and the seconds data will be set to 9999, and the status bit will be set to 1.

If the operation result is within the ranges, the status bit is set to 0.

Information A year is calculated as 365 days. Leap years are not supported.
The number of months is not calculated. Only the number of days is calculated.

Programming Example

The following table gives typical conditions for creating ladder programming that uses the SPEND instruction.

The following list shows time A (November 20, 2010, 02:10:50) before instruction execution, and time B (October 10, 2009, 00:50:20).

	Time A before Execution of Instruction	Time B
Year	MW00000 = 0010 hex (2010)	MW00006 = 0009 hex (2009)
Month/day	MW00001 = 1120 hex (November 20)	MW00007 = 1010 hex (October 10)
Hour/minutes	MW00002 = 0210 hex (2:10)	MW00008 = 0050 hex (0:50)
Seconds	MW00003 = 0050 hex (50 seconds)	MW00009 = 0020 hex (20 seconds)

The screenshot displays a ladder logic program with two expression windows and a SPEND instruction.

Top Expression Window:

```
// time A (before subtracted)
MW00000=0x0010; // 2010
16=16
MW00001=0x1120; // Nov. 20th
4384=4384
MW00002=0x0210; // 2:10
528=528
MW00003=0x0050; // 50 second
80=80

// time B
MW00006=0x0009; // 2009
9=9
MW00007=0x1010; // Oct. 10th
4112=4112
MW00008=0x0050; // 0:50
80=80
MW00009=0x0020; // 20 second
32=32
```

SPEND Instruction:

[W]Src	[W]Dest	[B] [Sts]
MW00006	MW00000	MB000040
---	---	0

Bottom Expression Window:

```
// time A (after subtracted)--> spend time
MW00000=MW00000; // year
1=1
MW00001=MW00001; // month day
65=65
MW00002=MW00002; // hour min
288=288
MW00003=MW00003; // second
48=48
ML00004=ML00004; // total second
35083230=35083230
```

4.3.15 Invert Sign (INV)

The execution result of this SPEND instruction example is shown below.

	Time A after Execution of Instruction
Years	MW00000 = 1 = 0001 hex (1 year)
Months/days	MW00001 = 65 = 0041 hex (0 months, 41 days)
Hours/minutes	MW00002 = 288 = 0120 hex (1 hour, 20 minutes)
Seconds	MW00003 = 48 = 0030 hex (30 seconds)
Total number of seconds	ML00004 = 35083230

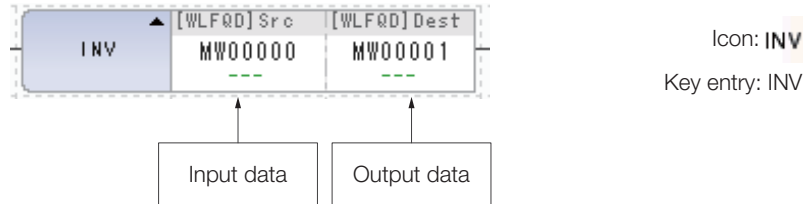
4.3.15 Invert Sign (INV)

The sign of the input data is inverted and the result is stored in the output data.



Format

The format of this instruction is shown below.



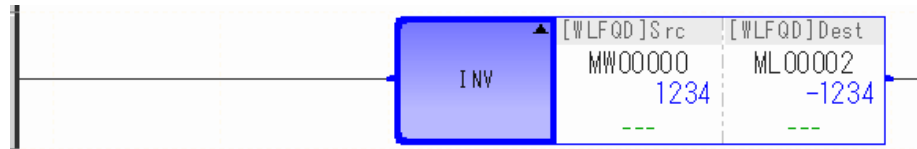
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	×	○	○	○	○	○	×	○	○
Dest (Output data)	×	○*	○*	○*	○*	○*	×	○	×

* C and # registers cannot be used.

Programming Example

In the following programming example, the INV instruction inverts the sign of 1,234 in input data A in MW00000 and stores the result in the output data in ML00002.

$-1 \times \text{MW00000}(1234) \rightarrow \text{ML00002} = -1234$



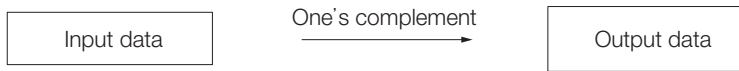
Information

When performing operations with different data types, the result of the operation will depend on the data type of the output register.

Chapter 3 Registers – Precautions for Operations Using Different Data Types on page 3-10

4.3.16 One's Complement (COM)

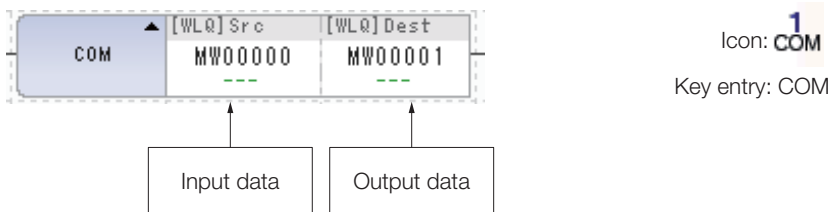
The one's complement of the input data is stored in the output data.



Note: This instruction inverts the 0's and 1's in the binary representation of the input data and stores the result in the output data.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	×	○	○	○	×	×	×	○	○
Dest (Output data)	×	○*	○*	○*	×	×	×	○	×

* C and # registers cannot be used.

Programming Example

In the following programming example, the one's complement of -3,856 (F0F0 hex) in the input data in MW00000 is stored in the output data in MW00001.

MW00000 = -3,856 (F0F0 hex) → MW00001 = 3,855 (0F0F hex)



Information

When performing operations with different data types, the result of the operation will depend on the data type of the output register.

Chapter 3 Registers – Precautions for Operations Using Different Data Types on page 3-10

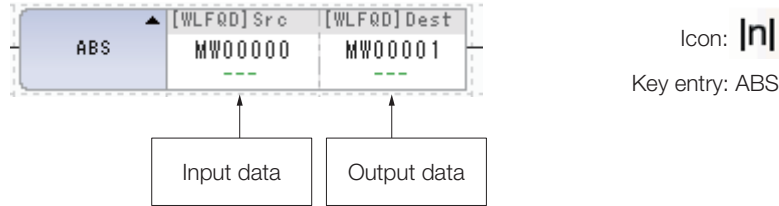
4.3.17 Absolute Value (ABS)

The absolute value of the input data is stored in the output data.



Format

The format of this instruction is shown below.



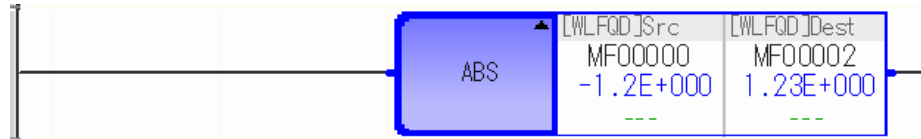
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	x	○	○	○	○	○	x	○	○
Dest (Output data)	x	○*	○*	○*	○*	○*	x	○	x

* C and # registers cannot be used.

Programming Example

In the following programming example, the absolute value of -1.23 in the input data in MF00000 is stored in the output data in MF00002.

$$| \text{MF00000} (-1.23) | \rightarrow \text{MF00002} = 1.23$$



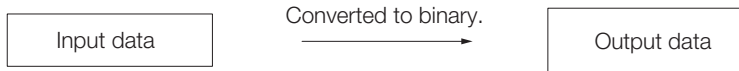
Information When performing operations with different data types, the result of the operation will depend on the data type of the output register.

Chapter 3 Registers – Precautions for Operations Using Different Data Types on page 3-10

4.3.18 Binary Conversion (BIN)

The value of the input data is converted from BCD data to binary data and stored in the output data.

If the input data is not BCD data, such as 123F hex, the result of the binary conversion will be incorrect.



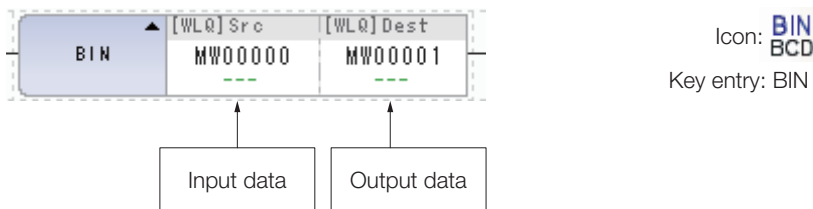
Note: The output data is computed as shown below when the input BCD data is abcd.

$$\text{Output data} = (a \times 1,000) + (b \times 100) + (c \times 10) + d$$

Information Input and output data are always displayed in decimal notation.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	×	○	○	○	×	×	×	○	○
Dest (Output data)	×	○*	○*	○*	×	×	×	○	×

* C and # registers cannot be used.

Programming Example

In the following example, the BCD data (1234 hex (4,660)) in input data A in MW00000 is converted to binary data (displayed in decimal notation as 1,234) and stored as the output data in MW00001.

$$\text{MW00000} = 1234 \text{ hex} : (1 \times 1,000) + (2 \times 100) + (3 \times 10) + 4 \rightarrow \text{MW00001} = 1,234$$



Information When performing operations with different data types, the result of the operation will depend on the data type of the output register.

Chapter 3 Registers – Precautions for Operations Using Different Data Types on page 3-10

4.3.19 BCD Conversion (BCD)

The input data is converted from binary data to BCD data and stored in the output data. If the input data is greater than 9,999, or a negative value, the result will be incorrect.



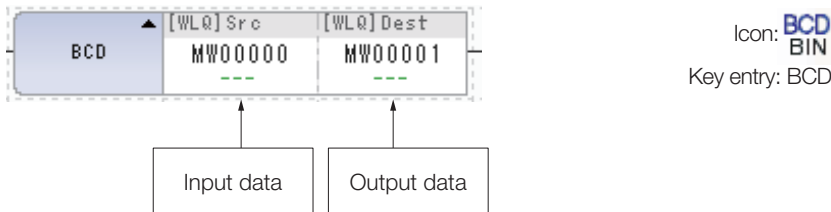
Note: The output data is computed as shown below when the input decimal data is abcd.

$$\text{Output data} = (a \times 4096) + (b \times 256) + (c \times 16) + d$$

Information Input and output data are always displayed in decimal notation.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	x	○	○	○	x	x	x	○	○
Dest (Output data)	x	○*	○*	○*	x	x	x	○	x

* C and # registers cannot be used.

Programming Example

In the following programming example, the binary data (displayed in decimal notation as 1,234) in input data A in MW00000 is converted to BCD data (1234 hex (4,660)) and stored as the output data in MW00001.

$$\text{MW00000} = 1,234 : (1 \times 4,096) + (2 \times 256) + (3 \times 16) + 4 \rightarrow \text{MW00001} = 1234 \text{ hex } (4,660)$$

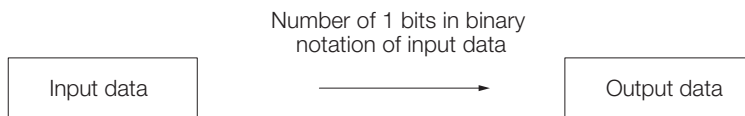


Information When performing operations with different data types, the result of the operation will depend on the data type of the output register.

Chapter 3 Registers – Precautions for Operations Using Different Data Types on page 3-10

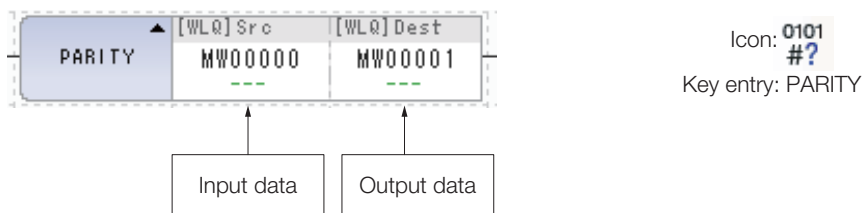
4.3.20 Parity Conversion (PARITY)

The number of bits set to 1 in the input data is calculated in binary notation and stored in the output data.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	×	○	○	○	×	×	×	○	○
Dest (Output data)	×	○*	○*	○*	×	×	×	○	×

* C and # registers cannot be used.

Programming Example

In the following programming example, the number of bits set to 1 in 255 (00FF hex) in the input data A in MW00000 is stored in the output data in MW00001.

Number of 1 bits in MW00000 (0FF hex) = 8 → MW00001 = 8



Information

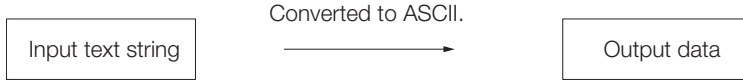
When performing operations with different data types, the result of the operation will depend on the data type of the output register.

Chapter 3 Registers – Precautions for Operations Using Different Data Types on page 3-10

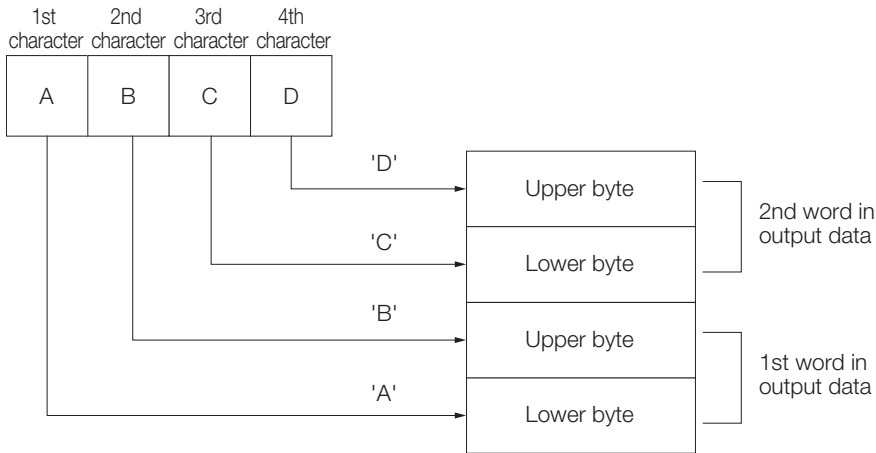
4.3.21 ASCII Conversion 1 (ASCII)

The input text string is converted to ASCII and stored in the output data. The text string is case sensitive.

The input text string can contain up to 32 characters (16 words).



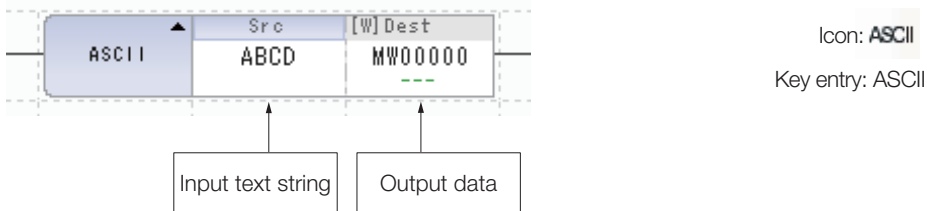
The ASCII value for each character in the input text string is stored as shown below.



Note: If the text string contains an odd number of characters, the upper byte of the last word will be set to zeros.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input text string)	x*1								
Dest (Output data)	x	0*2	x	x	x	x	x	x	x

*1. ASCII text

*2. C and # registers cannot be used.

Programming Example

In the following programming example, the input string "Hello" is converted to ASCII and stored in the output data in MW00000.

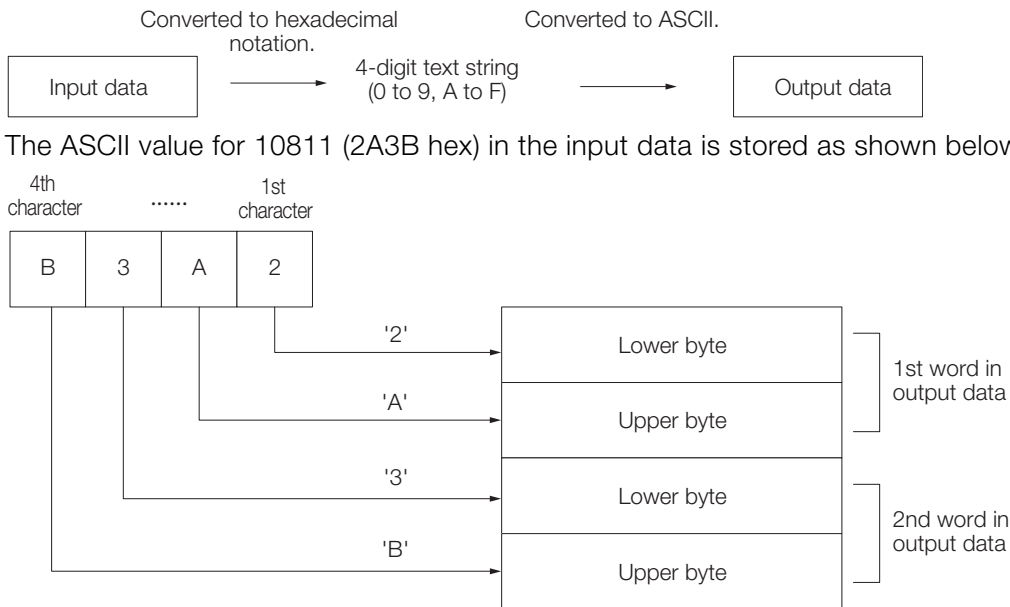


The ASCII values are stored as given in the following table.

Address	ASCII Value	Character
MW00000 (lower byte)	48 hex	H
MW00000 (upper byte)	65 hex	e
MW00001 (lower byte)	6 C hex	l
MW00001 (upper byte)	6 C hex	l
MW00002 (lower byte)	6 F hex	o
MW00002 (upper byte)	0	-

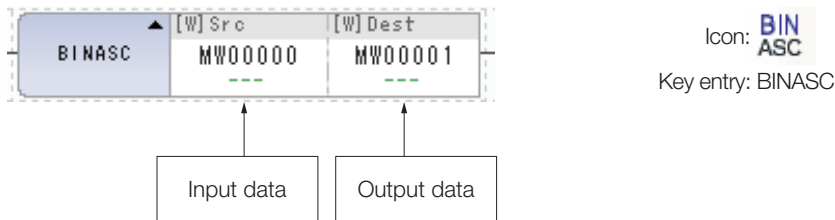
4.3.22 ASCII Conversion 2 (BINASC)

The 16-bit binary data stored in the 1-word input data is converted to four-digit hexadecimal ASCII and stored in the 2-word output data.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	×	○	×	×	×	×	×	×	○
Dest (Output data)	×	○*	×	×	×	×	×	×	×

* C and # registers cannot be used.

Programming Example

In the following programming example, 10,811 (2A3B hex) in the input data is converted to ASCII and stored in the output data in MW00000.



The ASCII values are stored as given in the following table.

Address	ASCII Value	Character
MW00000 (lower byte)	32 hex	2
MW00000 (upper byte)	41 hex	A
MW00001 (lower byte)	33 hex	3
MW00001 (upper byte)	42 hex	B

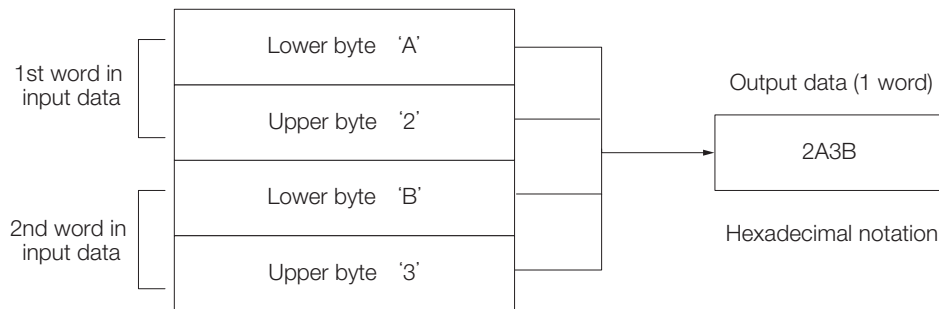
4.3.23 ASCII Conversion 3 (ASCBIN)

The value given as a 4-digit hexadecimal ASCII and stored in the 2-word input data is converted to 16-bit binary data and stored in a 1-word output data.

ASCII converted to binary data.

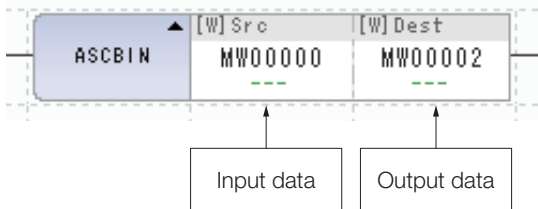


The following figure shows the output data when the first word of the input data is 4132 hex ('2' 'A'), and the second word is 4232 hex ('3' 'B').



Format

The format of this instruction is shown below.



Icon: **ASC**
BIN
Key entry: ASCBIN

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	×	○	×	×	×	×	×	×	×
Dest (Output data)	×	○*	×	×	×	×	×	×	×

* C and # registers cannot be used.

Programming Example

In the following programming example, the ASCBIN instruction is used to store the input data in MW00000 in the output data in MW00002.



The ASCII values are stored as given in the following table.

Address	ASCII Value	Character
MW00000 (lower byte)	32 hex	2
MW00000 (upper byte)	41 hex	A
MW00001 (lower byte)	33 hex	B
MW00001 (upper byte)	42 hex	3

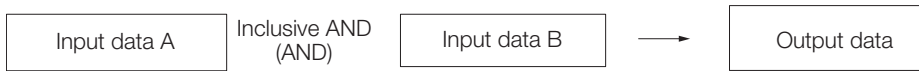
The output data in MW00000 is set to 10,811 (2A3B hex).

4.4 Logic Operations and Comparison Instructions

4.4.1 Inclusive AND (AND)

A logical AND operation is performed on input data A and input data B and the result is stored in the output data.

This instruction can be used only with integer or double-length integer data.

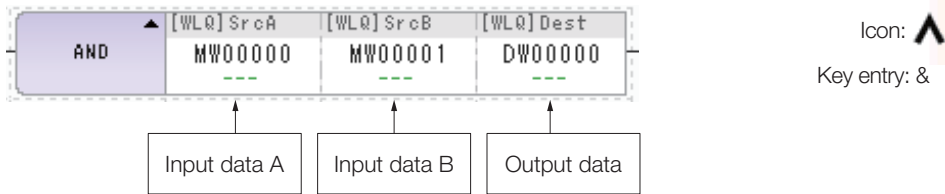


Each bit in the input data is evaluated as shown in the following truth table.

Input Data A	Input Data B	Output Data
0	0	0
0	1	0
1	0	0
1	1	1

Format

The format of this instruction is shown below.

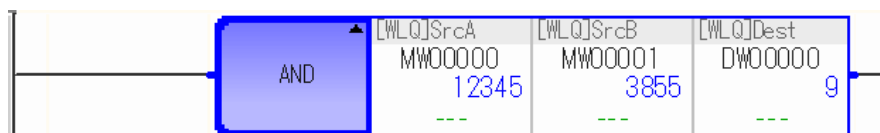
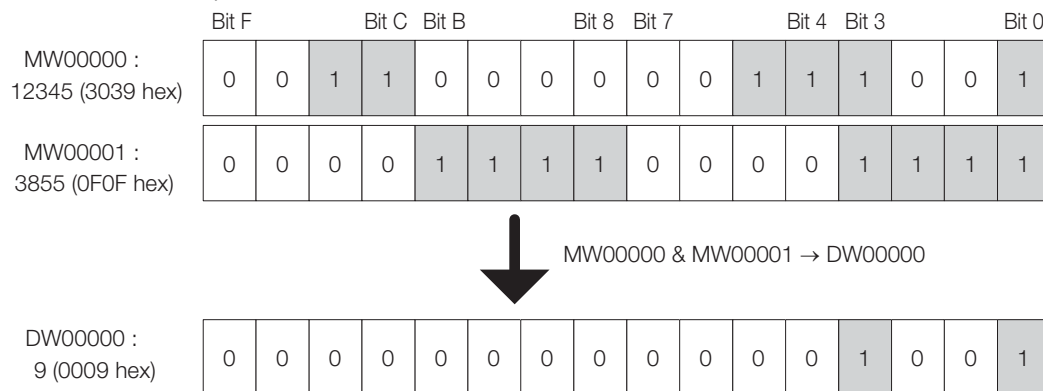


I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
SrcA (Input data A)	x	○	○	○	x	x	x	○	○
SrcB (Input data B)	x	○	○	○	x	x	x	○	○
Dest (Output data)	x	○*	○*	○*	x	x	x	○	x

* C and # registers cannot be used.

Programming Example

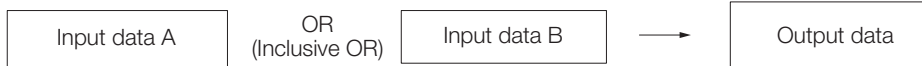
In the following programming example, a logical AND is performed on 12,345 (3039 hex) in input data A in MW00000 and 3,855 (0F0F hex) in input data B in MW00001, and the result is stored in the output data in DW00000.



4.4.2 Inclusive OR (OR)

A logical OR operation is performed on input data A and input data B and the result is stored in the output data.

This instruction can be used only with integer or double-length integer data.

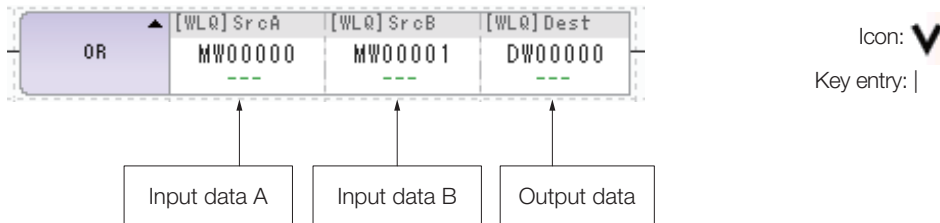


Each bit in the input data is evaluated as shown in the following truth table.

Input Data A	Input Data B	Output Data
0	0	0
0	1	1
1	0	1
1	1	1

Format

The format of this instruction is shown below.

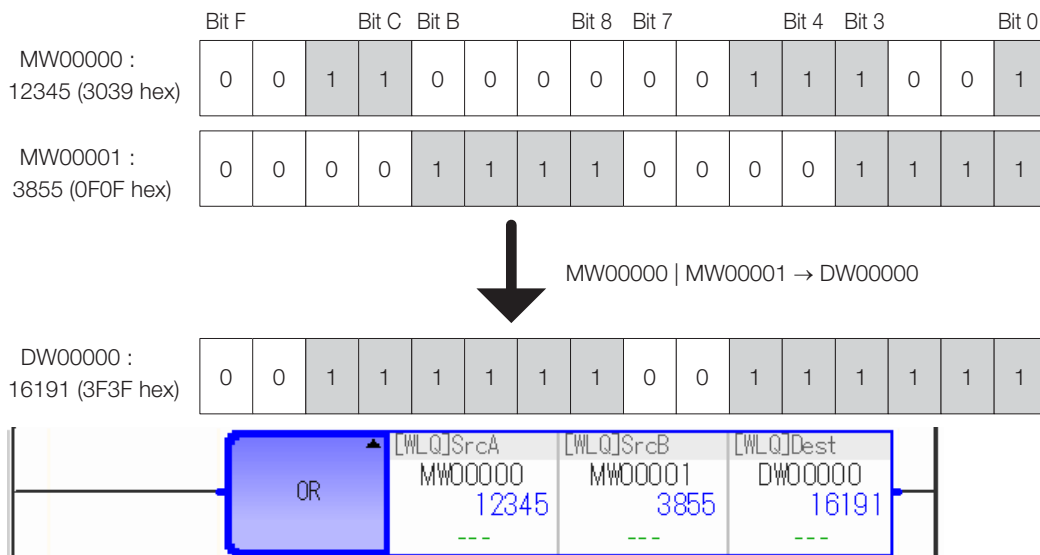


I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
SrcA (Input data A)	×	○	○	○	×	×	×	○	○
SrcB (Input data B)	×	○	○	○	×	×	×	○	○
Dest (Output data)	×	○*	○*	○*	×	×	×	○	×

* C and # registers cannot be used.

Programming Example

In the following programming example, a logical OR is performed on 12,345 (3039 hex) in input data A in MW00000 and 3,855 (0F0F hex) in input data B in MW00001, and the result is stored in the output data in DW00000.



4.4.3 Exclusive OR (XOR)

An exclusive logical OR operation is performed on input data A and input data B and the result is stored in the output data.

This instruction can be used only with integer or double-length integer data.

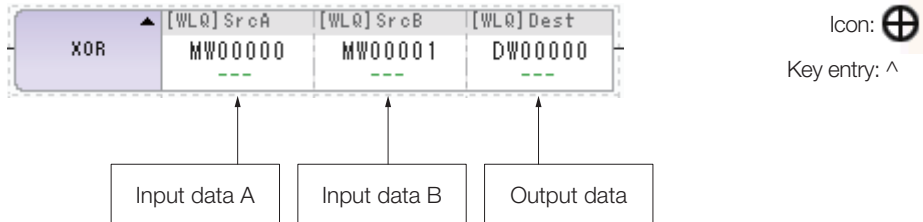


Each bit in the input data is evaluated as shown in the following truth table.

Input Data A	Input Data B	Output Data
0	0	0
0	1	1
1	0	1
1	1	0

Format

The format of this instruction is shown below.

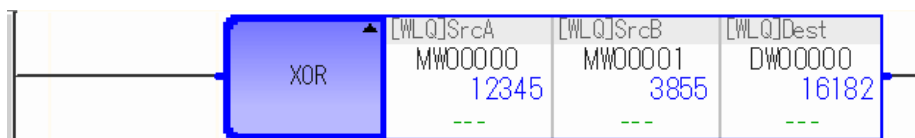
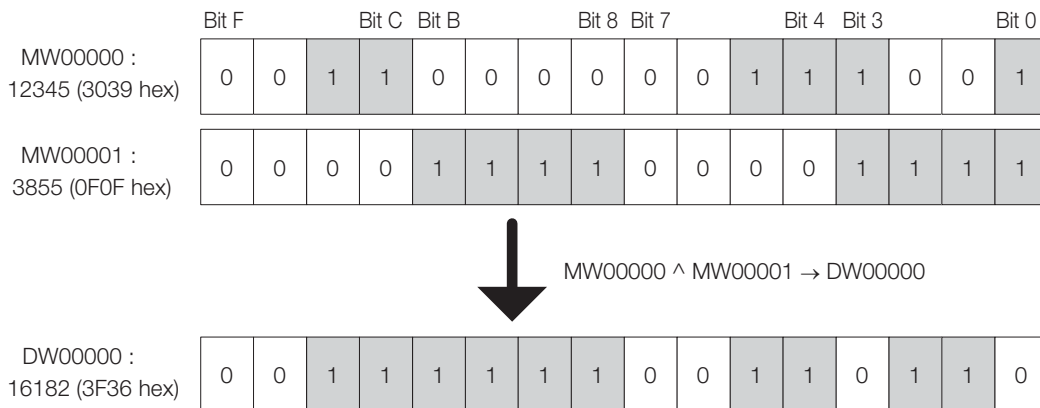


I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
SrcA (Input data A)	×	○	○	○	×	×	×	○	○
SrcB (Input data B)	×	○	○	○	×	×	×	○	○
Dest (Output data)	×	○*	○*	○*	×	×	×	○	×

* C and # registers cannot be used.

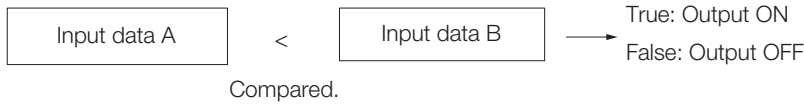
Programming Example

In the following programming example, an exclusive logical OR is performed on 12,345 (3039 hex) in input data A in MW00000 and 3,855 (0F0F hex) in input data B in MW00001, and the result is stored in the output data in DW00000.



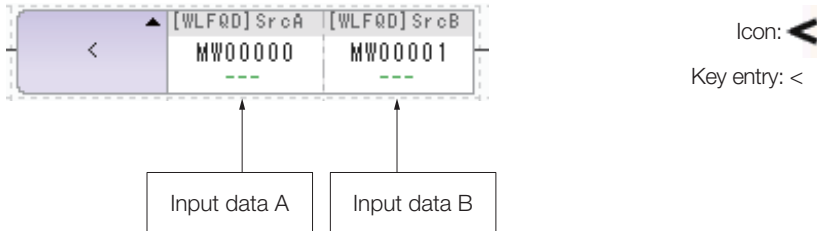
4.4.4 Less Than (<)

Input data A and input data B are compared and the result is stored in the bit output.



Format

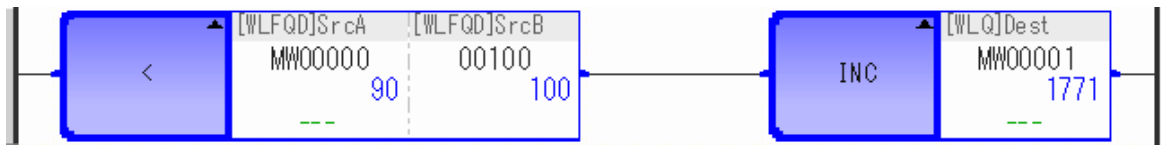
The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
SrcA (Input data A)	×	○	○	○	○	○	×	○	○
SrcB (Input data B)	×	○	○	○	○	○	×	○	○

Programming Example

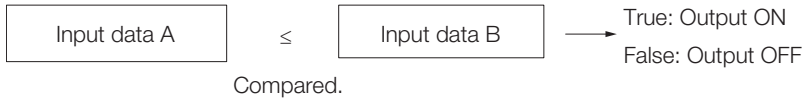
In the programming example shown below, the INC instruction on the right end of the line is executed because the comparison is true; that is, input data A is less than input data B when input data A in MW00000 is 90 and input data B is a constant set to 100.



Information With real number data, the value displayed by the MPE720 may not match the execution result of the comparison instruction due to a slight precision error.

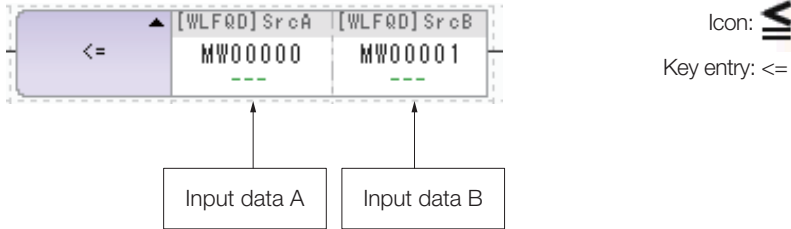
4.4.5 Less Than or Equal (\leq)

Input data A and input data B are compared and the result is stored in the bit output.



Format

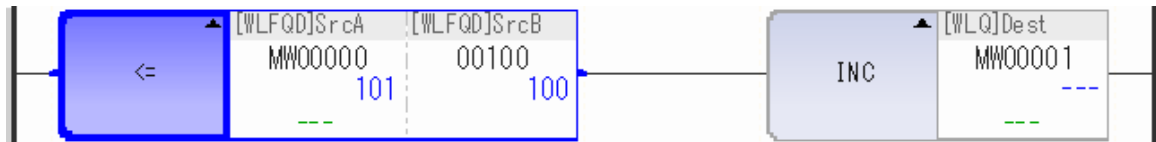
The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
SrcA (Input data A)	x	○	○	○	○	○	x	○	○
SrcB (Input data B)	x	○	○	○	○	○	x	○	○

Programming Example

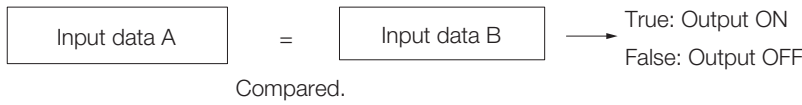
In the programming example shown below, the INC instruction on the right end of the line is not executed because the comparison is false; that is, input data A is greater than input data B when input data A in MW00000 is 101 and input data B is a constant set to 100.



Information With real number data, the value displayed by the MPE720 may not match the execution result of the comparison instruction due to a slight precision error.

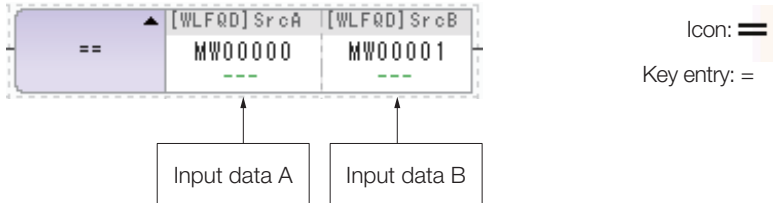
4.4.6 Equal (=)

Input data A and input data B are compared and the result is stored in the bit output.



Format

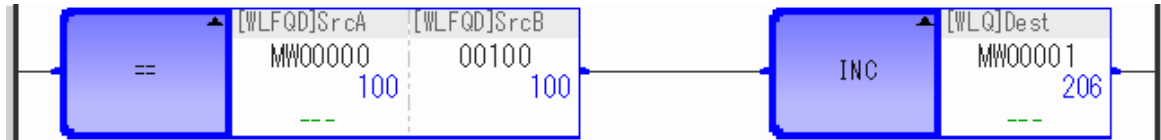
The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
SrcA (Input data A)	×	○	○	○	○	○	×	○	○
SrcB (Input data B)	×	○	○	○	○	○	×	○	○

Programming Example

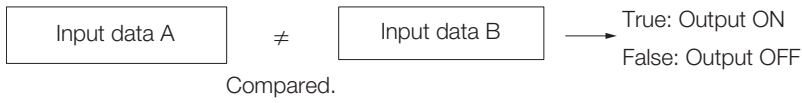
In the programming example shown below, the INC instruction on the right end of the line is executed because the comparison is true; that is, input data A is equal to input data B when input data A in MW00000 is 100 and input data B is a constant set to 100.



Information With real number data, the value displayed by the MPE720 may not match the execution result of the comparison instruction due to a slight precision error.

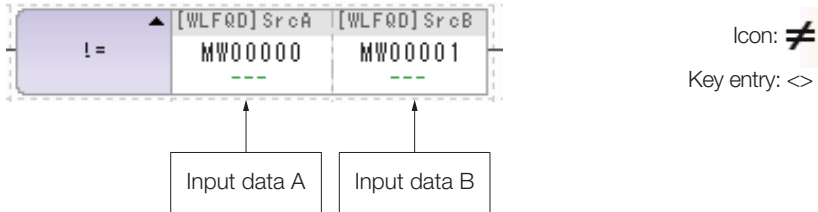
4.4.7 Not Equal (≠)

Input data A and input data B are compared and the result is stored in the bit output.



Format

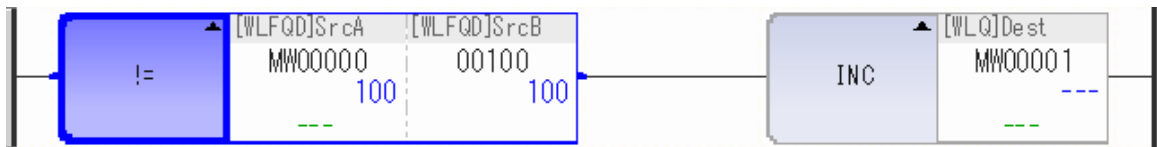
The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
SrcA (Input data A)	x	○	○	○	○	○	x	○	○
SrcB (Input data B)	x	○	○	○	○	○	x	○	○

Programming Example

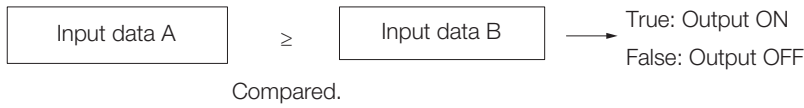
In the programming example shown below, the INC instruction on the right end of the line is not executed because the comparison is false and turns the output OFF; that is, input data A is equal to input data B when input data A in MW00000 is 100 and input data B is a constant set to 100.



Information With real number data, the value displayed by the MPE720 may not match the execution result of the comparison instruction due to a slight precision error.

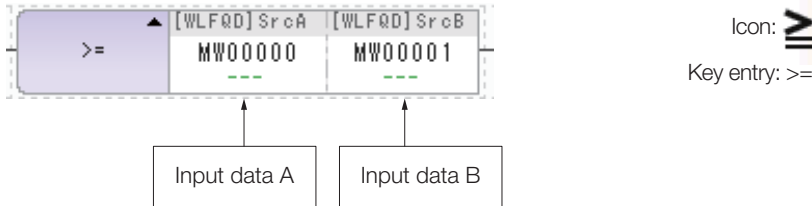
4.4.8 Greater Than or Equal (≥)

Input data A and input data B are compared and the result is stored in the bit output.



Format

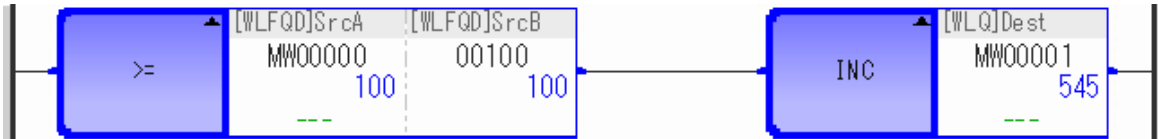
The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
SrcA (Input data A)	×	○	○	○	○	○	×	○	○
SrcB (Input data B)	×	○	○	○	○	○	×	○	○

Programming Example

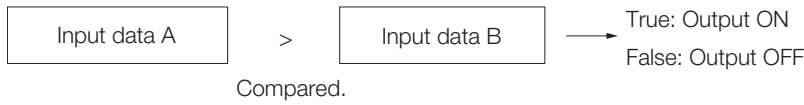
In the programming example shown below, the INC instruction on the right end of the line is executed because the comparison is true; that is, input data A is equal to or greater than input data B when input data A in MW00000 is 100 and input data B is a constant set to 100.



Information With real number data, the value displayed by the MPE720 may not match the execution result of the comparison instruction due to a slight precision error.

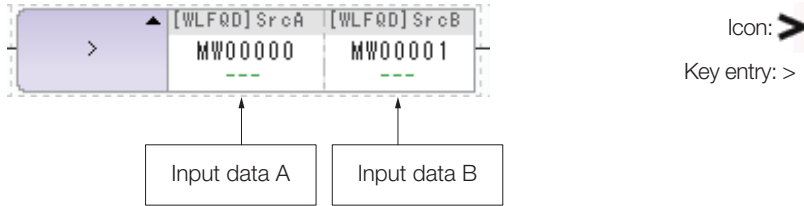
4.4.9 Greater Than (>)

Input data A and input data B are compared and the result is stored in the bit output.



Format

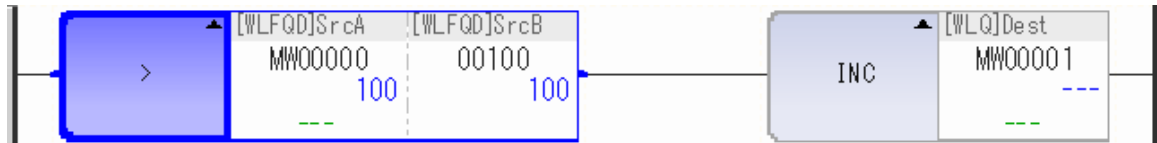
The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
SrcA (Input data A)	×	○	○	○	○	○	×	○	○
SrcB (Input data B)	×	○	○	○	○	○	×	○	○

Programming Example

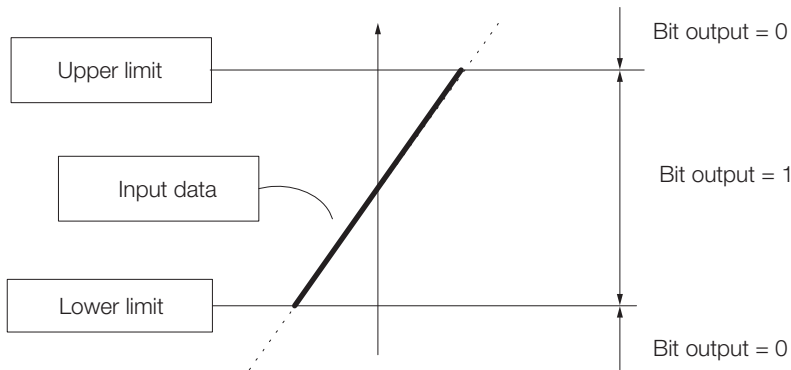
In the programming example shown below, the INC instruction on the right end of the line is not executed because the comparison is false; that is, input data A is not greater than input data B when input data A in MW00000 is 100 and input data B is a constant set to 100.



Information With real number data, the value displayed by the MPE720 may not match the execution result of the comparison instruction due to a slight precision error.

4.4.10 Range Check (RCHK)

A check is made to see if the input data is between upper limit and lower limit and the result is stored in the bit output.



• **Bit output = 1**

The bit output is set to 1 if the value of the input data is within the range that is greater than or equal to the lower limit, and less than or equal to the upper limit.

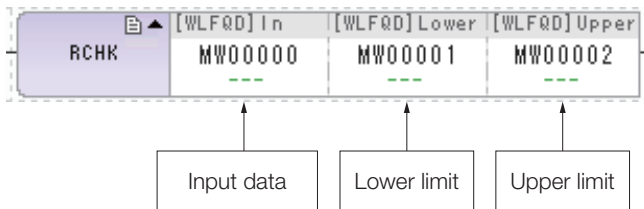
$$\boxed{\text{Lower limit}} \leq \boxed{\text{Input data}} \leq \boxed{\text{Upper limit}}$$

• **Bit output = 0**

The bit output is set to 0 if the value of the input data is outside the range that is greater than or equal to the lower limit, and less than or equal to the upper limit.

Format

The format of this instruction is shown below.



Icon: 
Key entry: RCHK

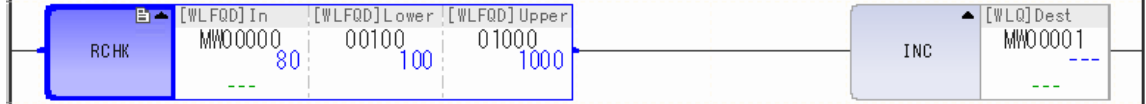
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In (Input data)	×	○	○	○	○	○	×	○	○
Lower (Lower limit)	×	○	○	○	○	○	×	○	○
Upper (Upper limit)	×	○	○	○	○	○	×	○	○

Information Always set the lower limit to a value that is less than or equal to the upper limit. If the lower limit is greater than the upper limit, the result will be invalid.

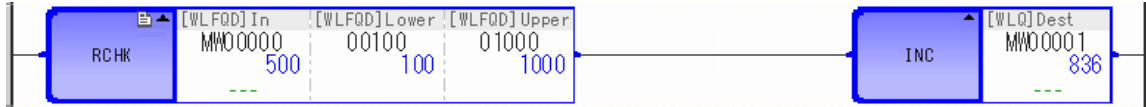
Programming Example

The following programming examples execute the RCHK instruction.

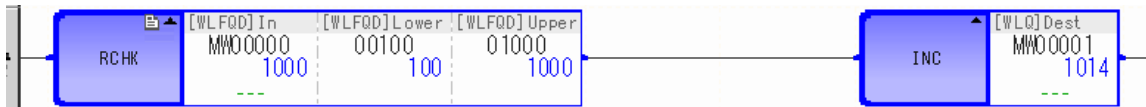
- When Input Data (MW00000) = 80, Lower Limit = 100, and Upper Limit = 1,000
The INC instruction on the right end of the line is not executed because the value of the input data is less than the lower limit and turns the bit output OFF.



- When Input Data (MW00000) = 500, Lower Limit = 100, and Upper Limit = 1,000
The INC instruction on the right end of the line is executed because the value of the input data is within the range that is greater than or equal to the lower limit and less than or equal to the upper limit, which sets the bit output to 1.



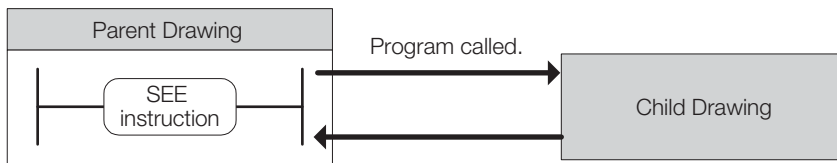
- When Input Data (MW00000) = 1,000, Lower Limit = 100, and Upper Limit = 1,000
The INC instruction on the right end of the line is executed because the value of the input data is within the range that is greater than or equal to the lower limit and less than or equal to the upper limit, which sets the bit output to 1.



4.5 Program Control Instructions

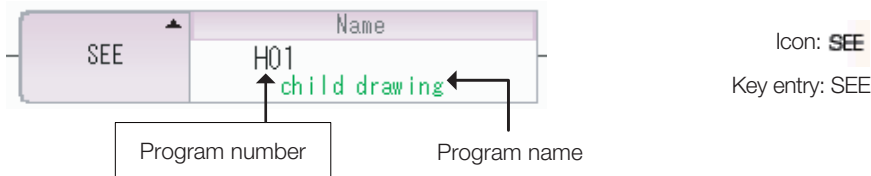
4.5.1 Call Sequence Program (SEE)

A child drawing is called from a parent drawing, or a grandchild drawing is called from a child drawing.



Format

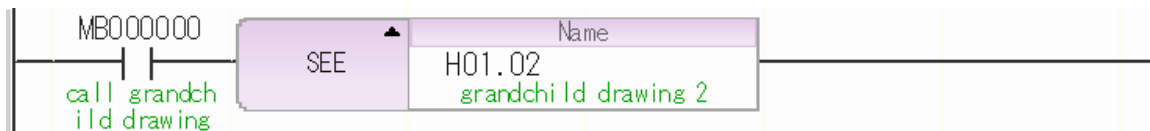
The format of this instruction is shown below.



I/O Item	Applicable Data Types
Name (Program number)	Registers may not be used. Specify the program number directly. The name of the specified program appears below the program number.

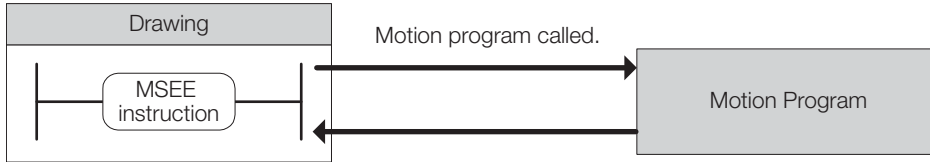
Programming Example

The SEE instruction calls drawing H01.02 when the MB000000 relay is ON. Thereafter, the process is executed and execution resumes from the next step after the SEE instruction. The SEE instruction does not call drawing H01.02 if the MB000000 relay is OFF.



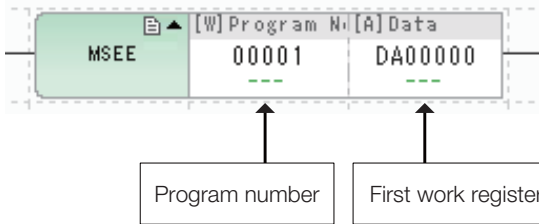
4.5.2 Call Motion Program (MSEE)

The specified motion program is called.
 Motion programs can be called only from H drawings.



Format

The format of this instruction is shown below.



Icon: **M See**
 Key entry: MSEE

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Program No. (Program number)	×	O*	×	×	×	×	×	×	O
Data (First work register)	×	×	×	×	×	×	O*	×	×

* M or D register only.

The following table shows the configuration of the work registers.

Address	Data Type	Name	Contents	I/O
0	W	Status Flags	Motion Program Status Flags	OUT
1	W	Control Signals	Motion Program Control Signals	IN
2	W	Interpolation Override	The override is used when executing interpolation instructions. Range: 0 to 32,767 Unit: 1 = 0.01%	IN
3	W	System Work Number	This is the system work number that calls the motion program.	IN

- Information** Specify the program number from 1 to 512.
 Refer to the following manual for details on motion programs.
 📖 MP3000 Series Motion Programming Manual (Manual No. SIEP C880725 14)

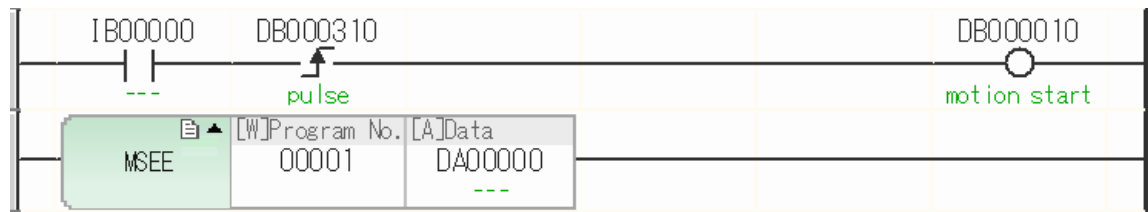
Programming Example

The following programming example shows how to execute the motion program MPM001 with program number 1.

When the IB00000 relay turns ON, the Request for Start of Program Operation (DB000010) in the control signals turns ON and executes the MPM001 motion program.

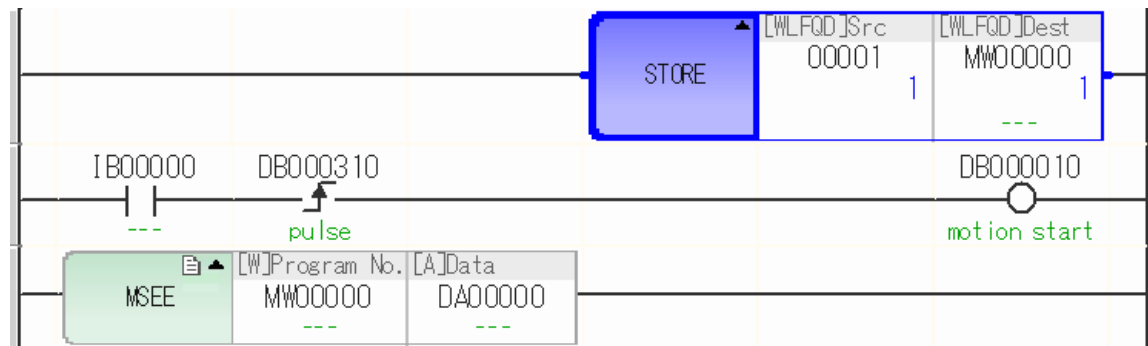
- **Direct Designation**

The program number is directly set to 1.



- **Indirect Designation**

The program number is set to MW0000.



Continue execution of the MSEE instruction until execution of the motion program is completed. When using indirect addressing, do not change the register value until the execution of the motion program is completed.

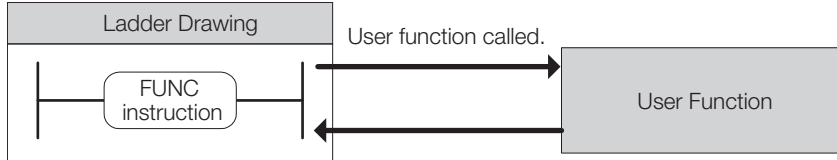
4.5.3 Call User Function (FUNC)

A user function is called. The user function must be defined before it can be called.

The Call User Function (FUNC) instruction can be nested to up to eight levels.

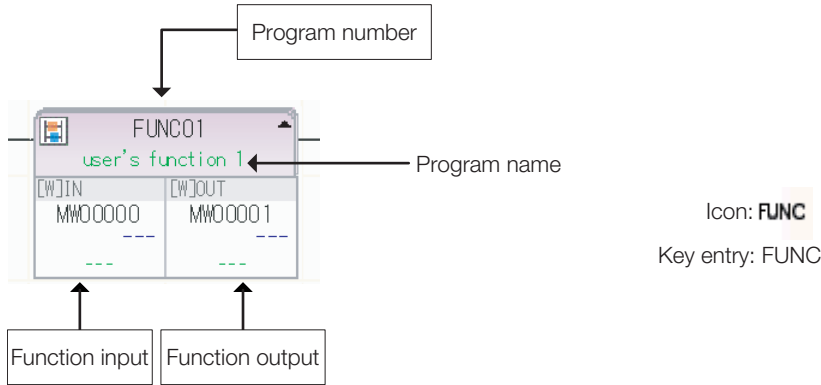
Refer to the following section for details on user functions.

1.3 Introduction – 1.3.3 User Functions on page 1-13



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types
Name (Program number)	Registers may not be used. Specify the program number directly. The name of the specified program appears above the instruction.
Function input	The register that is set in the function's input definition can be used.
Function output	The register that is set in the function's output definition can be used.

Programming Example

Refer to the following section for programming examples for user functions.

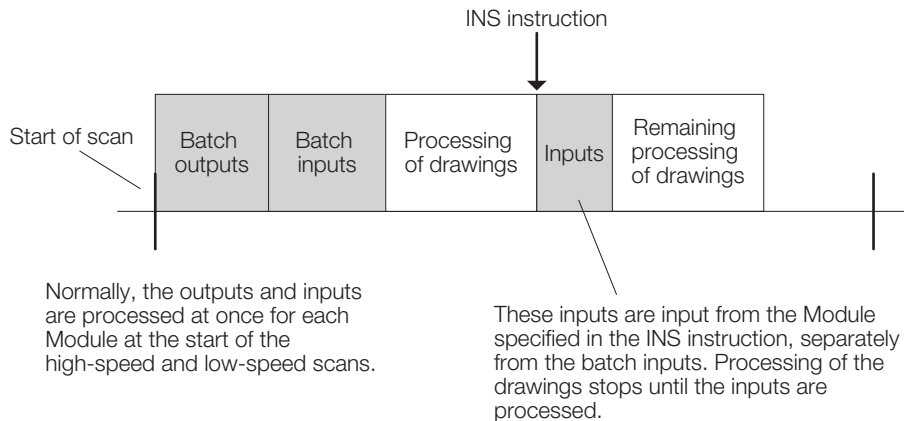
1.3 Introduction – 1.3.3 User Functions on page 1-13

4.5.4 Direct Input String (INS)

The INS instruction is executed in user programs to input data separately from the I/O processing that is performed by the system at the start of the high-speed and low-speed scans. When the INS instruction is executed, the inputs from the specified Module are processed according to the settings in the parameter table. The next instruction is not executed until input processing is completed.

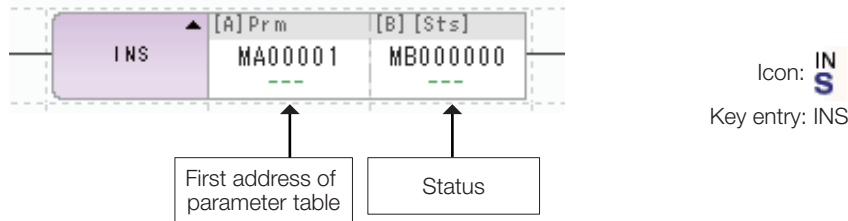
The following Modules can be specified.

- LIO-01/02 Module (LIO)
- LIO-04/05 Module (LIO32)
- LIO-06 Module (MIXIO)
- AI-01 Module (AI)



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Prm (First address of parameter table)	×	×	×	×	×	×	○*1	×	×
Sts (Status)*2	○*1	×	×	×	×	×	×	×	×

*1. C and # registers cannot be used.

*2. Optional.

The following figure shows the structure of the parameter table.

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RSSEL	Unit selection 1	Specify the Module to input from.	IN
1	W	MDSEL	Unit selection 2		IN
2	W	STS	Status	Each bit receives the input status for one word. 0: Normal 1: Error	OUT
3	W	N	Number of words	Specify the number of continuous words.	IN
4	W	ID1	Input data 1	Receives the data that was input. Contains 0 if an error occurs.	OUT
:	:	:	:		:
N + 3	W	IDN	Input data N		OUT

The following table gives details about the parameters in the Machine Controller.

Parameter	Module Name				
	LIO-01/02 (LIO)	LIO-04/05 (LIO32)	LIO-06 (MIXIO)	AI-01 (AI)	DI-01 (DI)
RSSEL	Specify the rack, unit, slot, and subslot of the target Module. Hexadecimal notation: zxuy hex x: Rack number from 1 to 7 u: Unit number from 0 to 4*1 y: Slot number from 0 to 9 z: Subslot number from 1 to maximum value (determined by Module specifications)				
MDSEL	0 (Not used.)	Offset: 0 or 1	Channel number - 1: 0 or 1	Channel number - 1: 0 to 7	Offset: 0 to 3
STS	Always 0.	Always 0.	Always 0.	*2	Always 0.
N	1	1 to 2 -MDSEL	1 to 2 -MDSEL	1 to 8 -MDSEL	1 to 4 -MDSEL

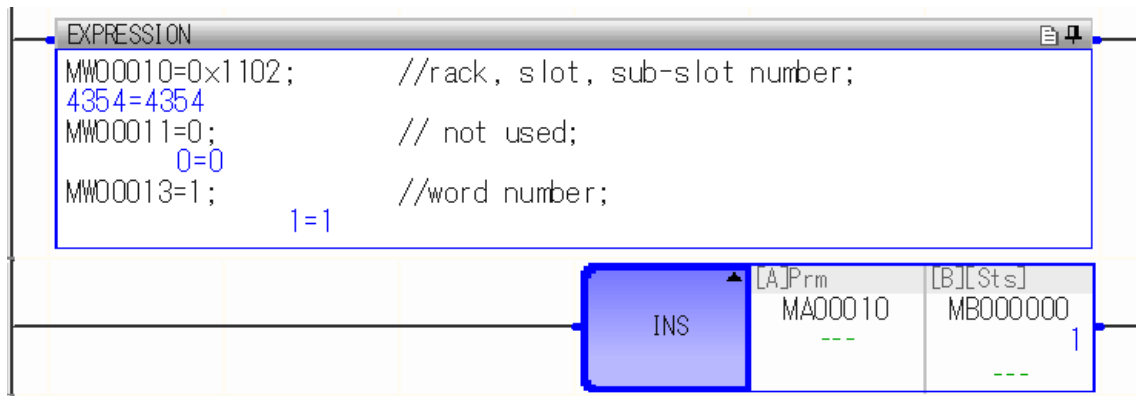
*1. A unit number setting of 0 specifies unit number 1.

*2. If a channel for which the allocation has been deleted in the AI Module detailed definition is specified for the INS instruction, the applicable channel number will be output for the bit. This is because it is not possible to read the data on channels for which allocations have been deleted. The relation between bits and channels is shown below.

- Bit 0: Channel 1
- Bit 1: Channel 2
- Bit 2: Channel 3
- Bit 3: Channel 4
- Bit 4: Channel 5
- Bit 5: Channel 6
- Bit 6: Channel 7
- Bit 7: Channel 8

Programming Example

When one word is input from the LIO at subslot number 1 on the LIO-01 Module mounted in rack 1, unit 1, and slot 2, the input data of the LIO is stored in the MW00014 status word.

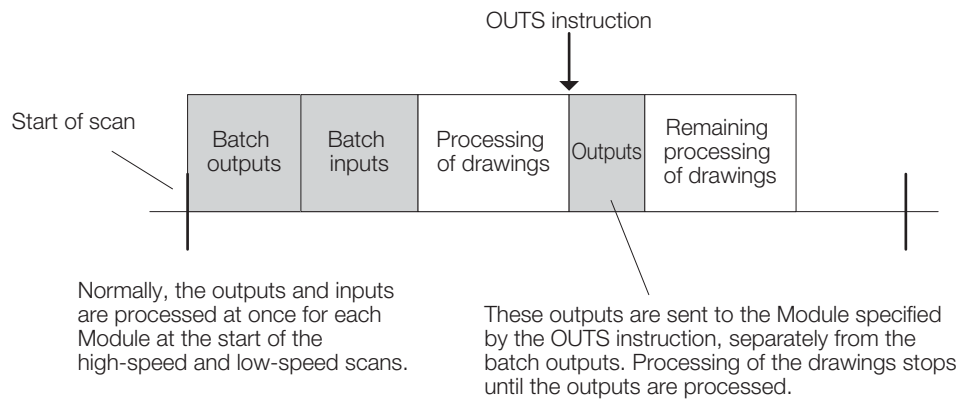


4.5.5 Direct Output String (OUTS)

The OUTS instruction is executed in user programs to output data separately from the I/O processing that is performed by the system at the start of the high-speed and low-speed scans. When the OUTS instruction is executed, the outputs to the specified Module are processed according to the settings in the parameter table.

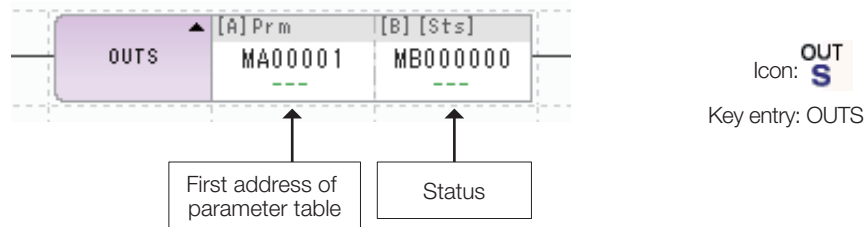
The following Modules can be specified.

- LIO-01/02 Module (LIO)
- LIO-04/05 Module (LIO32)
- LIO-06 Module (MIXIO)
- DO-01 Module (DO)
- AO-01 Module (AO)



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Prm (First address of parameter table)	×	×	×	×	×	×	O*1	×	×
Sts (Status)*2	O*1	×	×	×	×	×	×	×	×

*1. C and # registers cannot be used.

*2. Optional.

The following figure shows the structure of the parameter table.

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RSSEL	Unit selection 1	Specify the Module to output to.	IN
1	W	MDSEL	Unit selection 2		IN
2	W	STS	Status	Each bit receives the input status for one word. 0: Normal 1: Error	OUT
3	W	N	Number of words	Specify the number of output words (always 1).	IN
4	W	OD1	Output data 1	Specify the data to output.	OUT
:	:	:	:		:
N + 3	W	ODN	Output data N		OUT

The following table gives details about the parameters in the Machine Controller.

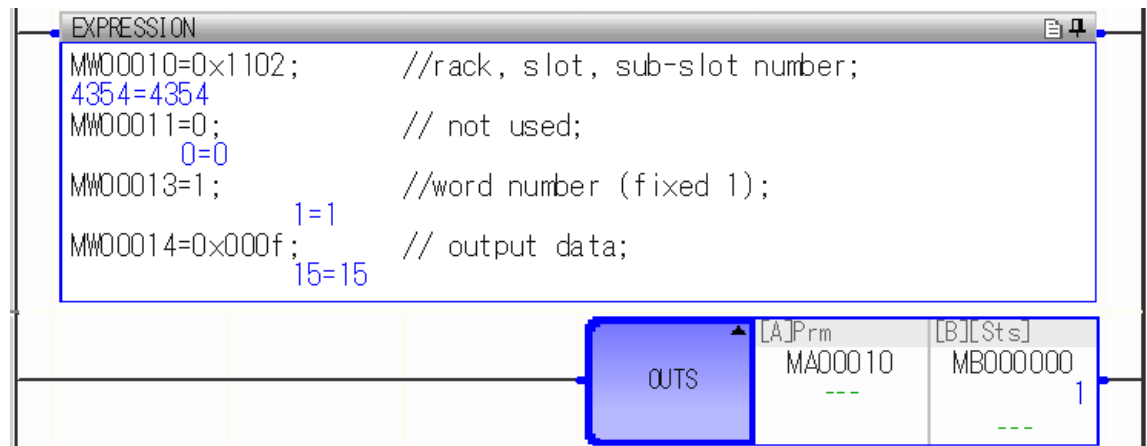
Parameter	Module Name				
	LIO-01/02 (LIO)	LIO-04/05 (LIO32)	LIO-06 (MIXIO)	DO-01 (DO)	AO-01 (AO)
RSSEL	Specify the rack, unit, slot, and subslot of the target Module. Hexadecimal notation: zxuy hex x: Rack number from 1 to 7 u: Unit number from 0 to 4*1 y: Slot number from 0 to 9 z: Subslot number from 1 to maximum value (determined by Module specifications)				
MDSEL	0 (Not used.)	Offset: 0 or 1	Offset: 0 or 1	Offset: 0 to 3	Channel number - 1: 0 to 3
STS	Always 0.	Always 0.	Always 0.	Always 0.	*2
N	1	1 to 2 - MDSEL	1 to 2 - MDSEL	1 to 4 - MDSEL	1 to 4 - MDSEL

*1. A unit number setting of 0 specifies unit number 1.

*2. If a channel for which the allocation has been deleted in the AO Module detailed definition is specified for the OUTS instruction, the applicable channel number will be output for the bit. This is because it is not possible to read the data on channels for which allocations have been deleted. The relation between bits and channels is shown below.
Bit 0: Channel 1
Bit 1: Channel 2
Bit 2: Channel 3
Bit 3: Channel 4

Programming Example

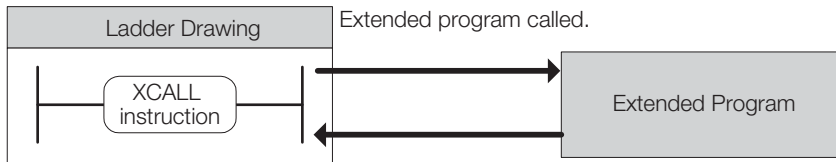
When one word is output to the LIO at subslot number 1 on the LIO-01 Module mounted in rack 1, unit 1, and slot 2, the data in the MW00014 status word is output to LIO.



4.5.6 Call Extended Program (XCALL)

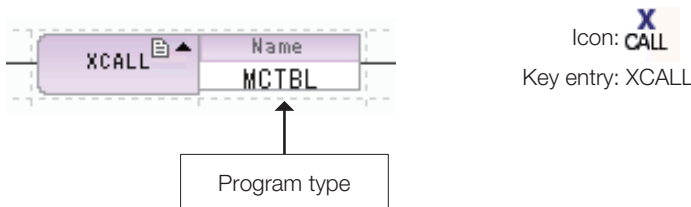
An extended program, such as a table program that contains a table of constants, is executed. The MPE720 converts an extended program into a ladder program. Converted ladder programs can be executed with the XCALL instruction.

Although more than one XCALL instruction can be used in a single drawing, the same extended program cannot be called more than once.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types
Name (Program type)	Registers may not be used. Specify the following type. MCTBL: Constants table

Programming Example

This example shows how to call a MCTBL constants table.

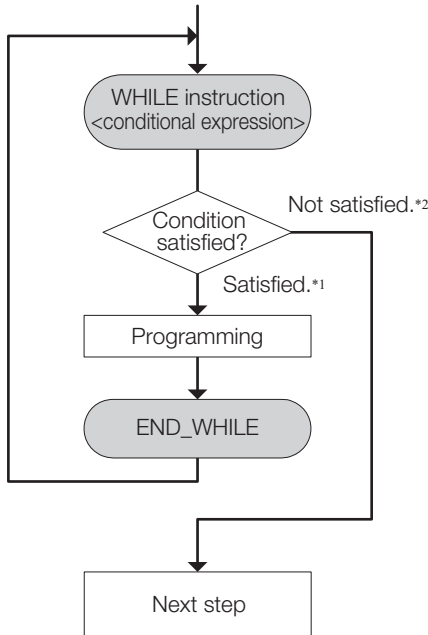


Information I/O conversion tables, interlock tables, and part assembly tables cannot be used with the Machine Controller.

4.5.7 WHILE Construct (WHILE, END_WHILE)

The programming between the WHILE and END_WHILE instructions is executed when the conditional expression for the WHILE instruction is satisfied. After the last line is executed, program execution returns to the WHILE instruction. Execution of the programming is repeated for as long as the conditional expression is satisfied.

If the conditional expression is not satisfied, program execution jumps to the next step following the END_WHILE instruction. None of the programming between the WHILE and END_WHILE instructions is executed.

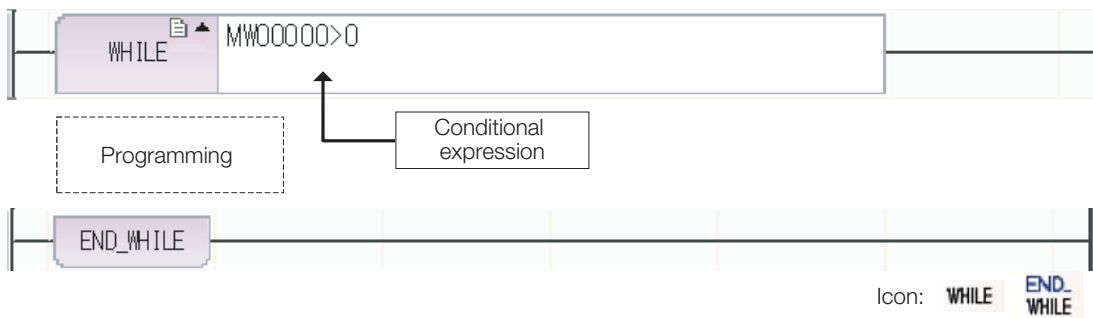


*1. The programming is executed and then execution returns to the WHILE instruction.

*2. The programming is not executed and execution jumps to the next step.

Format

The format of this instruction is shown below.



Key entry: WHILE and WEND

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Conditional expression	O*	O*	O*	O*	O*	O*	×	O*	O*

* Write with the format for an EXPRESSION instruction. Refer to the following appendix for details on the format used to write the expression.

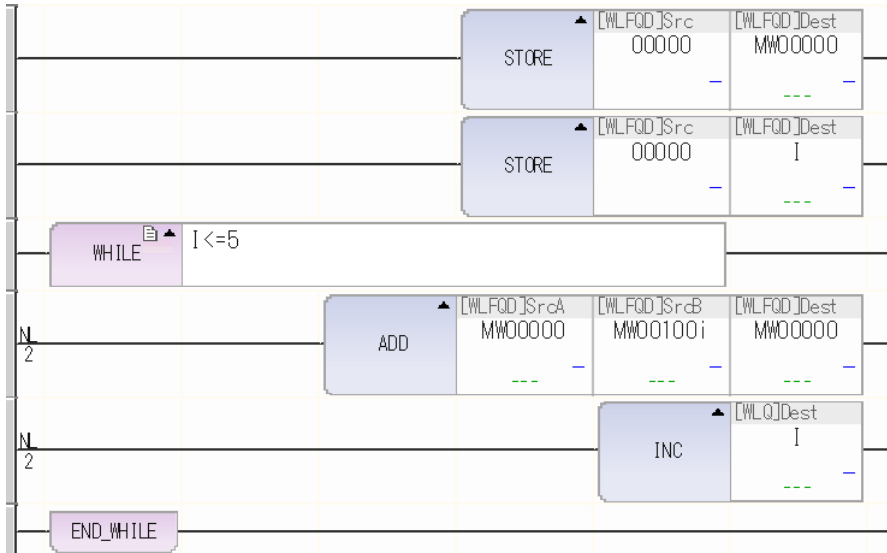
Appendix C Format for EXPRESSION Instructions

Programming Example

In the following programming example, the registers from MW00100 to MW00105 are added together and stored in the MW00000 register.

The conditional expression is $I \leq 5$, so the ADD (+) instruction is executed while I is 0 to 5.

The conditional expression is no longer satisfied when I is 6, so program execution jumps to the next step following the END_WHILE instruction.



Important

Execution of the programming is repeated for as long as the conditional expression for the WHILE instruction is satisfied. If the conditional expression never becomes unsatisfied, or if it takes too much time to become unsatisfied, the Machine Controller system will shut down. In the example given above, an endless loop would occur if the programming did not include the instruction that increments I.

Additional Information

◆ Applicable Conditional Expressions

The conditional expression for a WHILE instruction must be written with the format for an EXPRESSION instruction to produce a Boolean (TRUE or FALSE) result. Numerical expressions that include substitution operators will not be recognized.

Expression Example	Notation	Remarks
MB000001 == true	OK	True: ON
MB000001 != false	OK	False: OFF
MW00002 < 100	OK	
MF00002 < sin(60.0)	OK	
MW00001 == 0x00FF OK	OK	Prefix hexadecimal numbers with 0x.
MB000001 = true	NG	
MW00001 = MW00002	NG	

Note: Refer to the following appendix for details on applicable instructions, operation order, and notation conventions.

Appendix C Format for EXPRESSION Instructions

◆ Nesting Depth

The FOR, WHILE, and IF constructs can contain other constructs. This is called nesting. The maximum depth of a nested structure that uses FOR, WHILE, and IF statements is limited to 8 levels.

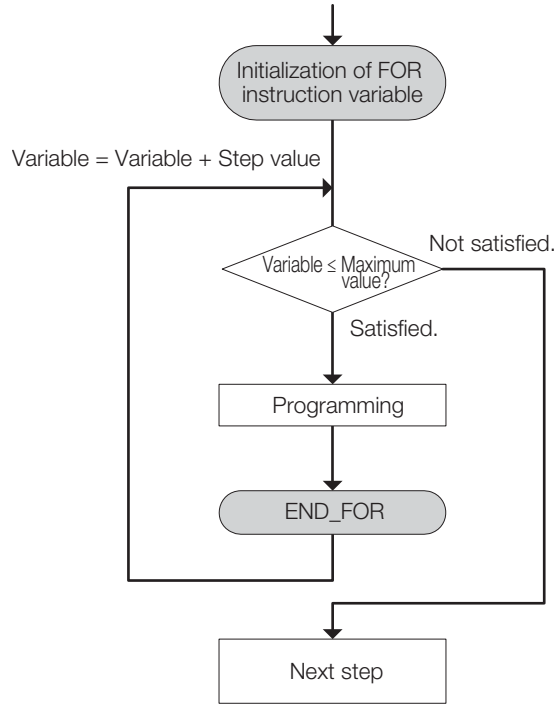
If an instruction is preceded by a contact, it is treated like an IF construct and is included in the number of nesting levels.

4.5.8 FOR Construct (FOR, END_FOR)

The programming between the FOR and END_FOR instructions is repeatedly executed.

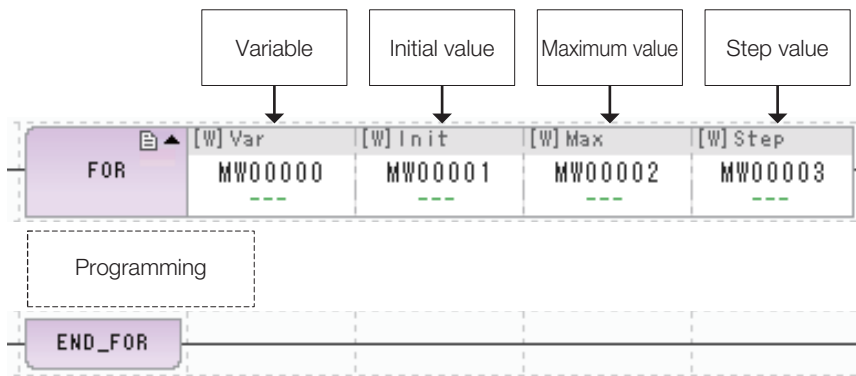
The initial value starts with the value in a register specified as the variable. This variable is incremented by the step value each time execution is repeated.

The conditional expression for the FOR instruction is no longer satisfied when the value of the variable exceeds the maximum value, so program execution jumps to the next step.



Format

The format of this instruction is shown below.



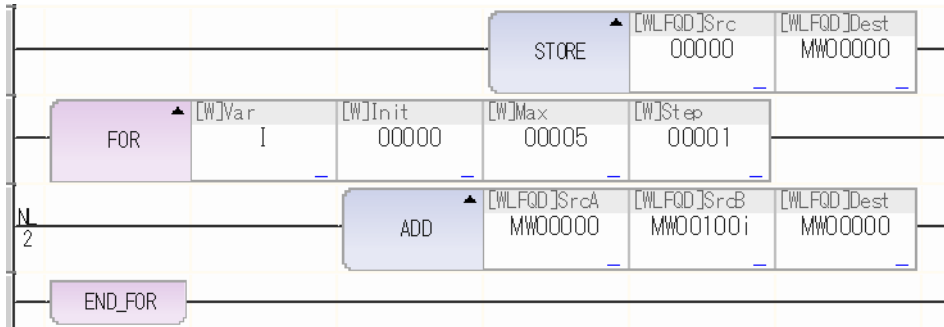
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Var (Variable)	x	O*	x	x	x	x	x	O	x
Init (Initial value)	x	O	x	x	x	x	x	O	O
Max (Maximum value)	x	O	x	x	x	x	x	O	O
Step (Step value)	x	O	x	x	x	x	x	O	O

* C and # registers cannot be used.

Programming Example

In the following programming example, the registers from MW00100 to MW00105 are added together and stored in the MW00000 register.

In this example, variable *I* is initialized to 0 by storing 0. Thereafter, the ADD (+) instruction is executed until variable *I* exceeds the maximum value of 5. The conditional expression is no longer satisfied when *I* is 6, so program execution jumps to the next step following the END_FOR instruction.



Additional Information

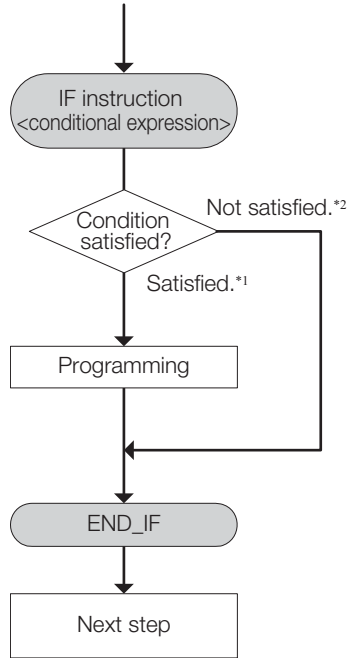
The FOR, WHILE, and IF constructs can contain other constructs. This is called nesting. The maximum depth of a nested structure that uses FOR, WHILE, and IF statements is limited to 8 levels.

If an instruction is preceded by a contact, it is treated like an IF construct and is included in the number of nesting levels.

4.5.9 IF Construct (IF, END_IF)

Execution of the programming between the IF and END_IF instructions is repeated for as long as the conditional expression for the IF instruction is satisfied.

The programming is not executed if the conditional expression is not satisfied.

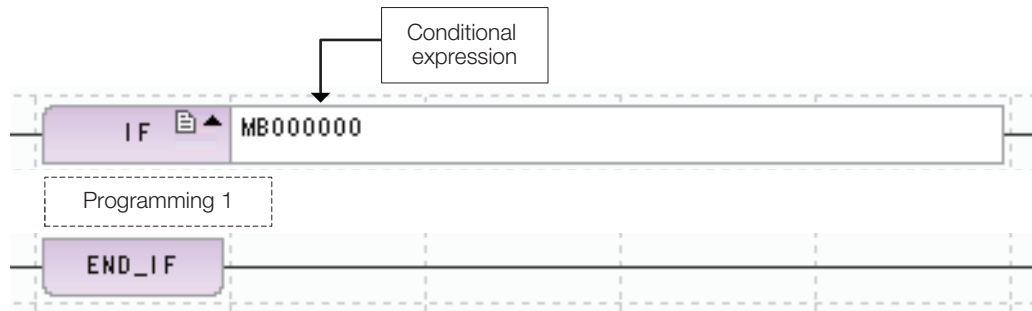


*1. The programming is executed and execution jumps to the next step.

*2. The programming is not executed and execution jumps to the next step.

Format

The format of this instruction is shown below.



Icon: IF , END_IF

Key entry: IF and IEND

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Conditional expression	○*	○*	○*	○*	○*	○*	×	○*	○*

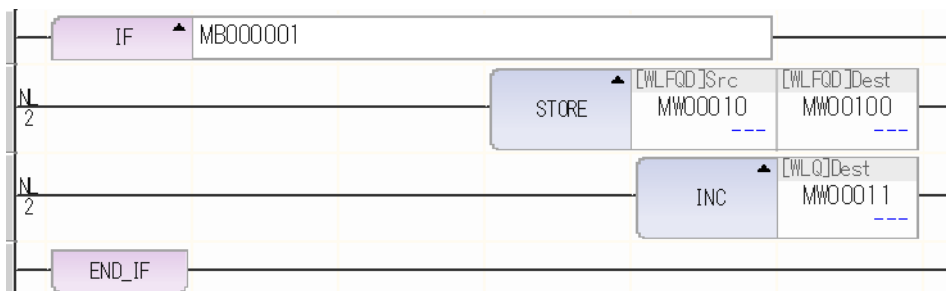
* Write with the format for an EXPRESSION instruction.

Refer to the following appendix for details on the format used to write the expression.

Appendix C Format for EXPRESSION Instructions

Programming Example

When the conditional expression (MB000001) for the IF instruction turns ON, the value of MW00010 is set in MW01000 and MW00011 is incremented.




Additional Information

◆ Applicable Conditional Expressions

The conditional expression for an IF instruction must be written with the format for an EXPRESSION instruction to produce a Boolean (TRUE or FALSE) result. Numerical expressions that include substitution operators will not be recognized.

Expression Example	Notation	Remarks
MB000001 == true	OK	True: ON
MB000001 != false	OK	False: OFF
MW00002 < 100	OK	
MF00002 < sin(60.0)	OK	
MW00001 == 0x00FF OK	OK	Prefix hexadecimal values with 0x.
MB000001 = true	NG	
MW00001 = MW00002	NG	

Note: Refer to the following appendix for details on applicable instructions, operation order, and notation conventions.

 Appendix C Format for EXPRESSION Instructions

◆ Nesting Depth

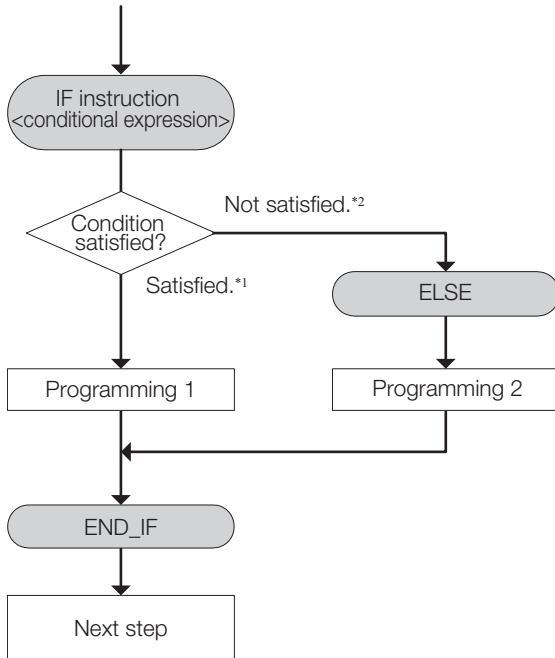
The FOR, WHILE, and IF constructs can contain other constructs. This is called nesting. The maximum depth of a nested structure that uses FOR, WHILE, and IF statements is limited to 8 levels.

If an instruction is preceded by a contact, it is treated like an IF construct and is included in the number of nesting levels.

4.5.10 IF-ELSE Construct (IF, ELSE, END_IF)

When the conditional expression for the IF instruction is satisfied, only programming 1 is executed. Programming 2 is not executed.

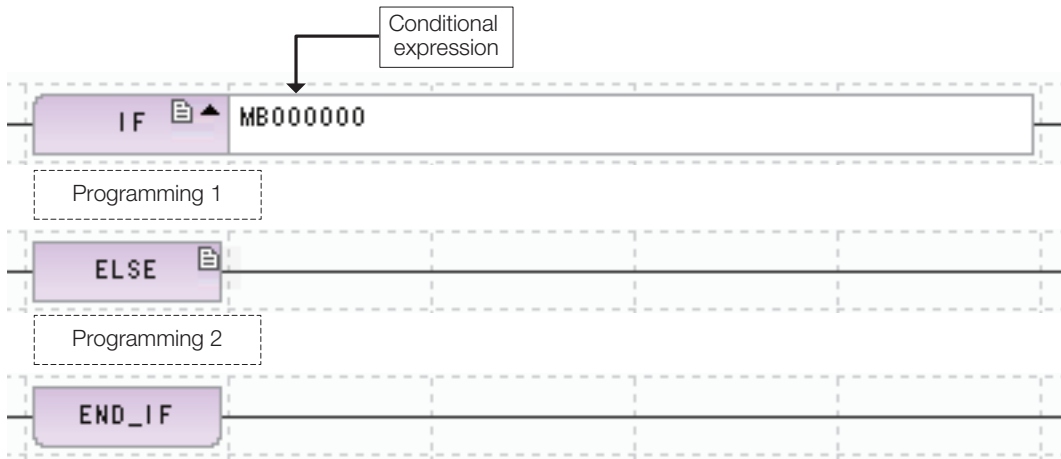
If the conditional expression is not satisfied, only programming 2 is executed. Programming 1 is not executed.



*1. Programming 1 is executed and execution jumps to the next step.
 *2. Programming 2 is executed and execution jumps to the next step.

Format

The format of this instruction is shown below.



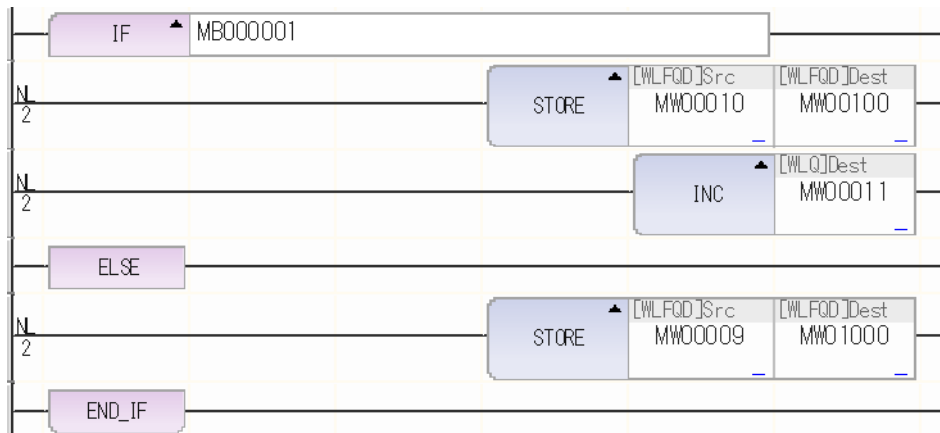
Icon: IF, ELSE, and END_IF
 Key entry: IF, ELSE, and IEND

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Conditional expression	○*	○*	○*	○*	○*	○*	×	○*	○*

* Write with the format for an EXPRESSION instruction. Refer to the following appendix for details on the format used to write the expression.
 Appendix C Format for EXPRESSION Instructions

Programming Example

When the conditional expression (MB000001) for the IF instruction turns ON, the value of MW00010 is set in MW01000 and MW00011 is incremented. When the conditional expression (MB000001) for the IF instruction turns OFF, the value of MW00009 is set in MW01000.



Additional Information

The conditional expressions that can be used, and the nesting depth is the same as for IF constructs.

4.5.11 Expression (EXPRESSION)

An expression may contain the following elements:

- A variable name or structure can be used in place of a register, similar to C language.
- Basic functions such as the SIN and COS functions.
- Arithmetic operators, logical operators, comparison operators, and substitution operators
- Arrays

EXPRESSION Instruction
<pre> MW00000 = 10; MW00001 = DATA1; ML00002 = MW00000 + 100; MF00004 = sin(MF00006); MW00006 = 0x3FFF; : </pre>

Format

The format of this instruction is shown below.

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Operation expression	O*	O*	O*	O*	O*	O*	×	O*	O*

* Write with the format for an EXPRESSION instruction.
 Refer to the following appendix for details on the format used to write the expression.
 Appendix C Format for EXPRESSION Instructions

Programming Example

In the following programming example, multiple operations are programmed in a single EXPRESSION instruction.

Additional Information

The EXPRESSION instruction can be programmed with numeric expressions in addition to expressions that return Boolean TRUE or FALSE values.

Expression Example	Notation	Remarks
MB000000 = true;	OK	True: ON
MW00000 = MW00001+10	OK	-
MW00000 = 0x00FF;	OK	Prefix hexadecimal values with 0x.
MB000000 == true;	NG	-
MW00001 > MW00000;	NG	-

Note: Refer to the following appendix for details on applicable instructions, operation order, and notation conventions.
 Appendix C Format for EXPRESSION Instructions

4.6 Basic Function Instructions

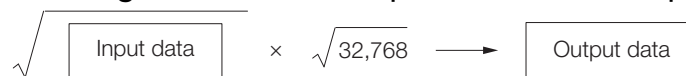
4.6.1 Square Root (SQRT)

The square root of the integer or real number input data is calculated and the result is stored in the output data.

Double-length integers cannot be used.

Information If the input data is less than 0, the absolute value of the input data will be used to perform the operation and output the result.

◆ Integer SQRT: The Input Data and Output Data Are Integer Data.

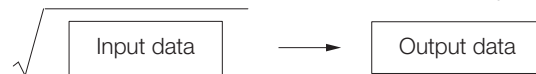


Information With integer SQRT instructions, the result is calculated using the following formula, unlike the square root used in mathematics.

$$\text{sign}(\text{input data}) \times \sqrt{|\text{input data}| \times 32,768}$$

This is the same as multiplying the result of the mathematical square root by $\sqrt{32,768}$. If the input is a negative number, the square root of the absolute value is calculated, and the negative number is given as the operation result. The maximum operation error is ± 2 .

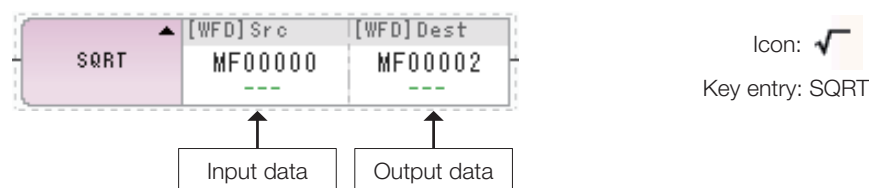
◆ Real Number SQRT: For Any Other Data Types



Information The SQRT instruction uses the immediately preceding operation result (real number data) as the input and returns the square root as real number data.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	×	○	×	×	○	○	×	○	○
Dest (Output data)	×	○*	×	×	○*	○*	×	○	×

* C and # registers cannot be used.

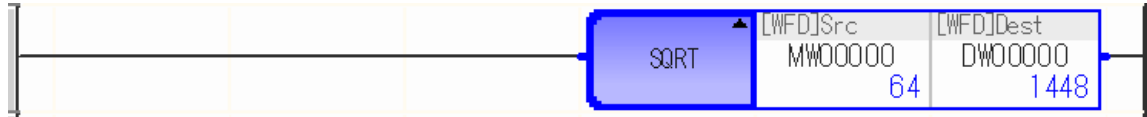
Programming Example

The following programming examples demonstrate the SQRT instruction using integer and real number input data.

• **Integer SQRT**

The square root of 64, an integer in the input data in MW00000, is multiplied by $128\sqrt{2}$ and the result is stored in the output data in DW00000.

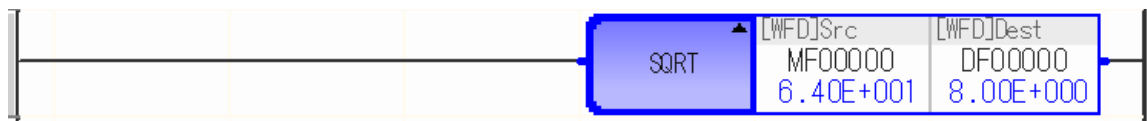
$$\sqrt{64} \times 128\sqrt{2} \rightarrow DW00000 = 1448$$



• **Real Number SQRT**

The square root of 64.0, a real number in the input data in MF00000, is calculated and the result is stored in the output data in DF00000.

$$\sqrt{64.0} \rightarrow DF00000 = 8.0$$



4.6.2 Sine (SIN)

The sine of the integer or real number input data is calculated and the result is stored in the output data.

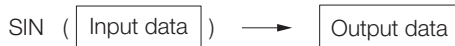
Double-length integers cannot be used.

◆ **Integer Input Data and Output Data**



- Note: 1. The input data is in degrees, where 1 = 0.01 degree.
- 2. The operation result is multiplied by 10,000 and stored in the output data.

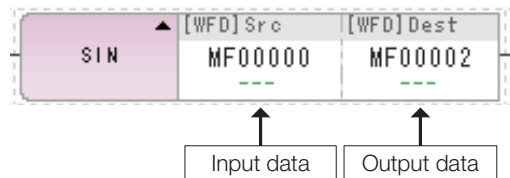
◆ **Real Number Input Data and Output Data**



Note: The input data is in degrees.

Format

The format of this instruction is shown below.



Icon: **sin**
Key entry: SIN

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	×	○	×	×	○	○	×	○	○
Dest (Output data)	×	○*	×	×	○*	○*	×	○	×

* C and # registers cannot be used.

◆ Integer

The input data is in degrees, where 1 = 0.01 degree.

Therefore, the SIN function instruction can operate on values between -327.78 and 327.67 degrees.

The output of the SIN function is multiplied by 10,000, so the data will be output between -10,000 and 10,000.

◆ Real Number

The input data is in degrees.

Programming Example

The following programming examples demonstrate the SIN instruction using integer and real number input data.

• Integer SIN

The sine of 9,000, an integer in the input data in MW00000, is calculated and the result is stored in the output data in DW00000.

$\text{SIN}(90.00 \text{ deg}) \times 10,000 \rightarrow \text{DW00000} = 10,000$



• Real Number SIN

The sine of 90.0, a real number in the input data in MF00000, is calculated and the result is stored in the output data in DF00000.

$\text{SIN}(90.0 \text{ deg}) \rightarrow \text{DF00000} = 1.0$



4.6.3 Cosine (COS)

The cosine of the integer or real number input data is calculated and the result is stored in the output data.

Double-length integers cannot be used.

◆ Integer Input Data and Output Data

$\text{COS}(\text{Input data}) \times 10,000 \rightarrow \text{Output data}$

- Note:
1. The input data is in degrees, where 1 = 0.01 degree.
 2. The operation result is multiplied by 10,000 and stored in the output data.
 3. The input data must be between -327.68 and 32.767 degrees. Any other number will not produce the correct result.

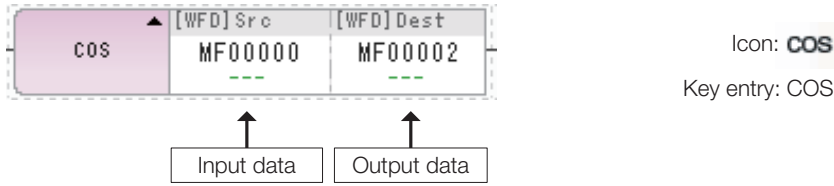
◆ Real Number Input Data and Output Data

$\text{COS}(\text{Input data}) \rightarrow \text{Output data}$

Note: The input data is in degrees.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	×	○	×	×	○	○	×	○	○
Dest (Output data)	×	○*	×	×	○*	○*	×	○	×

* C and # registers cannot be used.

◆ Integer

The input data is in degrees, where 1 = 0.01 degree.

Therefore, the COS function instruction can operate on values between -327.78 and 327.67 degrees.

The output of the COS function is multiplied by 10,000, so the data will be output between -10,000 and 10,000.

◆ Real Number

The input data is in degrees.

Programming Example

The following programming examples demonstrate the COS instruction using integer and real number input data.

• Integer COS

The cosine of 18,000, an integer in the input data in MW00000, is calculated and the result is stored in the output data in DW00000.

$$\text{COS (180.00 deg)} \times 10,000 \rightarrow \text{DW00000} = -10,000$$



• Real Number COS

The cosine of 180.0, a real number in the input data in MF00000, is calculated and the result is stored in the output data in DF00000.

$$\text{COS (180.0 deg)} \rightarrow \text{DF00000} = -1.0$$



4.6.4 Tangent (TAN.)

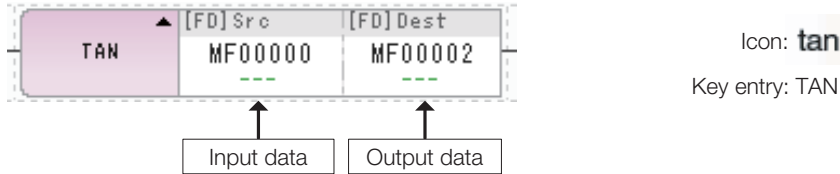
The tangent of the real number input data is calculated and the result is stored in the output data.

TAN () →

Note: The input data is in degrees.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	×	×	×	×	○	○	×	×	○
Dest (Output data)	×	×	×	×	○*	○*	×	×	×

* C and # registers cannot be used.

Programming Example

In the following programming example, the tangent of 45 in input data in MW00000 is calculated and the result is stored in the output data in DF00000.

TAN (45.0 deg) → DF00000 = 1.0



4.6.5 Arc Sine (ASIN)

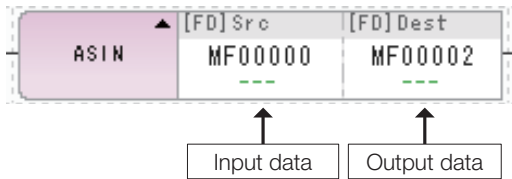
The arc sine of the real number input data is calculated and the result is stored in the output data.

$$\text{SIN} \left(\boxed{\text{Input data}} \right)^{-1} \longrightarrow \boxed{\text{Output data}}$$

Note: The output data is in degrees.

Format

The format of this instruction is shown below.



Icon: 

Key entry: ASIN

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	×	×	×	×	○	○	×	×	○
Dest (Output data)	×	×	×	×	○*	○*	×	×	×

* C and # registers cannot be used.

Set the input data to a value between -1.0 and 1.0. The output is set to 0 if the input value is out of range.

Programming Example

In the following programming example, the arc sine of 1.0 in input data in MF00000 is calculated and the result is stored in the output data in DF00000.

$$\text{SIN} (1.0)^{-1} \rightarrow \text{DF00000} = 90.0 \text{ (degrees)}$$



4.6.6 Arc Cosine (ACOS)

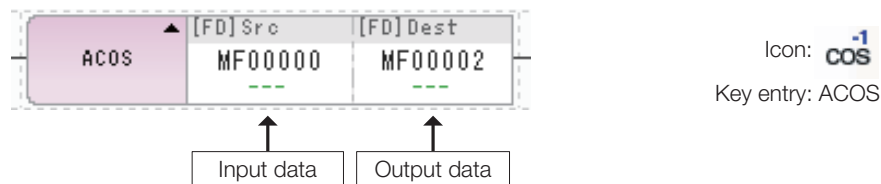
The arc cosine of the real number input data is calculated and the result is stored in the output data.

$$\text{COS} \left(\boxed{\text{Input data}} \right)^{-1} \longrightarrow \boxed{\text{Output data}}$$

Note: The output data is in degrees.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	×	×	×	×	○	○	×	×	○
Dest (Output data)	×	×	×	×	○*	○*	×	×	×

* C and # registers cannot be used.

Set the input data to a value between -1.0 and 1.0. The output is set to 0 if the input value is out of range.

Programming Example

In the following programming example, the arc sine of 0.5 in input data in MF00000 is calculated and the result is stored in the output data in DF00000.

$$\text{COS} (0.5)^{-1} \rightarrow \text{DF00000} = 60.0 \text{ (degrees)}$$



4.6.7 Arc Tangent (ATAN)

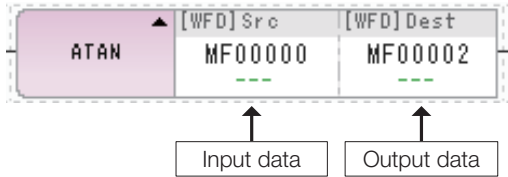
The arc tangent of the real number input data is calculated and the result is stored in the output data.


$$\text{TAN} \left(\boxed{\text{Input data}} \right)^{-1} \longrightarrow \boxed{\text{Output data}}$$

Note: The output data is in degrees.

Format

The format of this instruction is shown below.



Icon: 
Key entry: ATAN

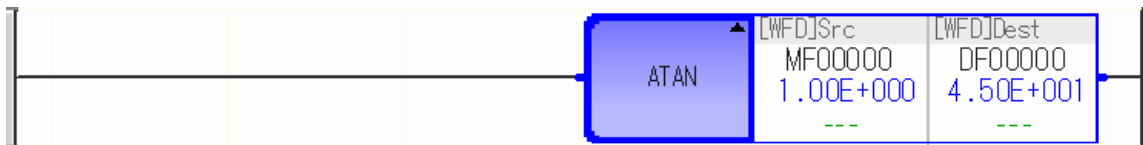
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	×	○	×	×	○	○	×	×	○
Dest (Output data)	×	○	×	×	○*	○*	×	×	×

* C and # registers cannot be used.

Programming Example

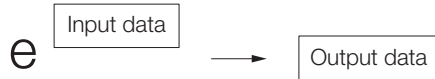
In the following programming example, the arc tangent of 1.0 in input data in MF00000 is calculated and the result is stored in the output data in DF00000.

$$\text{TAN} (1.0)^{-1} \rightarrow \text{DF00000} = 45.0 \text{ (degrees)}$$



4.6.8 Exponential (EXP)

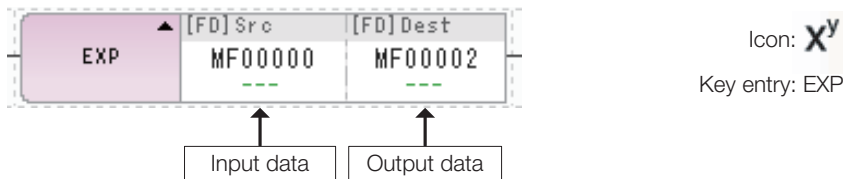
The value obtained by raising base e of the natural logarithm to the real number input data is calculated and the result is stored in the output data.



Note: “e” is the base of the natural logarithm.

Format

The format of this instruction is shown below.



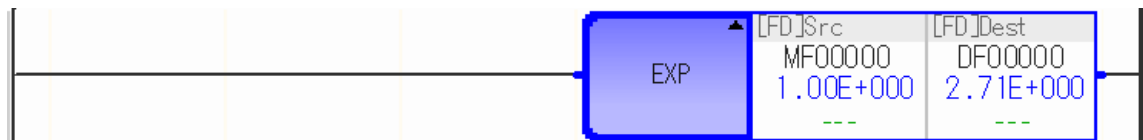
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	×	×	×	×	○	○	×	×	○
Dest (Output data)	×	×	×	×	○*	○*	×	×	×

* C and # registers cannot be used.

Programming Example

The following programming example calculates base e of the natural logarithm raised to 1.0 in the input data in MF00000, and stores the result in the output data in DF00000.

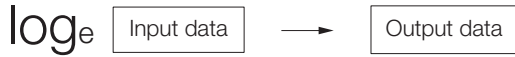
$$e^{1.0} \rightarrow \text{DF00000} = 2.718282$$



Information If the operation result overflows, the output data will be set to the maximum value $3.402\text{E}+38$ and an operation error will not occur.

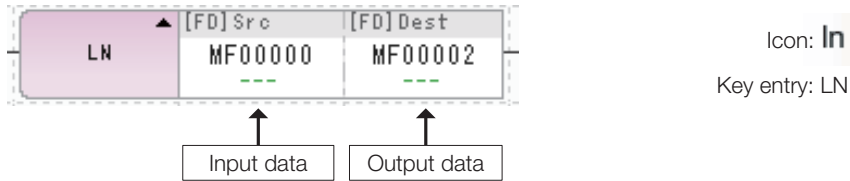
4.6.9 Natural Logarithm (LN)

The natural logarithm of X ($\log_e X$), when the real number input data is X, is calculated and the result is stored in the output data.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	x	x	x	x	○	○	x	x	○
Dest (Output data)	x	x	x	x	○*	○*	x	x	x

* C and # registers cannot be used.

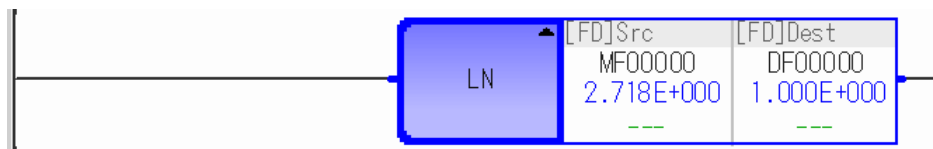
If the input data is less than 0, the absolute value of the input data will be used to perform the operation and output the result.

The output data is set to $-\infty$ if the input value is 0.

Programming Example

The following programming example calculates the natural logarithm when the input data is 2.718282 ($\approx e$) in MF00000, and stores the result in the output data in DF00000.

$$\log_e 2.718282 \approx \log_e e \rightarrow DF00000 = 1.0$$



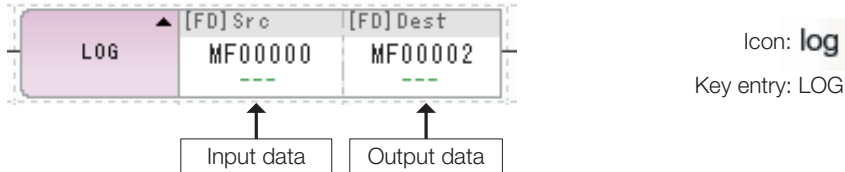
4.6.10 Common Logarithm (LOG)

The common logarithm of X ($\log_{10} X$), when the real number input data is X , is calculated and the result is stored in the output data.

$$\log_{10} \boxed{\text{Input data}} \longrightarrow \boxed{\text{Output data}}$$

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (Input data)	×	×	×	×	○	○	×	×	○
Dest (Output data)	×	×	×	×	○*	○*	×	×	×

* C and # registers cannot be used.

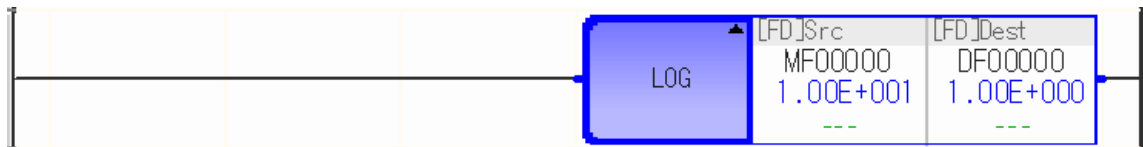
If the input data is less than 0, the absolute value of the input data will be used to perform the operation and output the result.

The output data is set to $-\infty$ if the input value is 0.

Programming Example

The following programming example calculates the common logarithm when the input data is 10.0 in MF00000, and stores the result in the output data in DF00000.

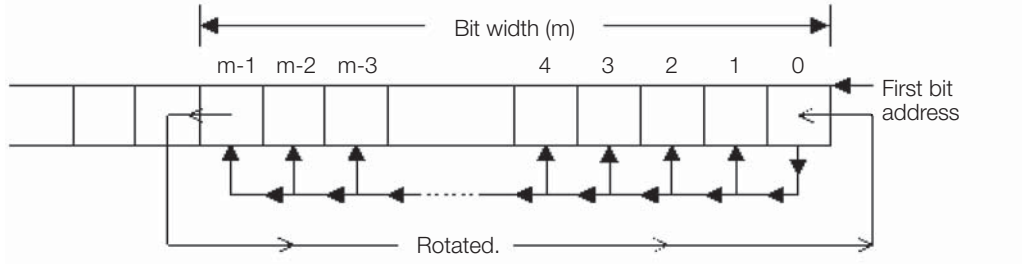
$$\log_{10} 10.0 \rightarrow DF00000 = 1.0$$



4.7 Data Shift Instructions

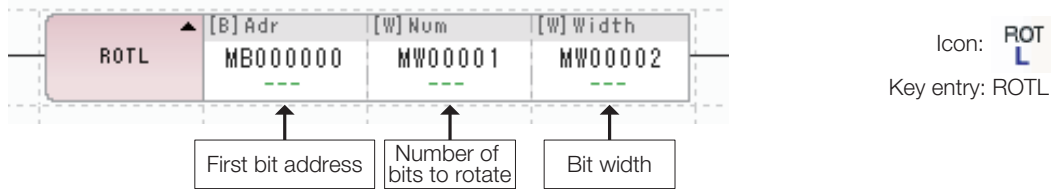
4.7.1 Bit Rotate Left (ROTL)

The data specified by the first bit address and bit width is rotated to the left by the specified number of bits.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Adr (First bit address)	O*	x	x	x	x	x	x	x	x
Num (Number of bits to rotate)	x	O	x	x	x	x	x	O	O
Width (Bit width)	x	O	x	x	x	x	x	O	O

* C and # registers cannot be used.

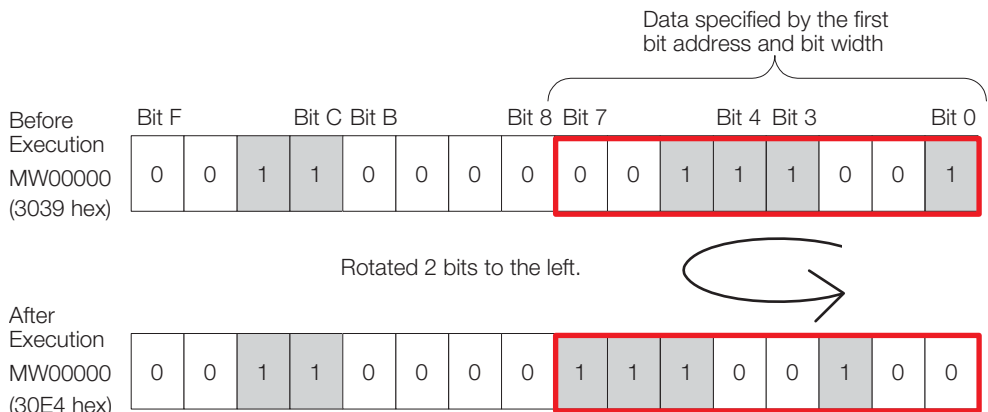
Programming Example

In the following programming example, the data specified as 8-bit wide from the first bit address at MB000000 is rotated to the left two bits.

The ROTL instruction is executed when switch 1 (DB000000) turns ON.

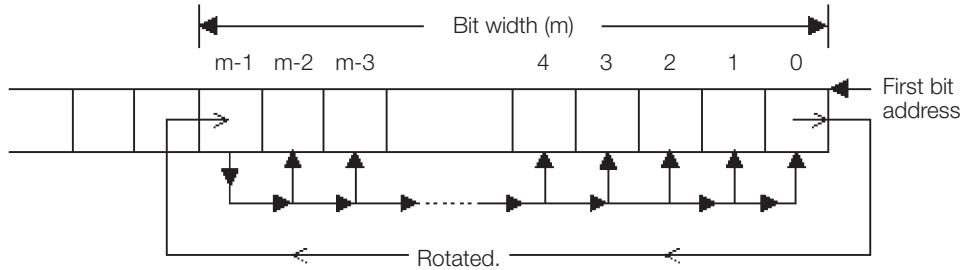


The following figure shows the operation when MW00000 is 12345 (3039 hex).



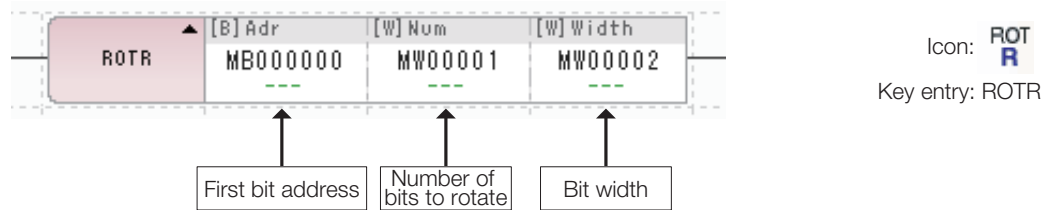
4.7.2 Bit Rotate Right (ROTR)

The data specified by the first bit address and bit width is rotated to the right by the specified number of bits.



Format

The format of this instruction is shown below.



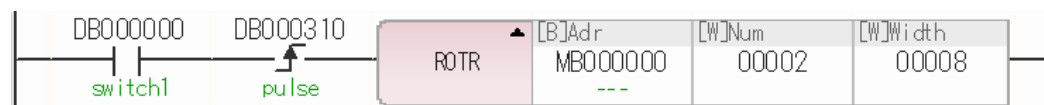
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Adr (First bit address)	O*	x	x	x	x	x	x	x	x
Num (Number of bits to rotate)	x	O	x	x	x	x	x	O	O
Width (Bit width)	x	O	x	x	x	x	x	O	O

* C and # registers cannot be used.

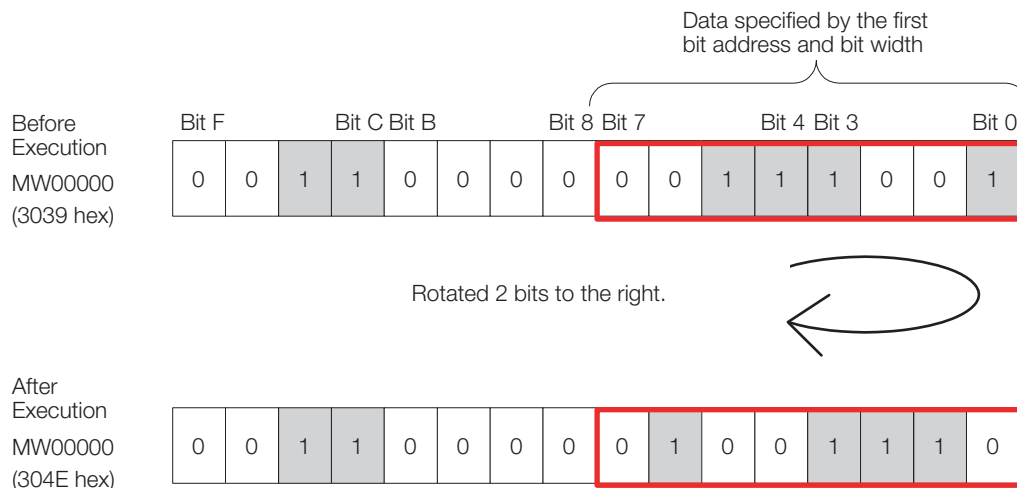
Programming Example

In the following programming example, the data specified as 8-bit wide from the first bit address at MB000000 is rotated to the right two bits.

The ROTR instruction is executed when switch 1 (DB000000) turns ON.

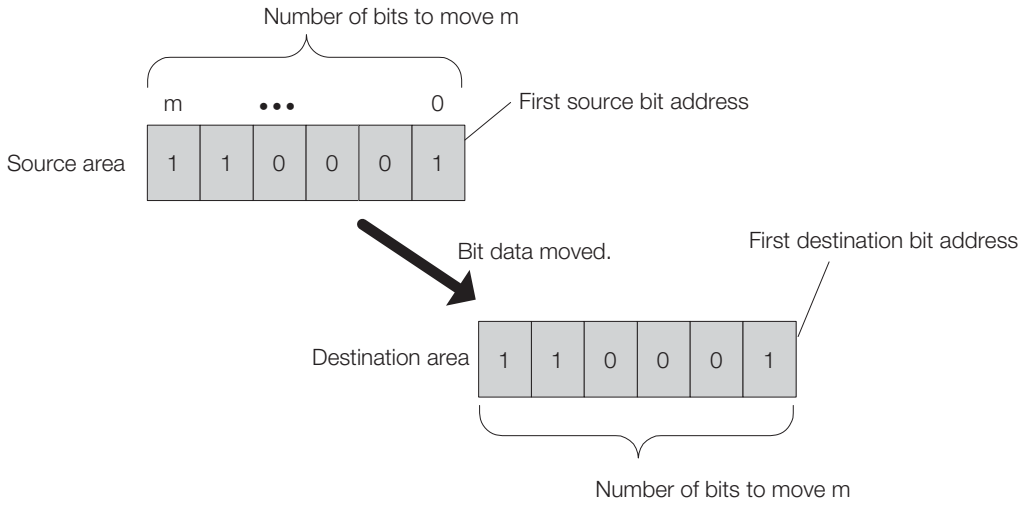


The following figure shows the operation when MW00000 is 12345 (3039 hex).



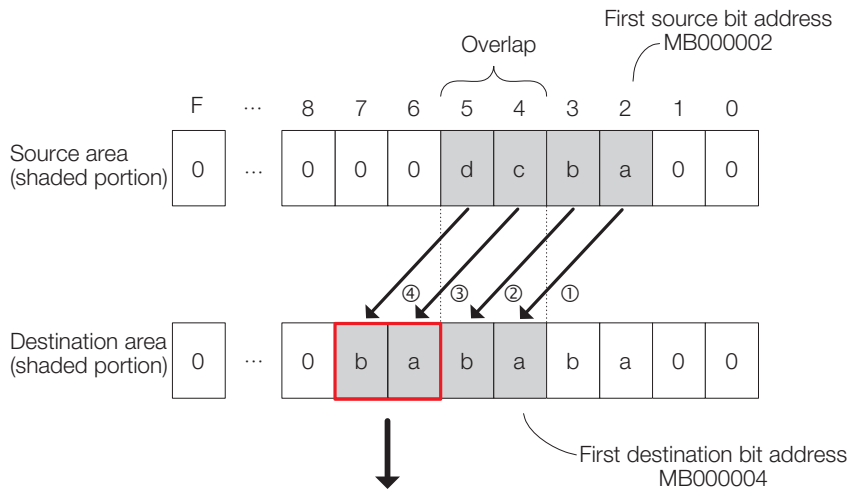
4.7.3 Move Bit (MOVB)

The designated number of bits of data is moved from memory starting at the first source bit address to memory starting at the first destination bit address.



Note: The bits are moved one bit at a time from the lowest relay address. If the source area and destination area overlap, the source data that is actually moved may not be the data that was in the source area when the instruction was executed.

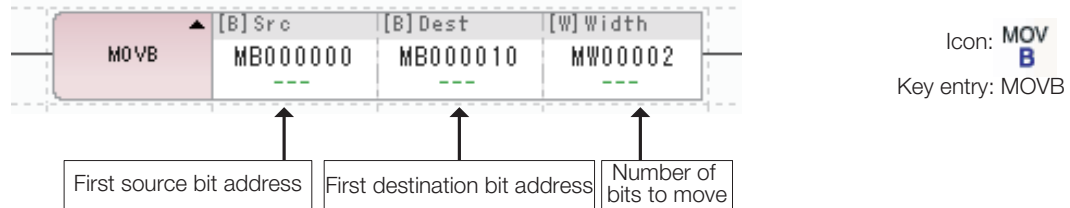
The following diagram shows an example where the source area and destination area overlap.



Bit status is moved in the following order: ① to ④. This means that the status of bits 2 and 3 are moved to bits 4 and 5 (① and ②) and then the status of bits 4 and 5 are moved (③ and ④).

Format

The format of this instruction is shown below.



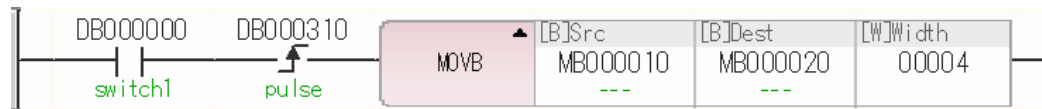
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (First source bit address)	○	×	×	×	×	×	×	×	×
Dest (First destination bit address)	○*	×	×	×	×	×	×	×	×
Width (Number of bits to move)	×	○	×	×	×	×	×	○	○

* C and # registers cannot be used.

Programming Example

In the following programming example, 4 bits of data starting from the first source bit address at MB000010 are moved to memory starting as the first destination bit address at MB000020.

The MOVE instruction is executed when switch 1 (DB000000) turns ON.

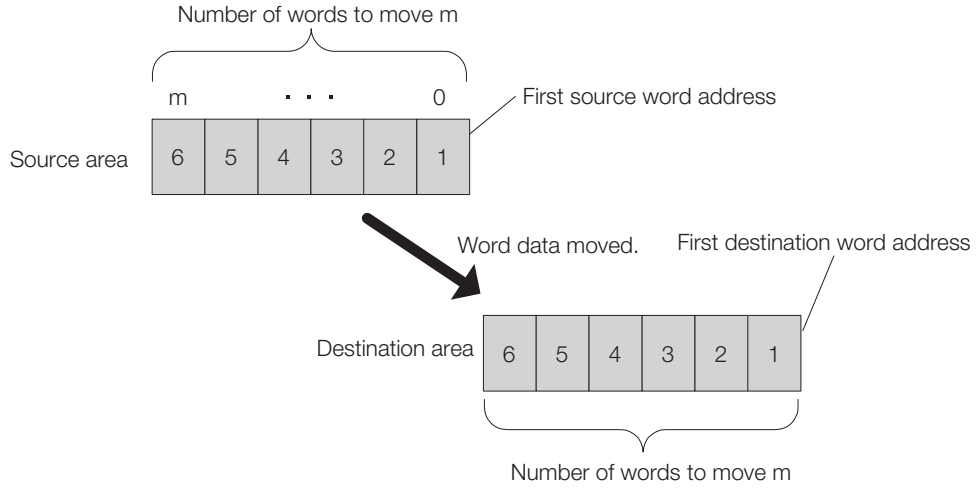


The following table illustrates how the data in the source area is moved to the destination area.

Source area		Destination area		
Register	Data	Register	Data before Execution of Instruction	Data after Execution of Instruction
MB000010	0	MB000020	0	0
MB000011	1	MB000021	0	1
MB000012	1	MB000022	0	1
MB000013	1	MB000023	0	1

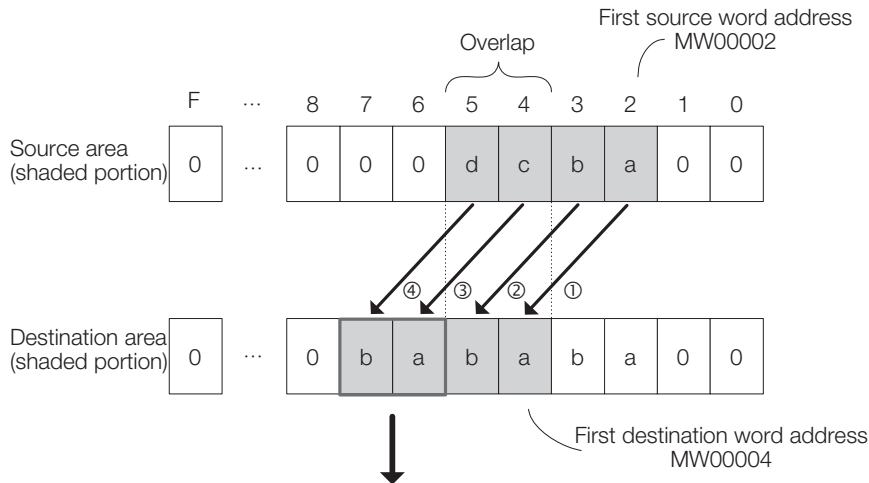
4.7.4 Move Word (MOVW)

The designated number of words of data are moved from memory starting at the first source word address to memory starting at the first destination word address.



Note: The words are moved one word at a time from the lowest register address. If the source area and destination area overlap, the source data that is actually moved may not be the data that was in the source area when the instruction was executed.

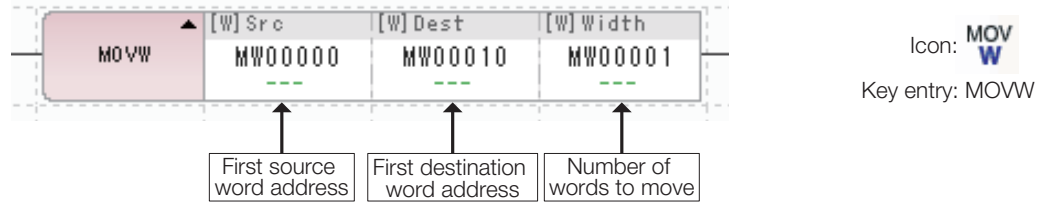
The following diagram shows an example where the source area and destination area overlap.



Word contents are moved in the following order: ① to ④. This means that the contents of MW00002 and MW00003 are moved to MW00004 and MW00005 (① and ②) and then the contents of MW00004 and MW00005 are moved (③ and ④).

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (First source word address)	×	○	×	×	×	×	×	○	×
Dest (First destination word address)	×	○*	×	×	×	×	×	○	×
Width (Number of words to move)	×	○	×	×	×	×	×	○	○

* C and # registers cannot be used.

Programming Example

In the following programming example, 4 words of data starting from the first source word address at MW00010 are moved to memory starting at the first destination word address at MW00020.

The MOVW instruction is executed when switch 1 (DB000000) turns ON.



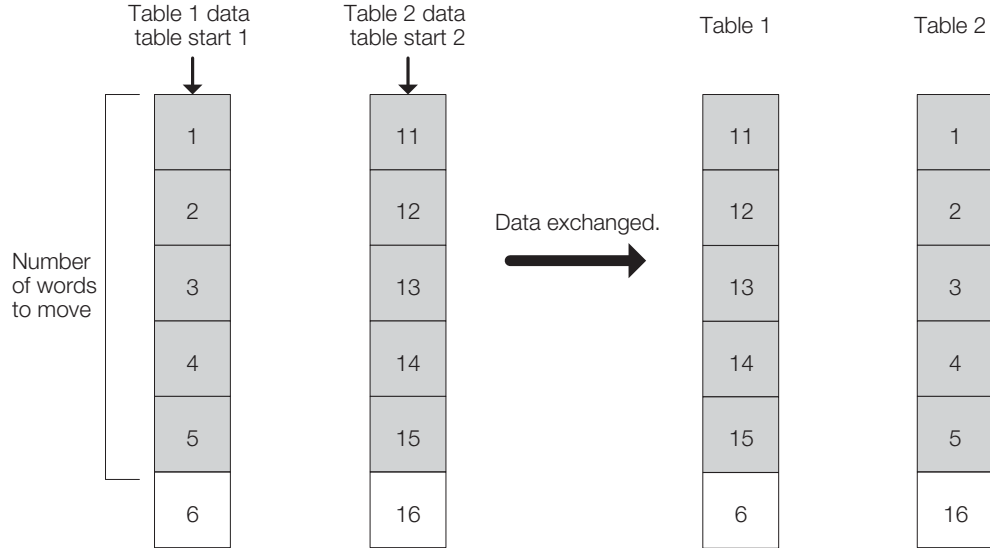
The following table illustrates how the data in the source area is moved to the destination area.

Source area		Destination area		
Register	Data	Register	Data before Execution of Instruction	Data after Execution of Instruction
MW00010	10	MW00020	0	10
MW00011	20	MW00021	0	20
MW00012	30	MW00022	0	30
MW00013	40	MW00023	0	40

4.7.5 Exchange (XCHG)

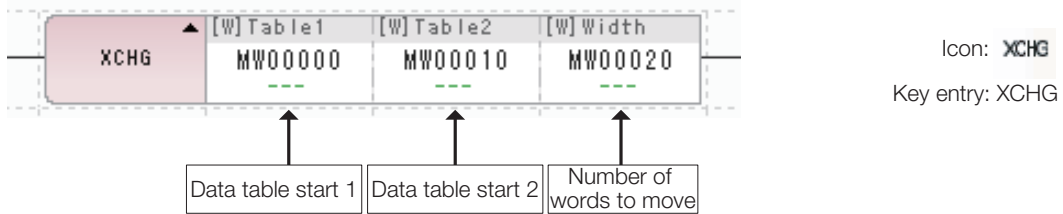
The designated number of data items are exchanged between table 1 and table 2.

The data contents of table 1 and table 2 specified by data table start 1, data table start 2, and the number of words to move are exchanged.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Table1 (Data table start 1)	x	O*	x	x	x	x	x	x	x
Table2 (Data table start 2)	x	O*	x	x	x	x	x	x	x
Width (Number of words to move)	x	O	x	x	x	x	x	O	O

* C and # registers cannot be used.

Programming Example

In the following programming example, 4 words of data are exchanged between table 1, which starts at MW00010, and table 2, which starts at MW00020.

The XCHG instruction is executed when switch 1 (DB000000) turns ON.

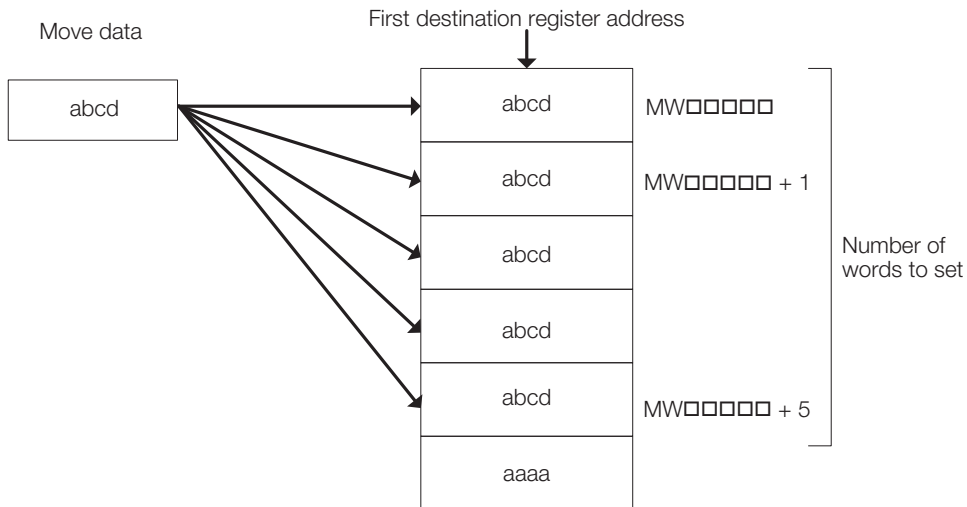


The following table illustrates how the data is exchanged between table 1 and table 2.

Table 1			Table 2		
Register	Data before Execution of Instruction	Data after Execution of Instruction	Register	Data before Execution of Instruction	Data after Execution of Instruction
MW00010	10	123	MW00020	123	10
MW00011	20	234	MW00021	234	20
MW00012	30	345	MW00022	345	30
MW00013	40	456	MW00023	456	40

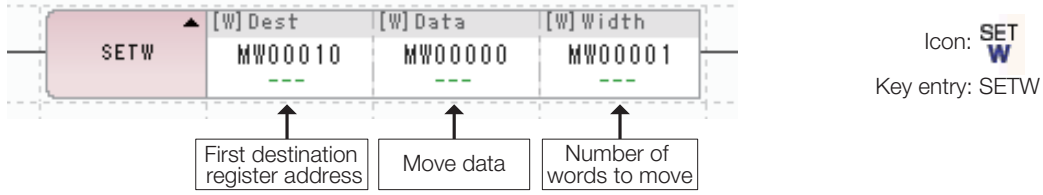
4.7.6 Table Initialization (SETW)

The designated data is stored in all registers in the area designated by the first register address and number of words to set. The data is stored one word at a time from the lowest register address to the highest.



Format

The format of this instruction is shown below.

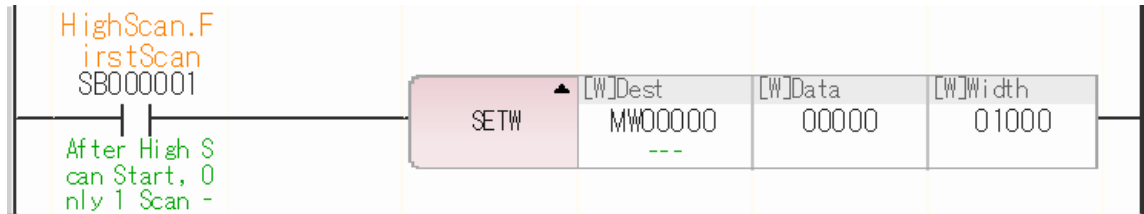


I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Dest (First destination register address)	×	○*	×	×	×	×	×	×	×
Data (Move data)	×	○	×	×	×	×	×	○	○
Width (Number of words to move)	×	○	×	×	×	×	×	○	○

* C and # registers cannot be used.

Programming Example

In the following programming example, the area of 1,000 words from MW00000 is initialized to the move data 0 on the first scan of the high-speed scan after the power is turned ON.

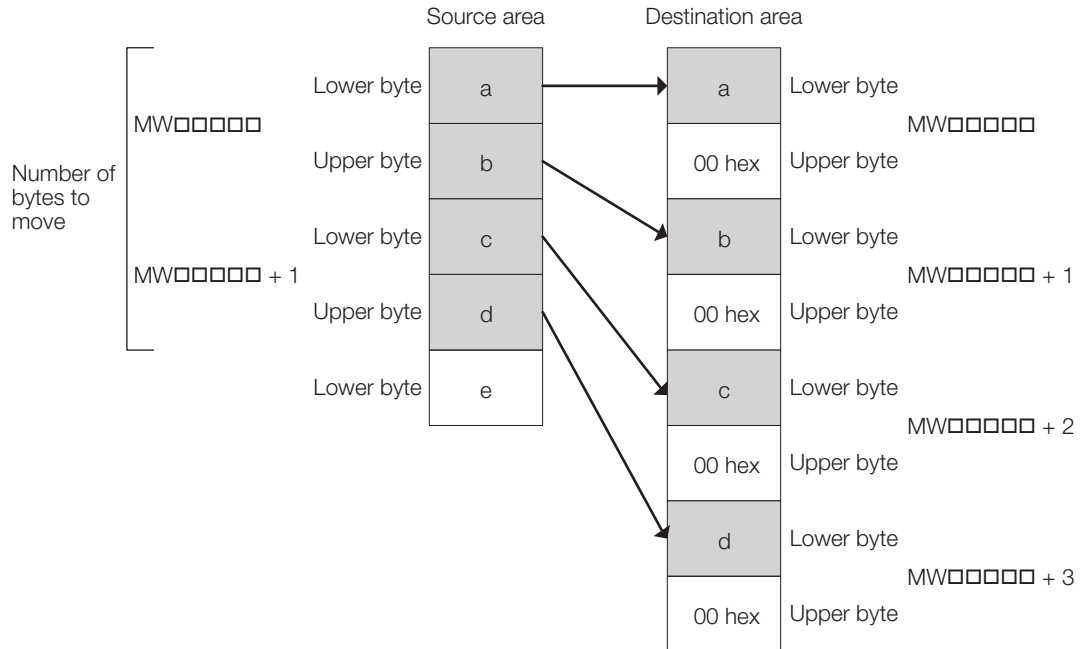


The following table illustrates how the registers are initialized to 0 after execution of the first scan of the high-speed scan when the power is turned ON.

Register	Data
MW00000	0
MW00001	0
:	:
MW00998	0
MW00999	0

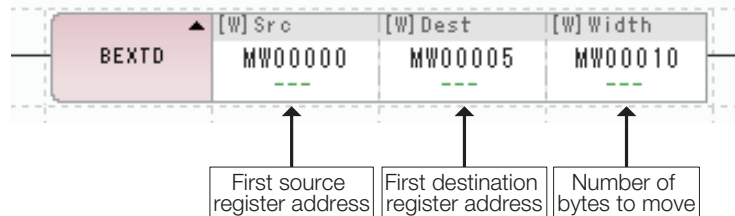
4.7.7 Byte-to-word Expansion (BEXTD)

The byte data of an area designated by the number of bytes from the first source register address is expanded into individual word data, one byte at a time, and moved to an area designated by the number of bytes from the first destination register address. When the byte is expanded into a word, the upper byte is set to 0.



Format

The format of this instruction is shown below.



Icon: Key entry: BEXTD

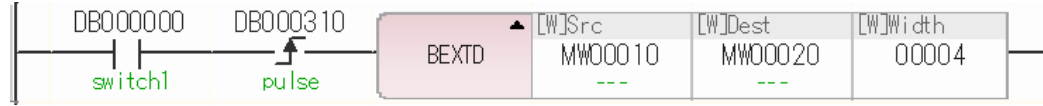
I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (First source register address)	×	○	×	×	×	×	×	×	×
Dest (First destination register address)	×	○*	×	×	×	×	×	×	×
Width (Number of bytes to move)	×	○	×	×	×	×	×	○	○

* C and # registers cannot be used.

Programming Example

In the following programming example, the data from an area of 4 bytes that starts from the first source register address at MW00010 is moved to an area of 4 bytes that starts from the first destination byte address at MW00020.

The BEXTD instruction is executed when switch 1 (DB000000) turns ON.



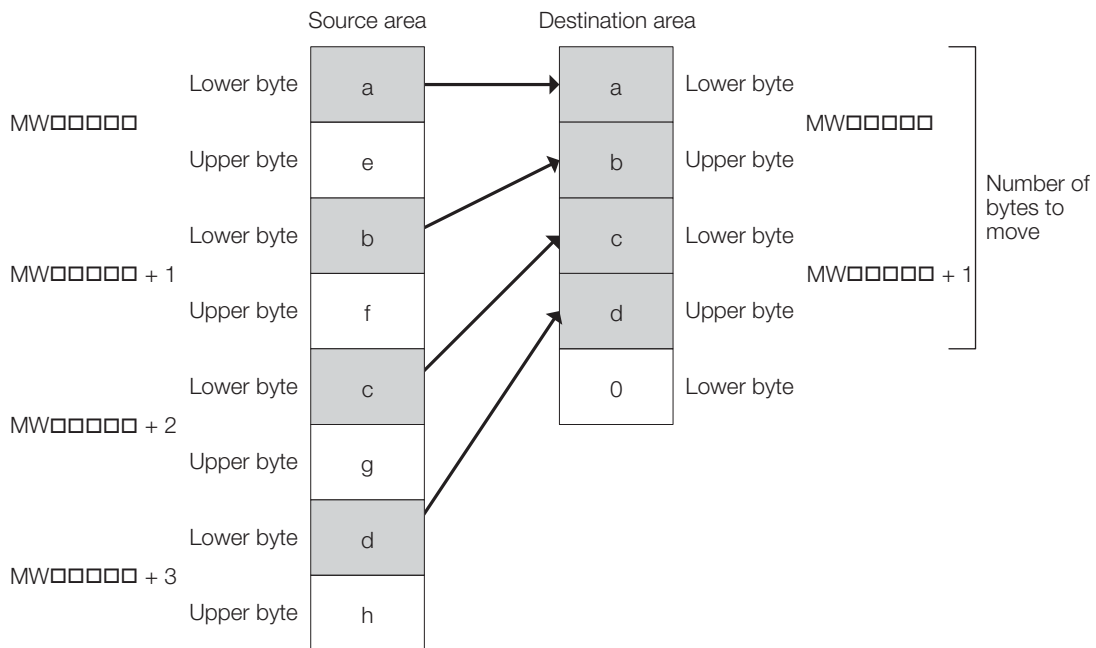
The following table illustrates how the byte data in the source area is expanded and moved into word data in the destination area.

Source area				Destination area		
Register		Data		Register		Data
MW00010	Lower byte	10 hex	⇒	MW00020	Lower byte	10 hex
	Upper byte	20 hex			Upper byte	00 hex
MW00011	Lower byte	30 hex		MW00021	Lower byte	20 hex
	Upper byte	40 hex			Upper byte	00 hex
				MW00022	Lower byte	30 hex
					Upper byte	00 hex
				MW00023	Lower byte	40 hex
					Upper byte	00 hex

4.7.8 Word-to-byte Compression (BPRESS)

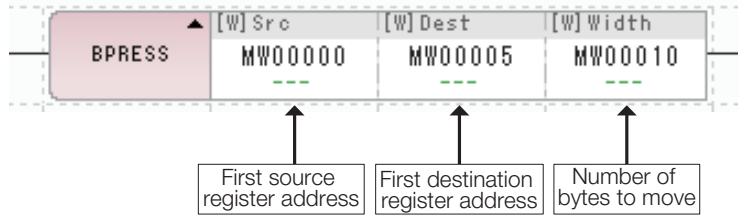
The lower byte of word data from the designated number of bytes starting from the first source register address is stored in the designated number of bytes starting at the first destination registration address, one byte at a time. This instruction performs the opposite operation of the BEXTD instruction.

The upper byte is discarded.



Format

The format of this instruction is shown below.



Icon: 
Key entry: BPRESS

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (First source register address)	×	○	×	×	×	×	×	×	×
Dest (First destination register address)	×	○*	×	×	×	×	×	×	×
Width (Number of bytes to move)	×	○	×	×	×	×	×	○	○

* C and # registers cannot be used.

Programming Example

In the following programming example, the lower byte of data from an area of 4 bytes that starts from the first source register address at MW00010 is moved to an area of 4 bytes that starts from the first destination register address at MW00020.

The BPRESS instruction is executed when switch 1 (DB000000) turns ON.

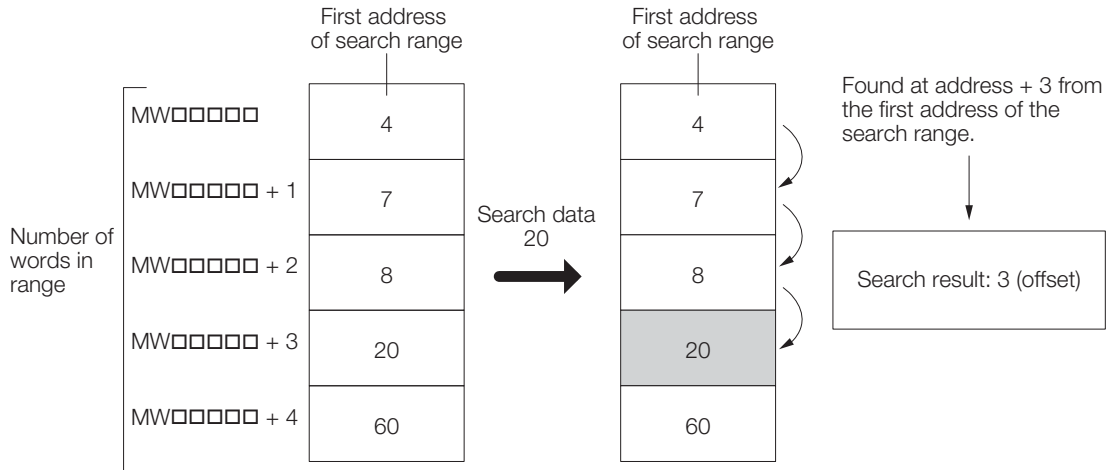


The following table illustrates how the word data in the source area is compressed and moved into byte data in the destination area.

Source area			⇒	Destination area		
Register		Data		Register		Data
MW00010	Lower byte	12 hex		MW00020	Lower byte	12 hex
	Upper byte	23 hex			Upper byte	34 hex
MW00011	Lower byte	34 hex		MW00021	Lower byte	56 hex
	Upper byte	45 hex			Upper byte	78 hex
MW00012	Lower byte	56 hex				
	Upper byte	67 hex				
MW00013	Lower byte	78 hex				
	Upper byte	89 hex				

4.7.9 Binary Search (BSRCH)

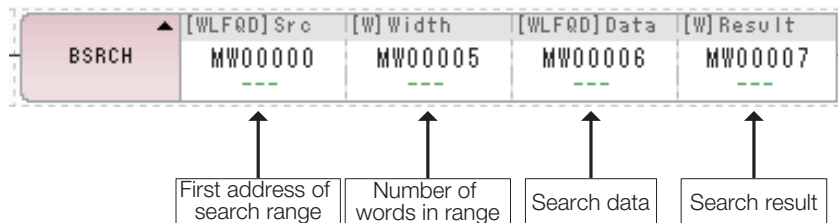
A search is made for the search data using a binary search method in the area designated by the number of words from the first address of the search range. The search result is output as the offset word number of the data that matches the search data from the first register in the start range. Always sort the data in the search range in ascending order.




- Note: 1. Always sort the search area in ascending order before executing the BSRCH instruction.
 2. The conceptual diagram shown here is for integers. The instruction operates in the same way for double-length integers and real numbers.

Format

The format of this instruction is shown below.



Icon: 
 Key entry: BSRCH

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (First address of search range)	×	○	○	○	○	○	×	×	×
Width (Number of words in range)	×	○	×	×	×	×	×	○	○
Data (Search data)	×	○	○	○	○	○	×	○	○
Result (Search result)	×	○*	×	×	×	×	×	×	×

* C and # registers cannot be used.



Term

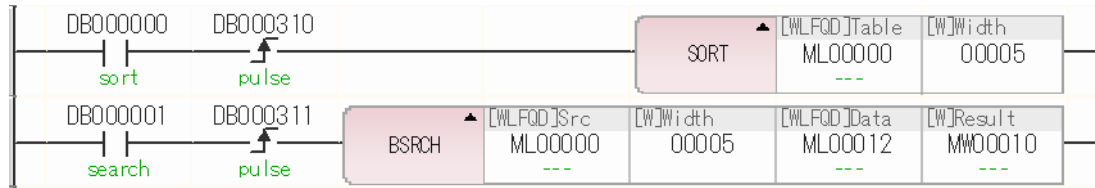
Binary Search

A binary search is a data searching algorithm that is used to quickly search for data in a sorted search area.

First, the median value of the search area is compared to the search data. If the search data is greater than the median value, the same search procedure is performed in the search area to the right of the median value. If the search data is less than the median value, the same search procedure is performed in the search area to the left of the median value. To use this search method, the data must first be sorted in ascending order.

Programming Example

The data from ML00000 to ML00008 is sorted when the sort command (DB00000) turns ON. Then, if the search command (DB000001) turns ON, the search data in ML00012 is searched for in the sorted data area.



The following table shows how the sort is processed when the first line is executed. Here, the data from ML00000 to ML00008 is as listed below, and the search data in ML00012 is 70. When the second line is executed, the search result in MW00010 is set to 4 as the result of finding 70.

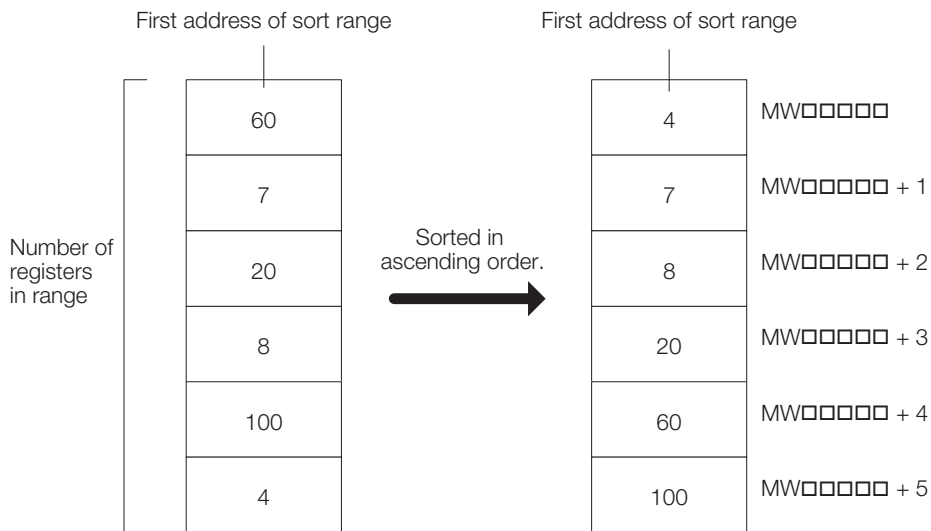
Register	Data before Execution of 1st Line	Data after Execution of 1st Line	Execution Result of 2nd Line
ML00000	100	15	ML00004 = 70, so MW00010 = 4
ML00002	30	30	
ML00004	90	70	
ML00006	15	90	
ML00008	70	100	

4.7.10 Sort (SORT)

The data in the range of registers from the first address of the sort range is sorted in ascending order.

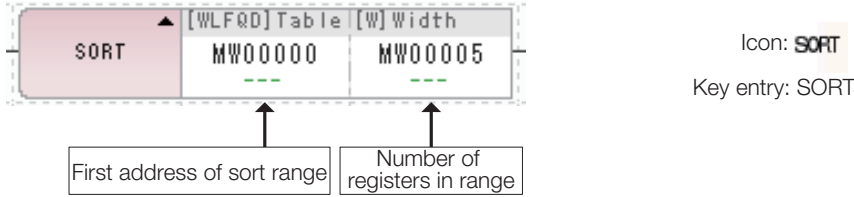
The following diagram describes the operation using integers as an example. The sort is performed in the same way for double-length integers and real numbers.

The maximum number of data items for a sort is 128.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Table (First address of sort range)	x	O*	O*	O*	O*	O*	x	x	x
Width (Number of registers in range)	x	O	x	x	x	x	x	O	O

* C and # registers cannot be used.

Programming Example

In the following programming example, the data from ML00000 to ML00008 is sorted in ascending order when the sort command (DB00000) turns ON.



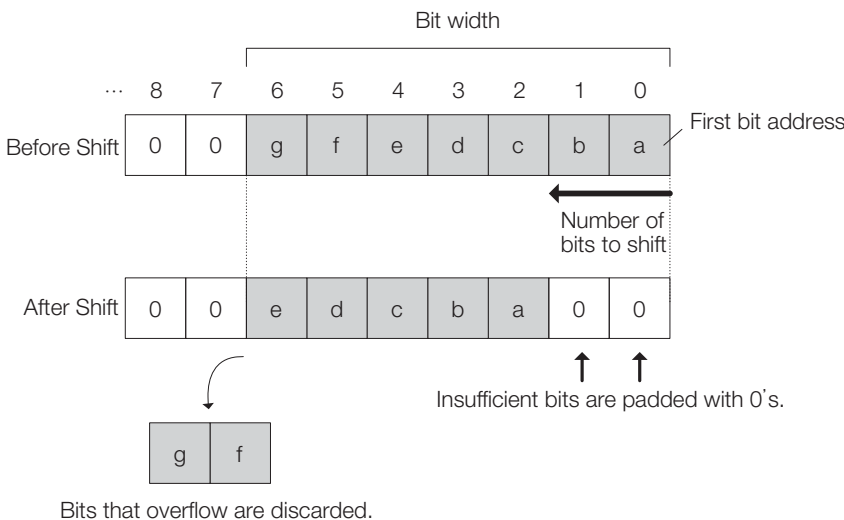
The following table shows how the data from ML00000 to ML00008 is sorted when the SORT instruction is executed.

Register	Data before Execution of Instruction	Data after Execution of Instruction
ML00000	100	15
ML00002	30	30
ML00004	90	70
ML00006	15	90
ML00008	70	100

4.7.11 Bit Shift Left (SHFTL)

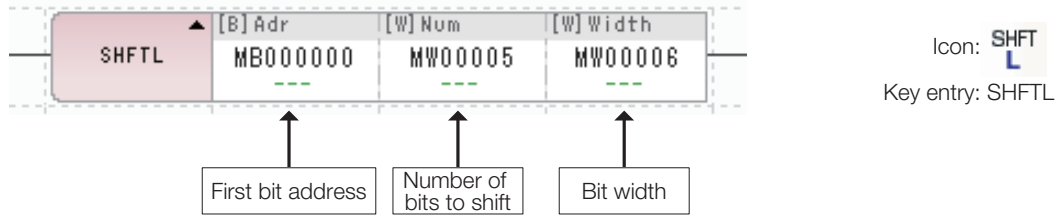
The bits specified by the first bit address and bit width are shifted to the left by the specified number of bits.

Data that overflows from the bit width is discarded and insufficient bits are padded with 0's



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Adr (First bit address)	○*	×	×	×	×	×	×	×	×
Num (Number of bits to shift)	×	○	×	×	×	×	×	○	○
Width (Bit width)	×	○	×	×	×	×	×	○	○

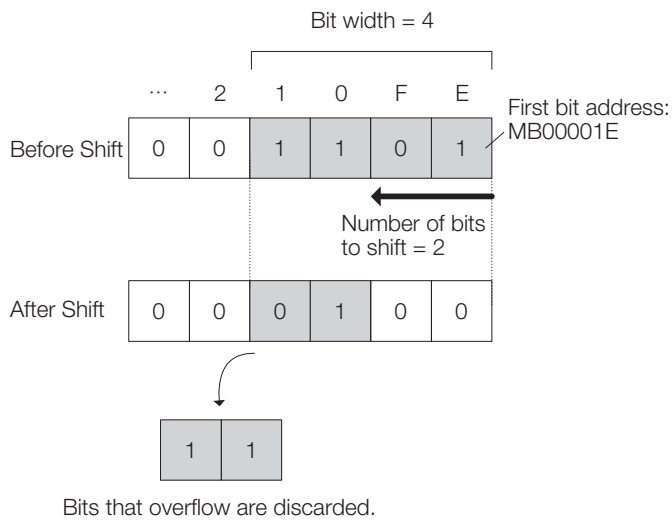
* C and # registers cannot be used.

Programming Example

In the following programming example, 4 bits from the first bit address at MB00001E are shifted two bits to the left when switch 1 (DB000000) turns ON.



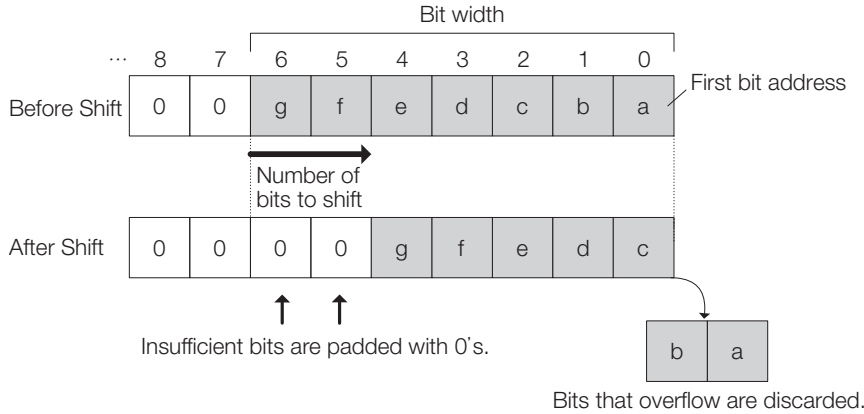
The following figure illustrates the result when the above program is executed.



4.7.12 Bit Shift Right (SHFTR)

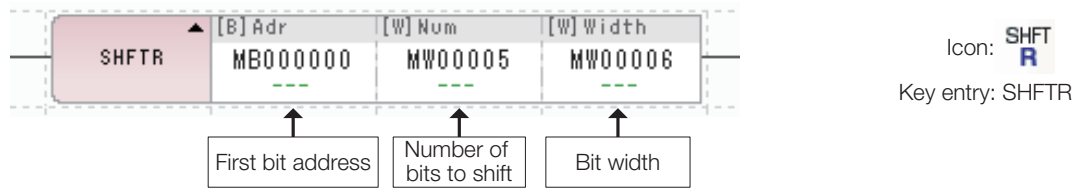
The bits specified by the first bit address and bit width are shifted to the right by the specified number of bits.

Data that overflows from the bit width is discarded and insufficient bits are padded with 0's



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Adr (First bit address)	O*	x	x	x	x	x	x	x	x
Num (Number of bits to shift)	x	O	x	x	x	x	x	O	O
Width (Bit width)	x	O	x	x	x	x	x	O	O

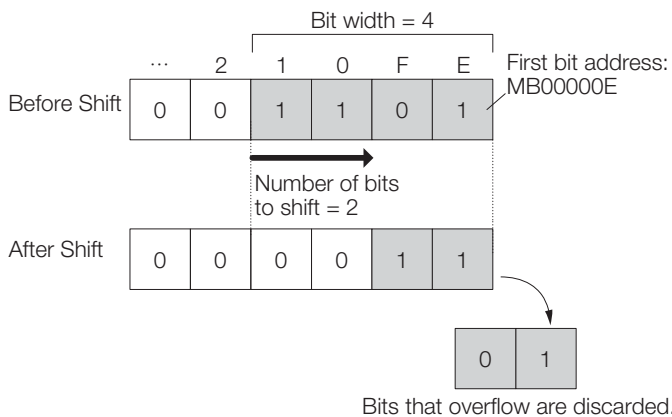
* C and # registers cannot be used.

Programming Example

In the following programming example, 4 bits from the first bit address at MB00001E are shifted two bits to the right when switch 1 (DB000000) turns ON.



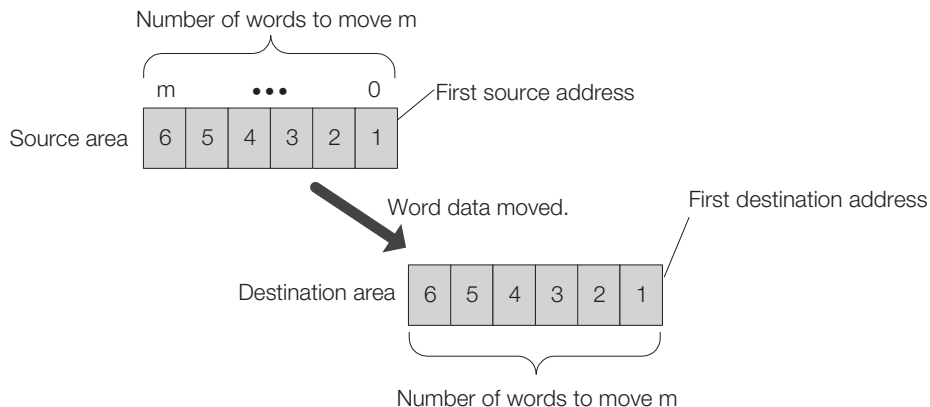
The following figure illustrates the result when the above program is executed.



4.7.13 Copy Word (COPYW)

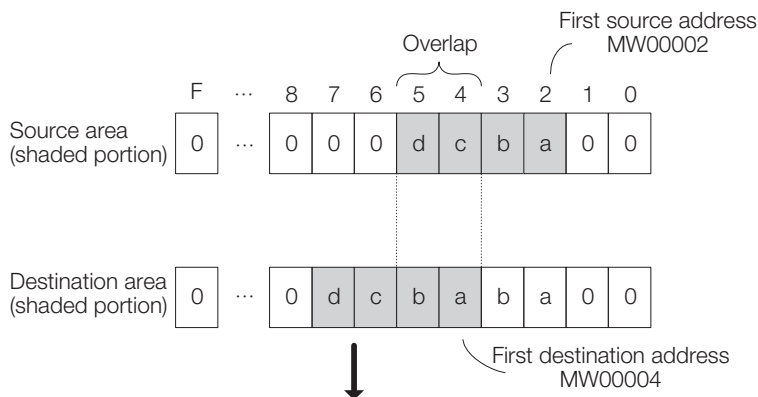
The word data in the area designated by the specified number of words is copied from the source area to the destination area.

The data for each block is copied from the source to the destination. Unlike the MOVW instruction, the data is copied to the destination as is, even if the source and destination overlap.



Note: This instruction differs from the MOVW instruction by the way it handles overlap between the source and destination areas.

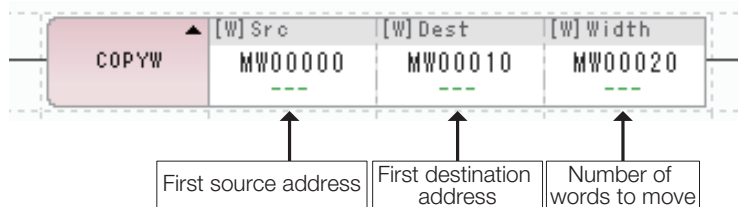
The following diagram shows an example where the source area and destination area overlap.




Unlike the MOVW instruction, all of the data in the source area is moved to the destination area, even if the two areas overlap.

Format

The format of this instruction is shown below.



Icon: 
Key entry: COPYW

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src (First source address)	×	○	×	×	×	×	×	○	×
Dest (First destination address)	×	○*	×	×	×	×	×	○	×
Width (Number of words to move)	×	○	×	×	×	×	×	○	○

* C and # registers cannot be used.

Programming Example

In the following programming example, 5 words of data starting from the first source address at MW00000 are copied to an area of 5 words that starts from the first destination address at MW00100 when switch 1 (DB000000) turns ON.

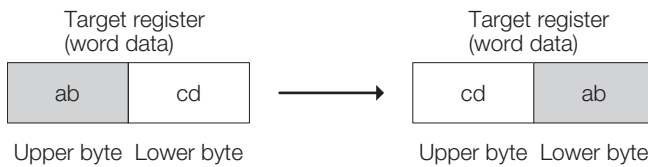


The following figure illustrates the result when the above program is executed.

Register	Data	Register	Data before Execution of Instruction	Data after Execution of Instruction
MW00000	1	MW00100	123	1
MW00001	2	MW00101	234	2
MW00002	3	MW00102	345	3
MW00003	4	MW00103	456	4
MW00004	5	MW00104	567	5

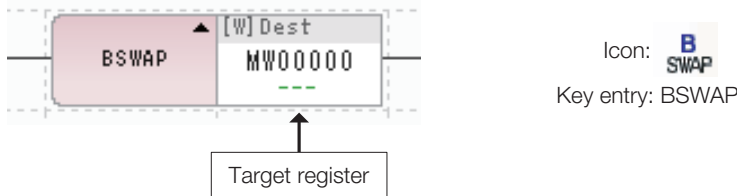
4.7.14 Byte Swap (BSWAP)

The upper byte and lower byte of the target register are swapped.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Dest (Target register)	x	O*	x	x	x	x	x	x	x

* C and # registers cannot be used.

Programming Example

In the following programming example, the upper byte and lower byte of the target register (MW00000) are swapped when switch 1 (DB000000) turns ON.

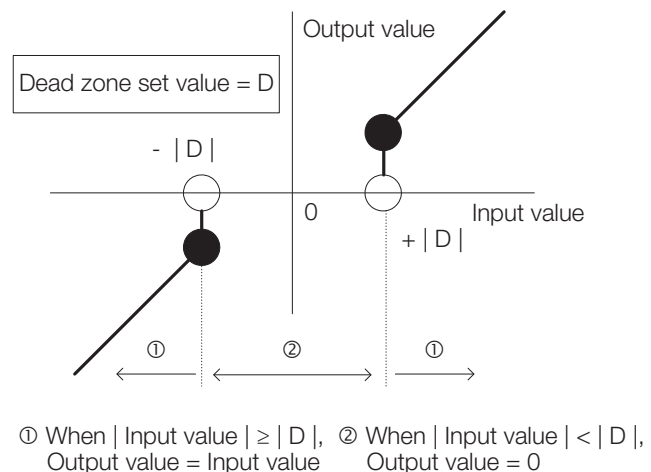
When MW00000 is 00FF hex, MW00000 will be FF00 hex after execution of the BSWAP instruction.



4.8 DDC Instructions

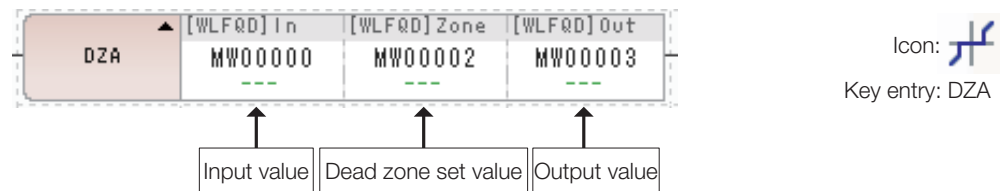
4.8.1 Dead Zone A (DZA)

The output value is calculated by comparing the input value against a predefined dead zone. As shown in the following figure, if the absolute value of the input value is greater than or equal to the absolute value of D, the input value is outside of the dead zone, so it becomes the output value. If the absolute value of the input value is less than the absolute value of D, the input value is inside of the dead zone, so the output is set to 0.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In (Input value)	x	o	o	o	o	o	x	o	o
Zone (Dead zone set value)	x	o	o	o	o	o	x	o	o
Out (Output value)	x	o*	o*	o*	o*	o*	x	o	x

* C and # registers cannot be used.

Programming Examples

In the following programming examples, the operation results are stored as the output value (MW00000) when the dead zone set value is set to 10,000.

The output values are calculated with respect to the input values in MW00001 to MW00003 as shown below.

• **Outside of the Dead Zone**

| MW00001(12,345) | ≥ | 10000 | so, MW00000 is 12,345.



| MW00002 (-12,345) | ≥ | 10,000 | so, MW00000 is -12,345.



• **Inside of the Dead Zone**

| MW00003 (6,789) | < | 10,000 | so, MW00000 is 0.

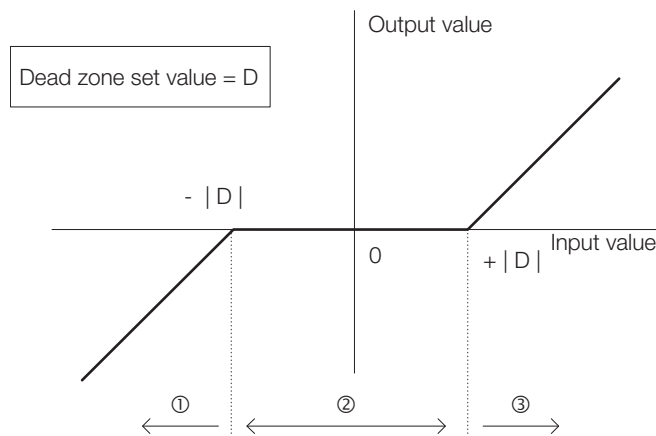


4.8.2 Dead Zone B (DZB)

The output value is calculated by comparing the input value against a predefined dead zone.

As shown in the following figure, if the absolute value of the input value is less than the absolute value of D, the input value is inside of the dead zone, so the output is set to 0.

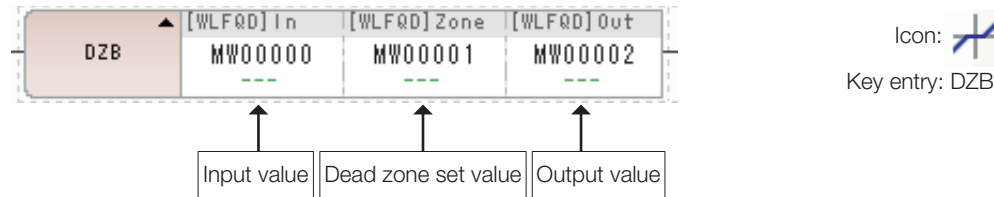
Unlike the DZA instruction, when the input value is outside of the dead zone, the sign of the input value determines whether the output value is obtained by adding the absolute value to or subtracting it from the input value.



- ① If Input value < 0 and | Input value | ≥ | D |
Output value = Input value + | D |
- ② If | Input value | < | D |
Output value = 0
- ③ If Input value ≥ 0 and | Input value | ≥ | D |
Output value = Input value - | D |

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In (Input value)	x	o	o	o	o	o	x	o	o
Zone (Dead zone set value)	x	o	o	o	o	o	x	o	o
Out (Output value)	x	o*	o*	o*	o*	o*	x	o	x

* C and # registers cannot be used.

Programming Examples

In the following programming examples, the operation results are stored as the output value (MW00000) when the dead zone set value is set to 10,000.

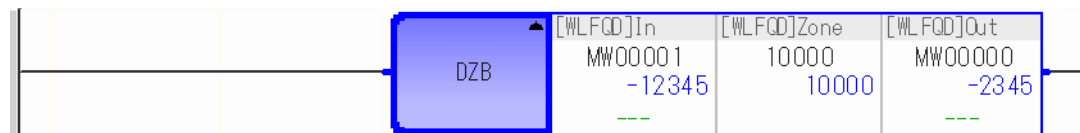
The output values are calculated with respect to the input values in MW00001 as shown below.

- **Outside of the Dead Zone**

Because $MW00001 (12,345) \geq 0$ and $|MW00001 (12,345)| \geq |10000|$, $MW00000 = 12,345 - |10,000| = 2,345$.



$|MW00001 (-12,345)| < 0$, $|MW00001 (-12,345)| \geq |10,000|$ so, $MW00000 = -12,345 + |10,000| = -2,345$.



- **Inside of the Dead Zone**

$|MW00001 (6789)| < |10000|$ so, MW00000 becomes 0.

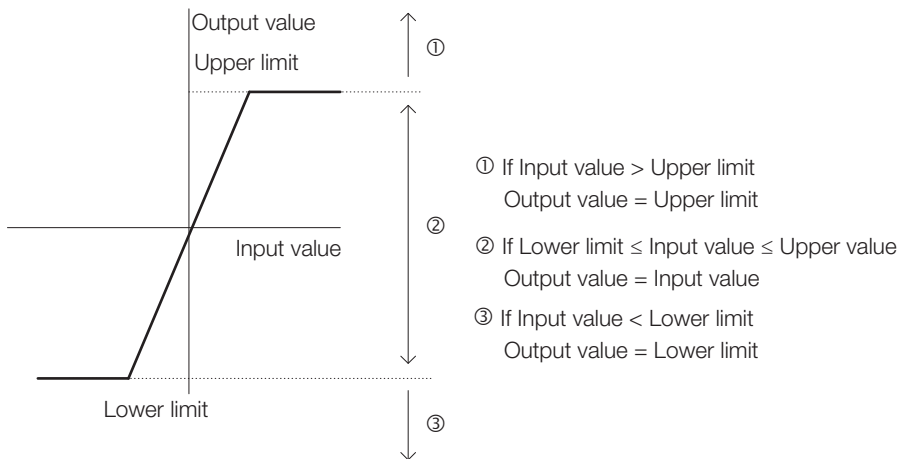


4.8.3 Upper/Lower Limit (LIMIT)

The output value is controlled so that it does not exceed the specified upper and lower limits for the input value.

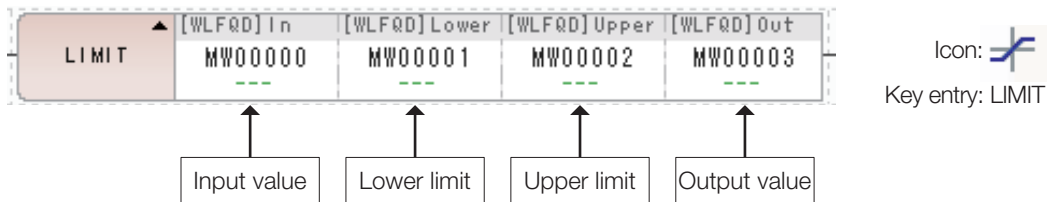
As shown in the following figure, if the input value is within the upper and lower limits, the input value is output unaltered.

The upper limit is output when the input value is greater than upper limit. The lower limit is output when the input value is less than the lower limit.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In (Input value)	×	○	○	○	○	○	×	○	○
Lower (Lower limit)	×	○	○	○	○	○	×	○	○
Upper (Upper limit)	×	○	○	○	○	○	×	○	○
Out (Output value)	×	○*	○*	○*	○*	○*	×	○	×

* C and # registers cannot be used.

Information Always set the lower limit to a value that is less than or equal to the upper limit.

Programming Examples

In the following programming examples, the operation results are stored as the output value (MW00000) when the lower limit is -100 and the upper limit is 10,000.

The output values are calculated with respect to the input values in MW00001 as shown below.

- **The Input Value Is Outside of the Upper and Lower Limits**

Because MW00001 (12,345) is greater than the upper limit (10,000), MW00000 becomes the upper limit (10,000).

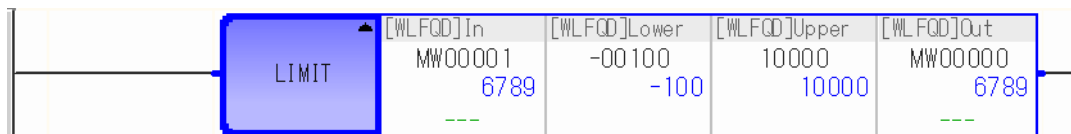


Because MW00001 (-12,345) is less than the lower limit (-100), MW00000 becomes the lower limit (-100).



- **The Input Value Is Within the Upper and Lower Limits**

Because the lower limit (-100) is less than MW00001 (6,789), which is less than the upper limit (10,000), MW00000 becomes 6,789.



4.8.4 PI Control (PI)

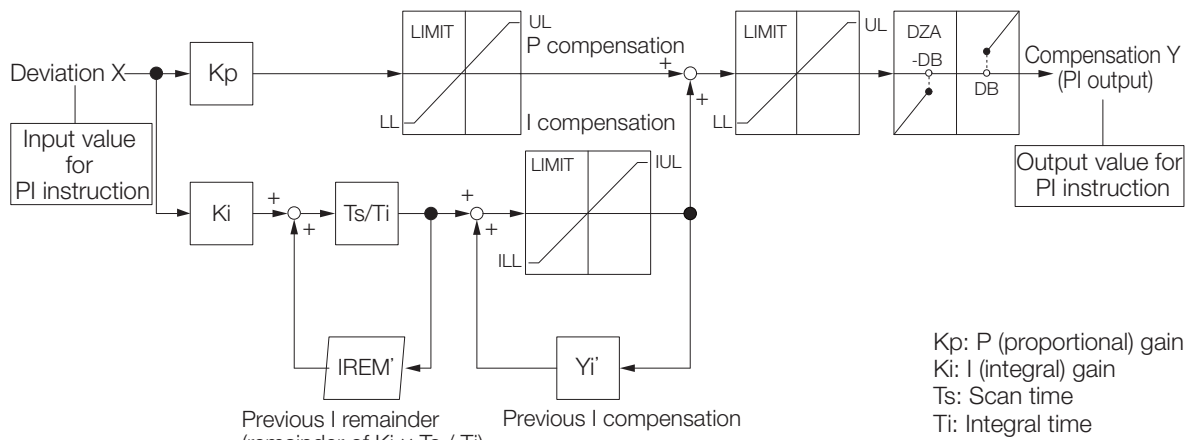
When deviation X is input, P and I operations and a range operation are performed based on predefined parameters in a parameter table, and the result is output as compensation Y.

When the reset integration bit in the parameter table is turned ON, the PI compensation is calculated using an I compensation value of 0.

The input value to the PI instruction can be an integer or a real number. Double-length integers cannot be used.

The structure of the parameter table is different for integers and real numbers.

Important If using an integer, set an integral multiple of 1 ms for the scan time.



Previous I remainder (remainder of $K_i \times T_s / T_i$)
 The previous I remainder (IREM') is used only with the integer PI instruction.

The previous I compensation (Y_i') is updated or not, based on the value of P + I compensation.

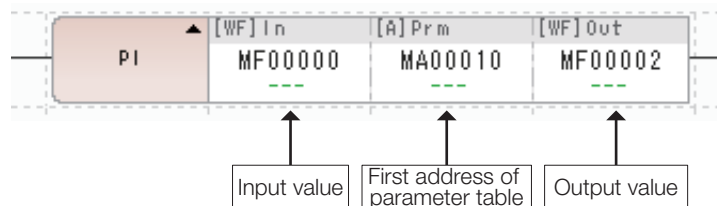
- If Y_i' is inside the range of the PI upper and lower limits (UL, LL) → Y_i' is updated. ($Y_i' = I$ compensation)
- If Y_i' is outside the range of the PI upper and lower limits (UL, LL)
 - If the P compensation and I compensation have the same sign (divergence) → Y_i' is not updated.
 - If the P compensation and I compensation do not have the same sign (convergence) → Y_i' is updated. ($Y_i' = I$ compensation)

The operation of the PI instruction can be expressed by the following formula, where X(s) is the input value and Y(s) is the output value.

$$\frac{Y(s)}{X(s)} = K_p + K_i \times \frac{1}{T_i \times s}$$

Format

The format of this instruction is shown below.



Icon: **PI**
Key entry: PI

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In (Input value)	x	○	x	x	○	x	x	○	○
Prm (First address of parameter table)	x	x	x	x	x	x	○*	○	○
Out (Output value)	x	○*	x	x	○*	x	x	○	x

* C and # registers cannot be used.

◆ Parameter Table for PI Instruction with Integers

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RLY	Relay I/O	Relay inputs and relay outputs*	IN/OUT
1	W	Kp	P gain	Gain for the P compensation (a gain of 1 is equivalent to 100)	IN
2	W	Ki	I gain	Gain for the input to the integration circuit (a gain of 1 is equivalent to 100)	IN
3	W	Ti	Integral time	Integral time (ms)	IN
4	W	IUL	Upper integration limit	Upper limit for the I compensation	IN
5	W	ILL	Lower integration limit	Lower limit for the I compensation	IN
6	W	UL	PI upper limit	Upper limit for the P + I compensation	IN
7	W	LL	PI lower limit	Lower limit for the P + I compensation	IN
8	W	DB	PI output dead zone	Dead zone width for the P + I compensation	IN
9	W	Y	PI output	PI compensation output (output to Out)	OUT
10	W	Yi	I compensation	I compensation storage	OUT
11	W	I REM	I remainder	I remainder storage	OUT

* The relay inputs and outputs are assigned as given below.

Bit	Symbol	Name	Specification	I/O
0	IRST	Reset integration bit	Turn ON the input to reset the integration operation.	IN
1 to 7	–	(Reserved.)	Spare input relays	IN
8 to F	–	(Reserved.)	Spare output relays	OUT

◆ Parameter Table for PI Instruction with Real Numbers

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RLY	Relay I/O	Relay inputs and relay outputs*	IN/OUT
1	W	-	(Reserved.)	Spare register	-
2	F	Kp	P gain	Gain for the P compensation (a gain of 1 is equivalent to 1.0)	IN
4	F	Ki	I gain	Gain for the input to the integration circuit (a gain of 1 is equivalent to 1.0)	IN
6	F	Ti	Integral time	Integral time (s)	IN
8	F	IUL	Upper integration limit	Upper limit for the I compensation	IN
10	F	ILL	Lower integration limit	Lower limit for the I compensation	IN
12	F	UL	PI upper limit	Upper limit for the P + I compensation	IN
14	F	LL	PI lower limit	Lower limit for the P + I compensation	IN
16	F	DB	PI output dead zone	Dead zone width for the P + I compensation	IN
18	F	Y	PI output	PI compensation output (output to Out)	OUT
20	F	Yi	I compensation	I compensation storage	OUT

* The relay input and output assignments are the same as for integers.

◆ Internal Operation of the Instruction

The deviation X input is used to calculate the output value (PI compensation) as shown below. In the formula shown below, Yi' is the previous I compensation of Yi and Ts is the scan time set value.

Information When IRST (reset integration) is turned ON, the PI compensation is calculated with the I compensation set to 0.

P compensation = Upper/lower limit (UL or LL) of (Kp × X)

Yi (I compensation) = Upper/lower limit (IUL or ILL) of { (Ki × X + IREM) / $\frac{T_i}{T_s}$ + Yi' }

Y (PI compensation) = P compensation + Upper/lower limit (UL or LL) and Dead zone A (Width DB) of the I compensation

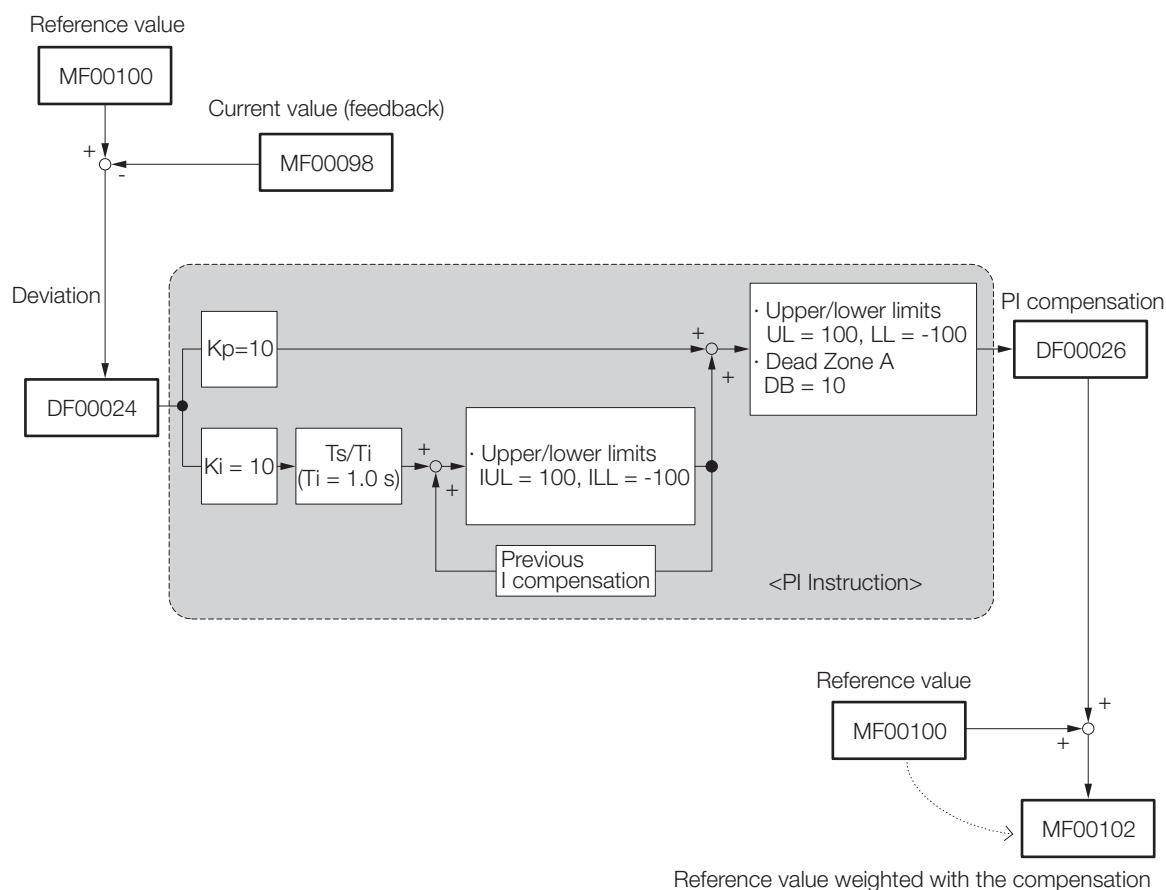
Programming Example

This programming example calculates the reference value in MF00102 weighted with the PI compensation.

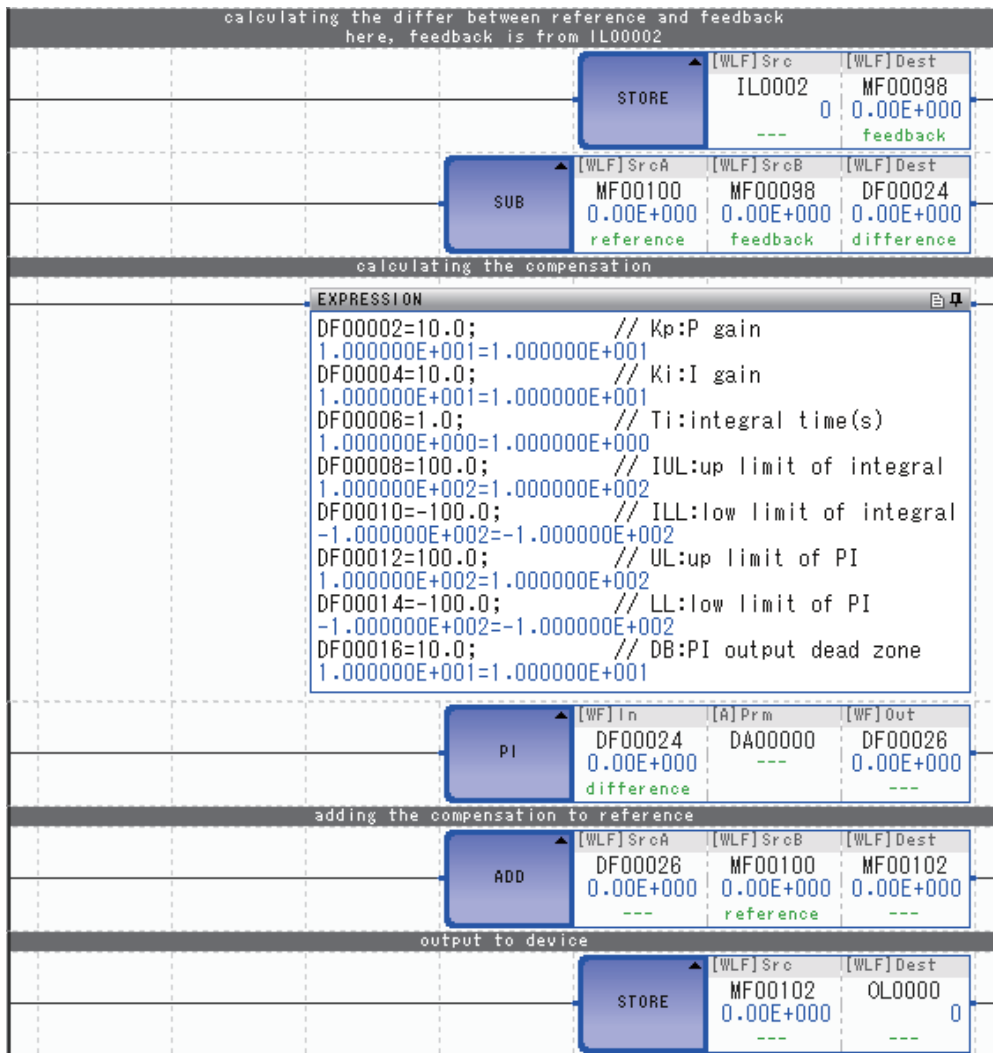
The deviation in DF00024 is obtained from the reference value in MF00100 and the current value in MF00098 and it is used as the input to the PI instruction.

The reference value to output is obtained by adding the original reference value in MF00100 to the PI compensation output in DF00026.

The following block diagram illustrates the programming example.



The programming example is shown below.




Note: The OL00000 (reference value) and IL00002 (feedback value) registers are assigned to external devices.

4.8.5 PD Control (PD)

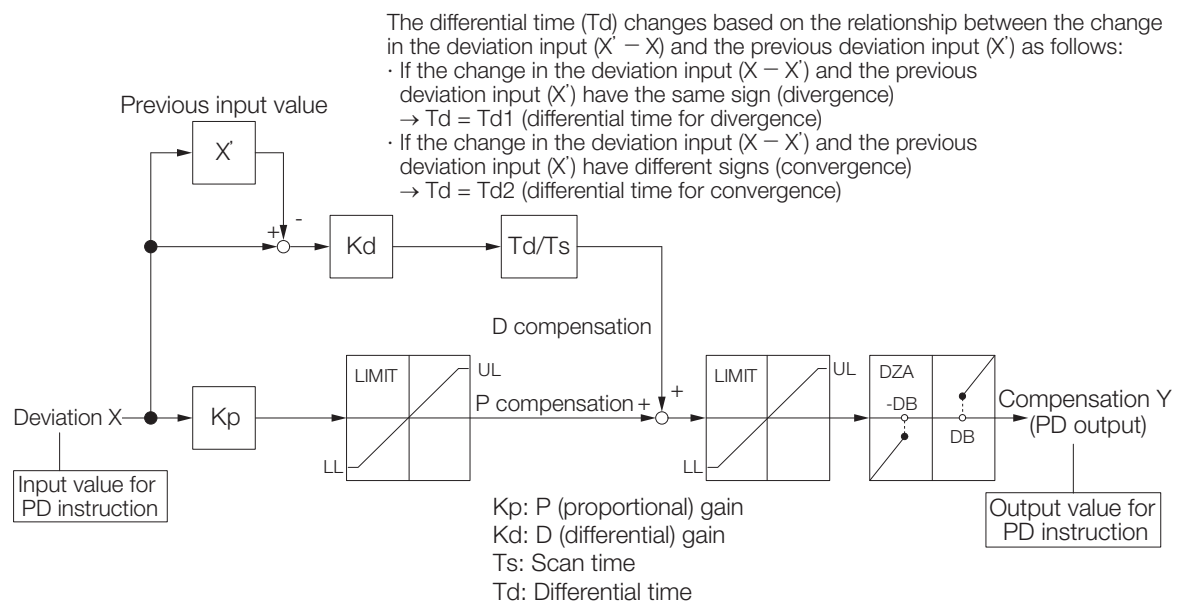
When deviation X is input, P and D operations and a range operation are performed based on predefined parameters in a parameter table, and the result is output as compensation Y.

The input value to the PD instruction can be an integer or a real number. Double-length integers cannot be used.

The structure of the parameter table is different for integers and real numbers.



If using an integer, set an integral multiple of 1 ms for the scan time.

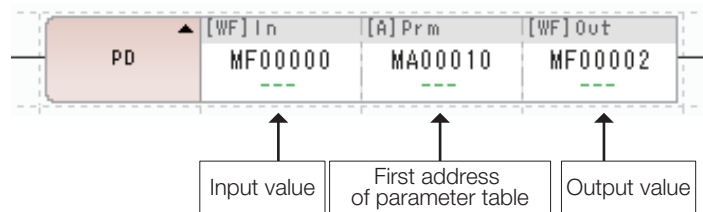


The operation of the PD instruction can be expressed by the following formula, where X(s) is the input value and Y(s) is the output value.

$$\frac{Y(s)}{X(s)} = Kp + Kd \times Td \times S$$

Format

The format of this instruction is shown below.



Icon: PD
 Key entry: PD

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In (Input value)	x	○	x	x	○	x	x	○	○
Prm (First address of parameter table)	x	x	x	x	x	x	○*	○	○
Out (Output value)	x	○*	x	x	○*	x	x	○	x

* C and # registers cannot be used.

◆ Parameter Table for PD Instruction with Integers

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RLY	Relay I/O	Relay inputs and relay outputs*	IN/OUT
1	W	Kp	P gain	Gain for the P compensation (a gain of 1 is equivalent to 100)	IN
2	W	Kd	D gain	Gain for the input to the differential circuit (a gain of 1 is equivalent to 100)	IN
3	W	Td1	Differential time for divergence	Differential time used when the input diverges (ms)	IN
4	W	Td2	Differential time for convergence	Differential time used when the input converges (ms)	IN
5	W	UL	PD upper limit	Upper limit for the P + D compensation	IN
6	W	LL	PD lower limit	Lower limit for the P + D compensation	IN
7	W	DB	PD output dead zone	Dead zone width for the P + D compensation	IN
8	W	Y	PD output	PD compensation output (output to Out)	OUT
9	W	X	Input value storage	Storage of current input value	OUT

* The relay inputs and outputs are assigned as given below.

Bit	Symbol	Name	Specification	I/O
0 to 7	–	(Reserved.)	Spare input relays	IN
8 to F	–	(Reserved.)	Spare output relays	OUT

◆ Parameter Table for PD Instruction with Real Numbers

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RLY	Relay I/O	Relay inputs and relay outputs*	IN/OUT
1	W	–	(Reserved.)	Spare register	–
2	F	Kp	P gain	Gain for the P compensation (a gain of 1 is equivalent to 1.0)	IN
4	F	Kd	D gain	Gain for the input to the differential circuit (a gain of 1 is equivalent to 1.0)	IN
6	F	Td1	Differential time for divergence	Differential time used when the input diverges (s)	IN
8	F	Td2	Differential time for convergence	Differential time used when the input converges (s)	IN
10	F	UL	PD upper limit	Upper limit for the P + D compensation	IN
12	F	LL	PD lower limit	Lower limit for the P + D compensation	IN
14	F	DB	PD output dead zone	Dead zone width for the P + D compensation	IN
16	F	Y	PD output	PD compensation output (output to Out)	OUT
18	F	X	Input value storage	Storage of current input value	OUT

* The relay input and output assignments are the same as for integers.

◆ Internal Operation of the Instruction

The deviation X input is used to calculate the PD compensation output as shown below.

In the formula shown below, X' is the previous input value of X, Ts is the scan time set value, and Td is the differential time.

The differential time (Td) is Td1 when X – X' and X' have the same sign, and Td2 when X – X' and X' have different signs.

P compensation = Upper/lower limit (UL or LL) of (Kp × X)

D compensation = Kd × (X – X') × Upper/lower limit (IUL or ILL) of $\frac{Td}{Ts}$

PD compensation = Upper/lower limit (UL or LL) of (P compensation + D compensation) and Dead zone A (Width DB)

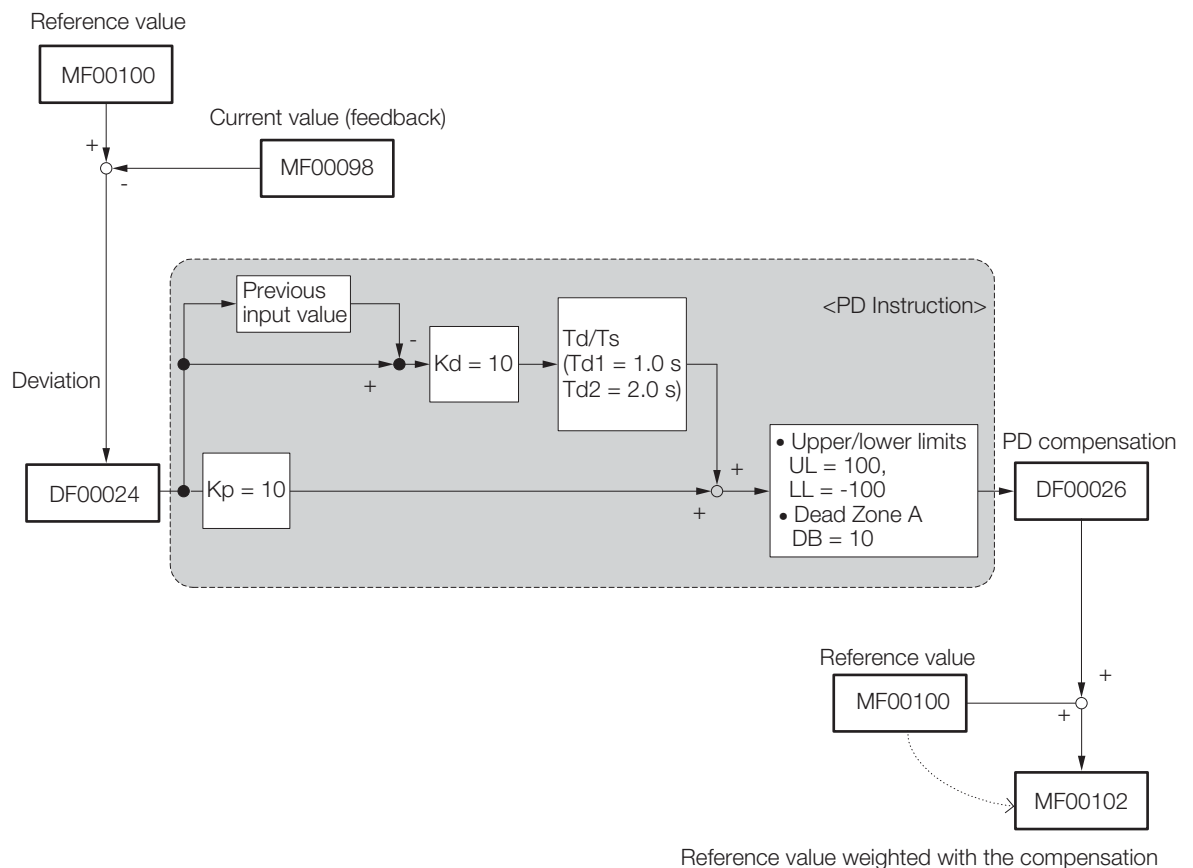
Programming Example

This programming example calculates the reference value in MF00102 weighted with the PD compensation.

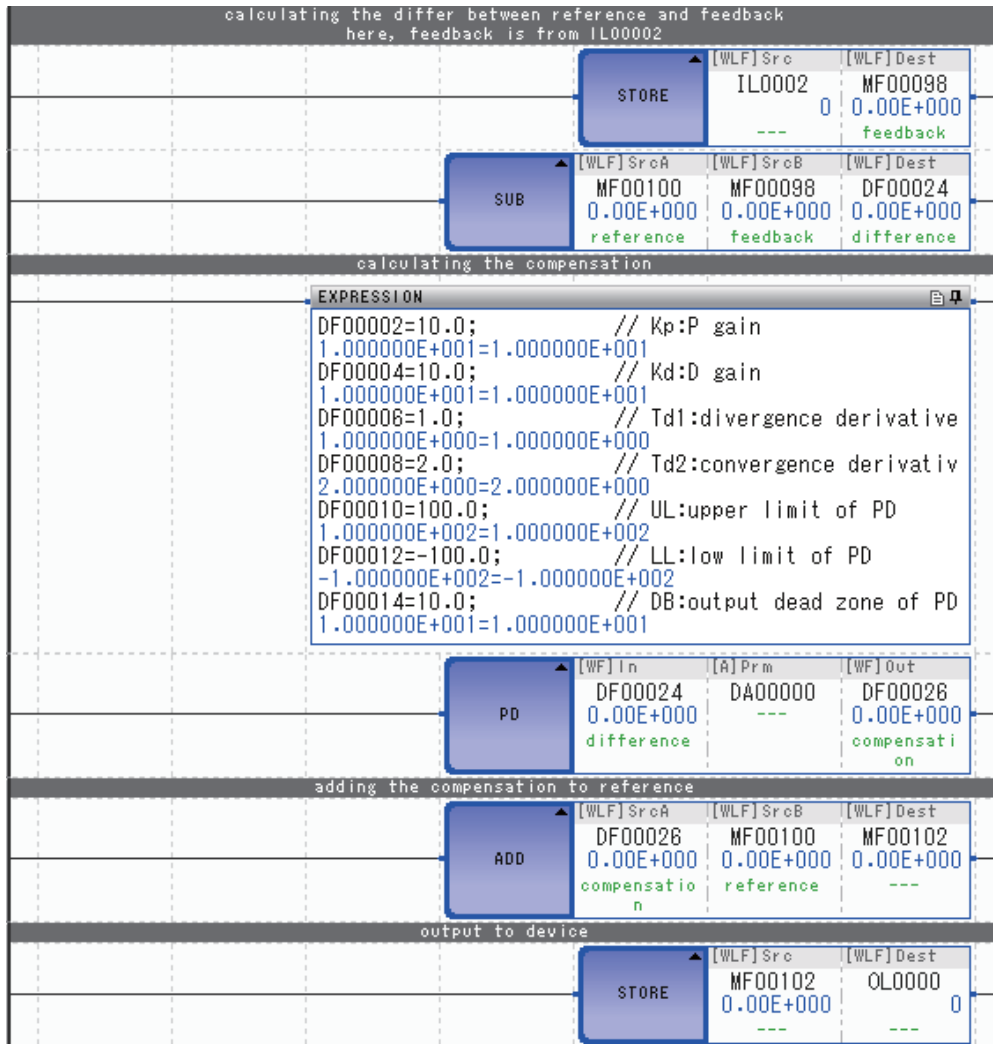
The deviation in DF00024 is obtained from the reference value in MF00100 and the current value in MF00098 and it is used as the input to the PD instruction.

The reference value to output is obtained by adding the original reference value in MF00100 to the PD compensation output in DF000026.

The following block diagram illustrates the programming example.



The programming example is shown below.



Note: The OL00000 (reference value) and IL00002 (feedback value) registers are assigned to external devices.

Additional Information

◆ Transfer Functions

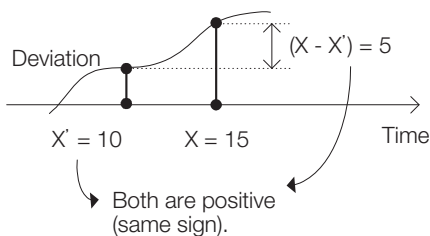
The transfer function of the P and D operations can be expressed by the formula shown below. X(s) is the input value and Y(s) is the output value.

$$\frac{Y(s)}{X(s)} = K_p + K_d \times T_d \times S$$

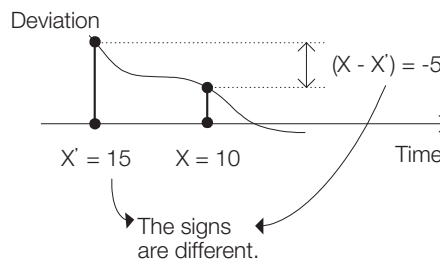
◆ Relation between Current Deviation X and Previous Deviation X' on the Divergence and Convergence Sides

The following figure shows the relation between the current deviation X and previous deviation X' on the divergence and convergence sides.

<Example of a Diverging Deviation>



<Example of a Converging Deviation>



4.8.6 PID Control (PID)

When deviation X is input, P, I, and D operations and a range operation are performed based on predefined parameters in a parameter table, and the result is output as compensation Y .

When the reset integration bit in the parameter table is turned ON, the PI compensation is calculated using an I compensation value of 0.

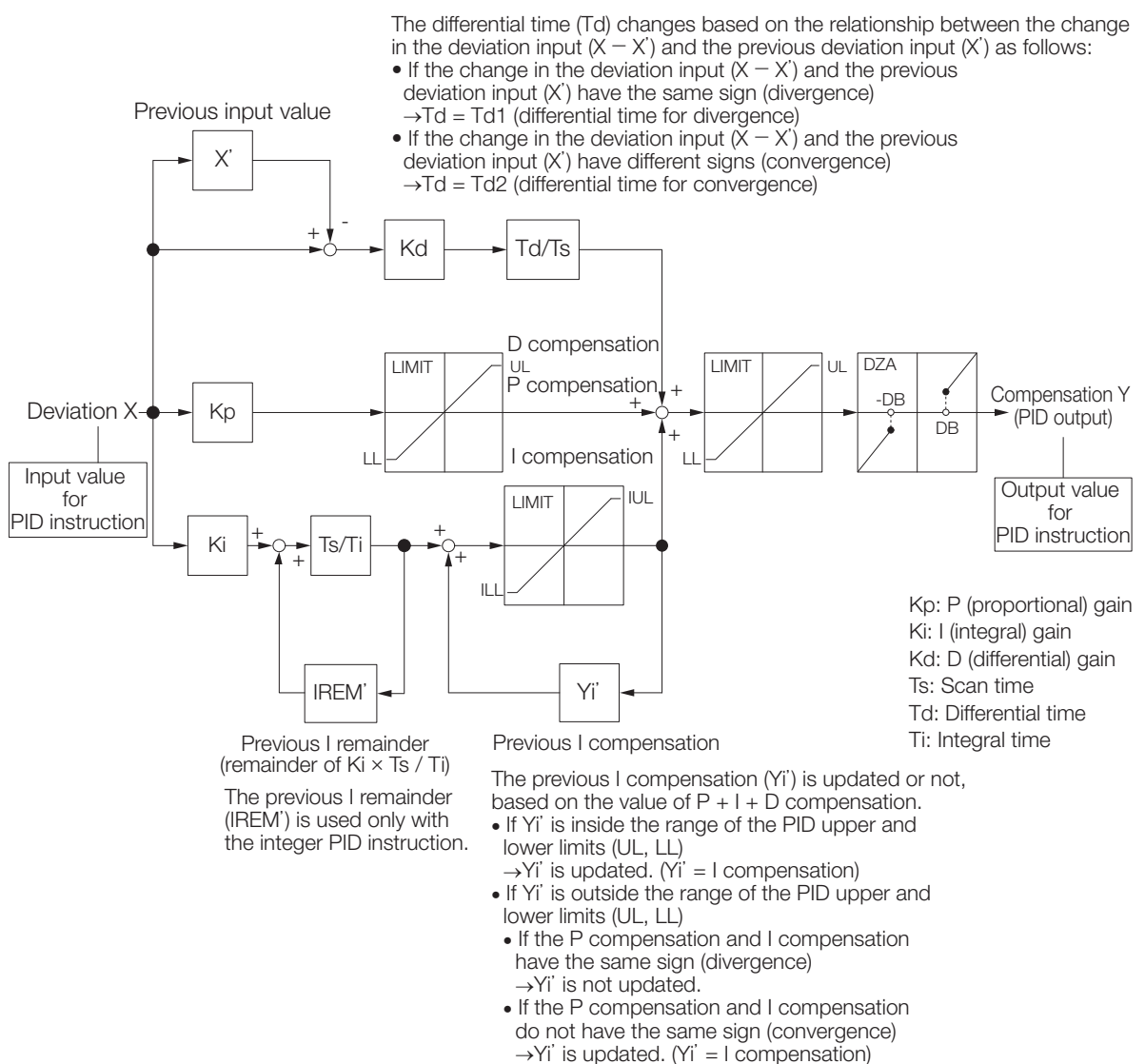
The input value to the PID instruction can be an integer or a real number. Double-length integers cannot be used.

The structure of the parameter table is different for integers and real numbers.



Important

If using an integer, set an integral multiple of 1 ms for the scan time.

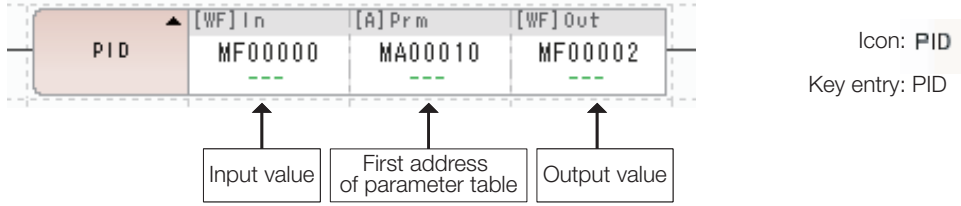


The operation of the PID instruction can be expressed by the following formula, where $X(s)$ is the input value and $Y(s)$ is the output value.

$$\frac{Y(s)}{X(s)} = K_p + K_i \times \frac{1}{T_i \times s} + K_d \times T_d \times s$$

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In (Input value)	x	○	x	x	○	x	x	○	○
Prm (First address of parameter table)	x	x	x	x	x	x	○*	○	○
Out (Output value)	x	○*	x	x	○*	x	x	○	x

* C and # registers cannot be used.

◆ Parameter Table for PID Instruction with Integers

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RLY	Relay I/O	Relay inputs and relay outputs*	IN/OUT
1	W	Kp	P gain	Gain for the P compensation (a gain of 1 is equivalent to 100)	IN
2	W	Ki	I gain	Gain for the input to the integration circuit (a gain of 1 is equivalent to 100)	IN
3	W	Kd	D gain	Gain for the input to the differential circuit (a gain of 1 is equivalent to 100)	IN
4	W	Ti	Integral time	Integral time (ms)	IN
5	W	Td1	Differential time for divergence	Differential time used when the input diverges (ms)	IN
6	W	Td2	Differential time for convergence	Differential time used when the input converges (ms)	IN
7	W	IUL	Upper integration limit	Upper limit for the I compensation	IN
8	W	ILL	Lower integration limit	Lower limit for the I compensation	IN
9	W	UL	PID upper limit	Upper limit for the P + I compensation	IN
10	W	LL	PID lower limit	Lower limit for the P + I compensation	IN
11	W	DB	PID output dead zone	Dead zone width for the P + I compensation	IN
12	W	Y	PID output	PI compensation output (output to Out)	OUT
13	W	Yi	I compensation	I compensation storage	OUT
14	W	IREM	I remainder	I remainder storage	OUT
15	W	X	Input value storage	Storage of current input value	OUT

* The relay inputs and outputs are assigned as given below.

Bit	Symbol	Name	Specification	I/O
0	IRST	Reset integration bit	Turn ON the input to reset the integration operation.	IN
1 to 7	-	(Reserved.)	Spare input relays	IN
8 to F	-	(Reserved.)	Spare output relays	OUT

◆ Parameter Table for PID Instruction with Real Numbers

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RLY	Relay I/O	Relay inputs and relay outputs*	IN/OUT
1	W	–	(Reserved.)	Spare register	IN
2	F	Kp	P gain	Gain for the P compensation (a gain of 1 is equivalent to 1.0)	IN
4	F	Ki	I gain	Gain for the input to the integration circuit (a gain of 1 is equivalent to 1.0)	IN
6	F	Kd	D gain	Gain for the input to the differential circuit (a gain of 1 is equivalent to 1.0)	IN
8	F	Ti	Integral time	Integral time (s)	IN
10	F	Td1	Differential time for divergence	Differential time used when the input diverges (s)	IN
12	F	Td2	Differential time for convergence	Differential time used when the input converges (s)	IN
14	F	IUL	Upper integration limit	Upper limit for the I compensation	IN
16	F	ILL	Lower integration limit	Lower limit for the I compensation	IN
18	F	UL	PID upper limit	Upper limit for the P + I + D compensation	IN
20	F	LL	PID lower limit	Lower limit for the P + I + D compensation	IN
22	F	DB	PID output dead zone	Dead zone width for the P + I + D compensation	IN
24	F	Y	PID output	PID compensation output (output to Out)	OUT
26	F	Yi	I compensation	I compensation storage	OUT
28	F	X	Input value storage	Storage of current input value	OUT

* The relay input and output assignments are the same as for integers.

◆ Internal Operation of the Instruction

The deviation X input is used to calculate the PID compensation output as shown below.

In the formula shown below, X' is the previous input value of X, Y' is the previous I compensation, Ts is the scan time set value, and Td is the differential time.

The differential time (Td) is Td1 when X – X' and X' have the same sign, and Td2 when X – X' and X' have different signs.

Information When IRST (reset integration) is turned ON, the PID compensation is calculated with the I compensation set to 0.

P compensation = Upper/lower limit (UL or LL) of (Kp × X)

Yi (I compensation) = Upper/lower limit (IUL or ILL) of $\{ (Ki \times X + IREM) / \frac{T_i}{T_s} + Y_i' \}$

D compensation = Kd × (X – X') × Upper/lower limit (IUL or ILL) of $\frac{T_d}{T_s}$

Y (PID compensation) = Upper/lower limits (UL or LL) of P + I + D compensation values and Dead zone A (Width DB)

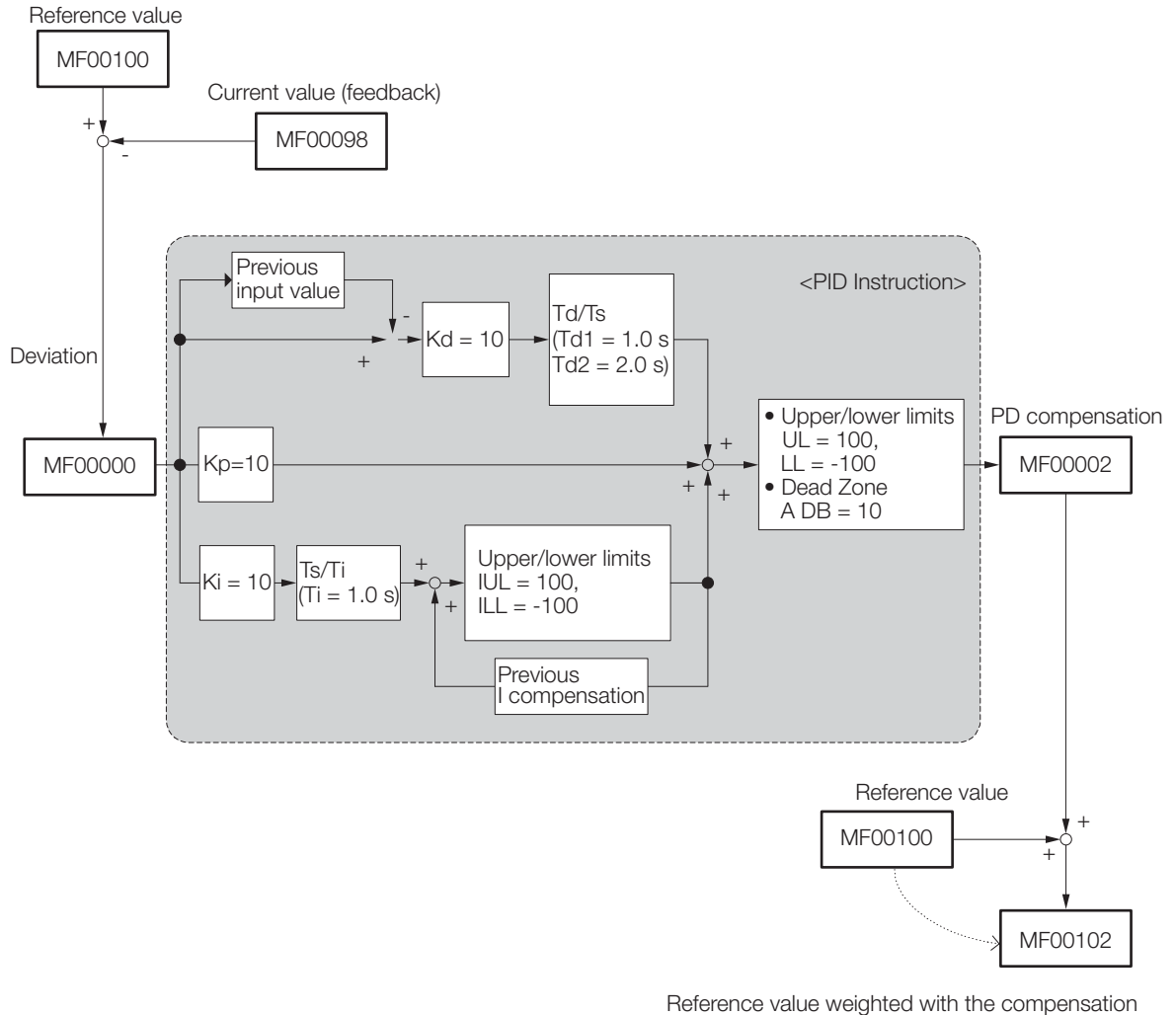
Programming Example

This programming example calculates the reference value in MF00102 weighted with the PID compensation.

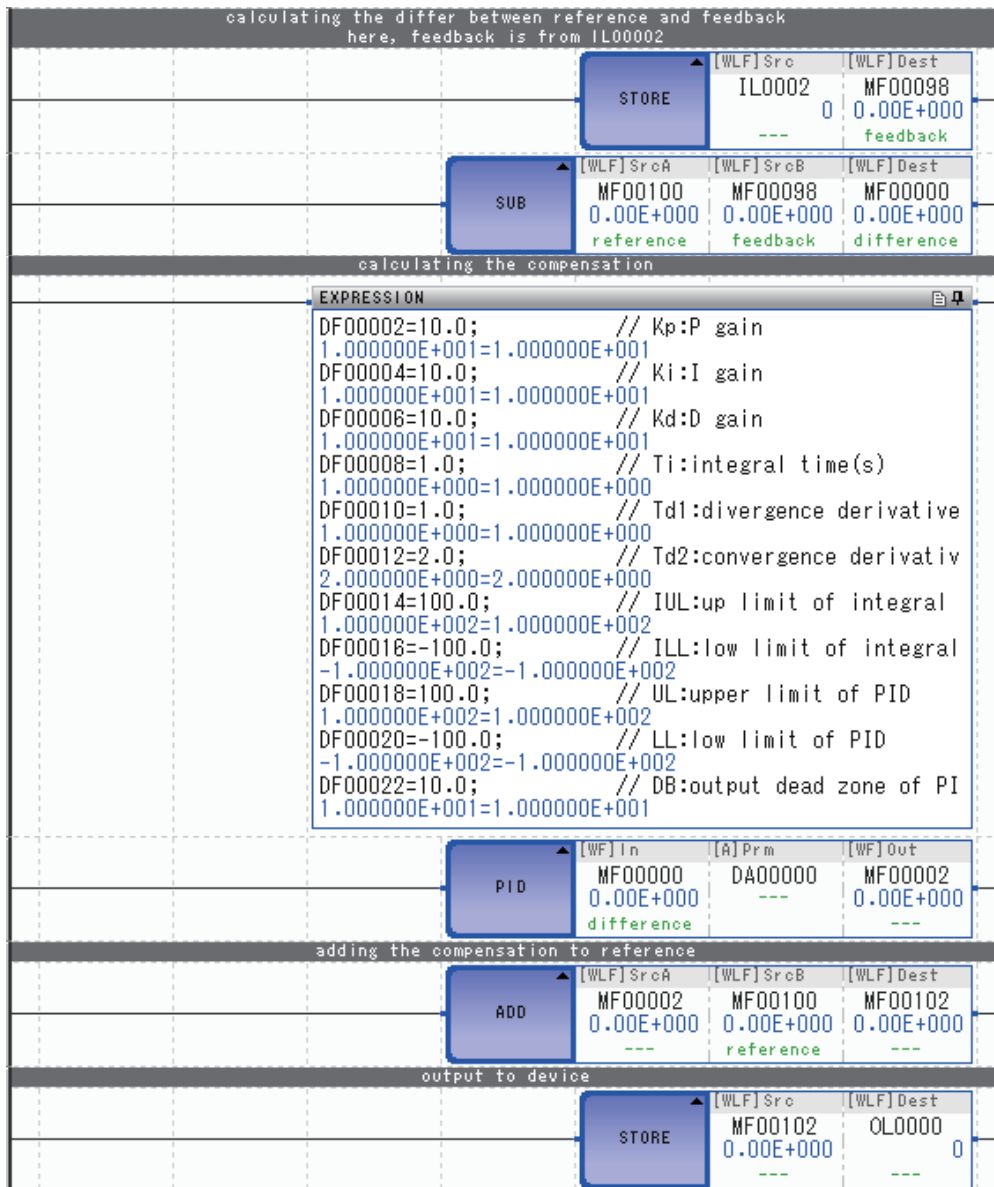
The deviation in MF00000 is obtained from the reference value in MF00100 and the current value in MF00098 and it is used as the input to the PID instruction.

The reference value to output is obtained by adding the original reference value in MF00100 to the PID compensation output in MF00002.

The following block diagram illustrates the programming example.



The programming example is shown below.




Note: The OL00000 (reference value) and IL00002 (feedback value) registers are assigned to external devices.

4.8.7 First-order Lag (LAG)

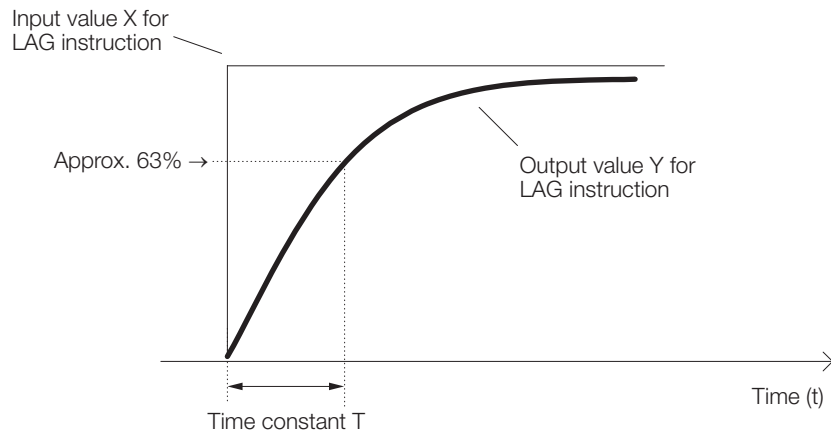
The first-order lag is calculated according to predefined parameters in a parameter table.

The input value to the LAG instruction can be an integer or a real number. Double-length integers cannot be used.

The structure of the parameter table is different for integers and real numbers.



Important If using an integer, set an integral multiple of 1 ms for the scan time.



The LAG operation in the figure shown above can be expressed by the formula shown below.

$$\frac{Y(s)}{X(s)} = \frac{1}{1 + T \times s}$$

Therefore,

$$T \times \frac{dY}{dt} + Y = X$$

The following operation is performed internally by the LAG instruction, where $dt = Ts$ and $dY = Y - Y'$.

In the formula shown below, Y' is the previous output value, Ts is the scan time set value, and REM is the remainder.

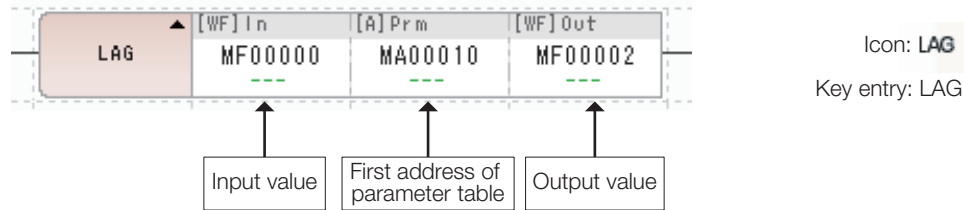
The unit for Ts is the same as the unit for T .

$$Y = \frac{T \times Y' + Ts \times X + REM}{T + Ts}$$

Information When IRST (LAG reset) is ON, Y outputs 0 and REM outputs 0.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In (Input value)	×	○	×	×	○	×	×	○	○
Prm (First address of parameter table)	×	×	×	×	×	×	○*	○	○
Out (Output value)	×	○*	×	×	○*	×	×	○	×

* C and # registers cannot be used.

◆ Parameter Table for LAG Instruction with Integers

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RLY	Relay I/O	Relay inputs and relay outputs*	IN/OUT
1	W	T	First order lag time constant	First order lag time constant (ms)	IN
2	W	Y	LAG output	LAG output (output to Out)	OUT
3	W	REM	Remainder	Remainder storage	OUT

* The relay inputs and outputs are assigned as given below.

Bit	Symbol	Name	Specification	I/O
0	IRST	LAG reset bit	Turn ON this input to reset the LAG operation.	IN
1 to 7	–	(Reserved.)	Spare input relays	IN
8 to F	–	(Reserved.)	Spare output relays	OUT

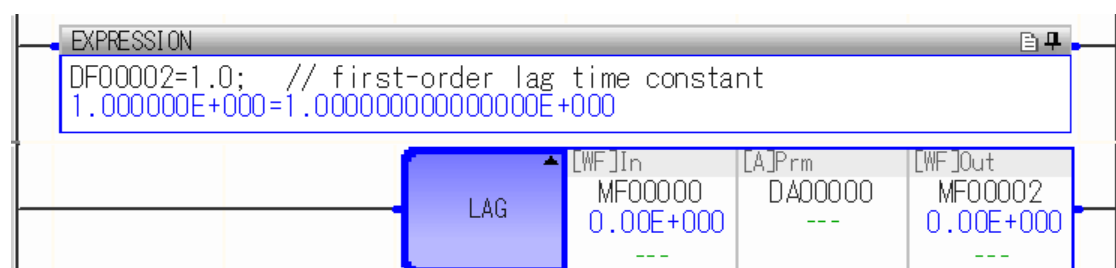
◆ Parameter Table for LAG Instruction with Real Numbers

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RLY	Relay I/O	Relay inputs and relay outputs*	IN/OUT
1	W	–	(Reserved.)	Spare register	–
2	F	T	First-order lag time constant	First-order lag time constant (s)	IN
4	F	Y	LAG output	LAG output (output to Out)	OUT

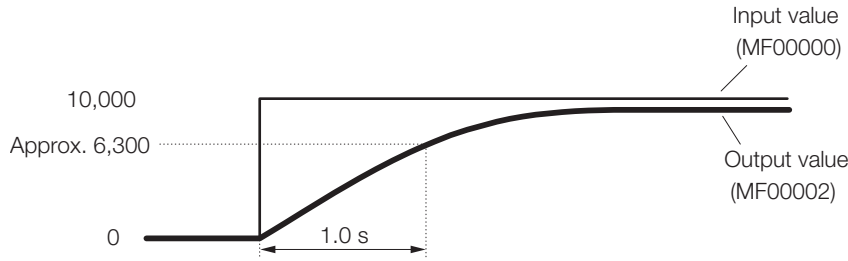
* The relay input and output assignments are the same as for integers.

Programming Example

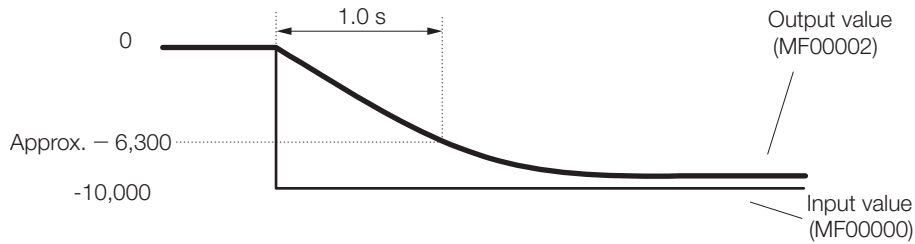
In the following programming example, the LAG instruction is executed where MF00000 is the input value in the parameter table, MF00002 is the output value, and the first-order lag time constant is set to 1.0.



MF000002 changes as shown below when MF00000 changes from 0 to 10,000.



MF000002 changes as shown below when MF00000 changes from 0 to -10,000.

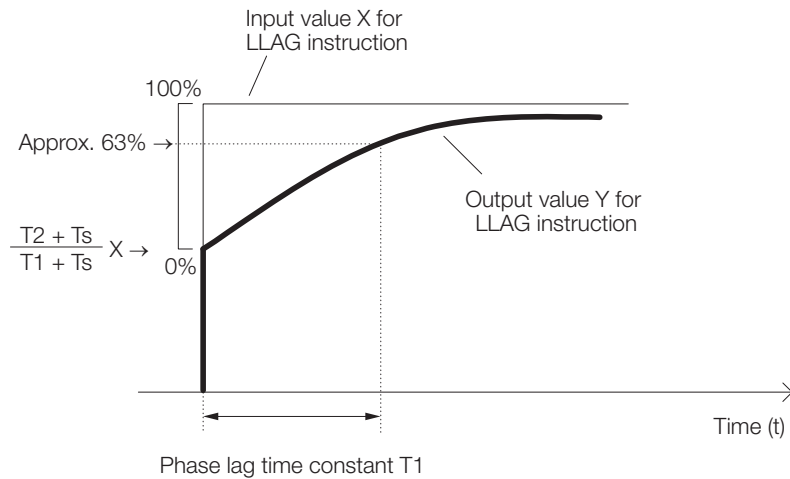


4.8.8 Phase Lead Lag (LLAG)

The phase lead and lag are calculated according to predefined parameters in a parameter table. The input value to the LLAG instruction can be an integer or real number. Double-length integers cannot be used.

The structure of the parameter table is different for integers and real numbers.

Important If using an integer, set an integral multiple of 1 ms for the scan time.



The LLAG operation in the figure shown above can be expressed by the formula shown below.

$$\frac{Y(s)}{X(s)} = \frac{1 + T2 \times s}{1 + T1 \times s}$$

Therefore,

$$T1 \times \frac{dY}{dt} + Y = T2 \times \frac{dX}{dt} + X$$

The following operation is performed internally by the LLAG instruction, where $dt = T_s$, $dY = Y - Y'$, and $dX = X - X'$.

In the formula shown below, Y' is the previous output value, X' is the previous input value, T_s is the scan time set value, and REM is the remainder.

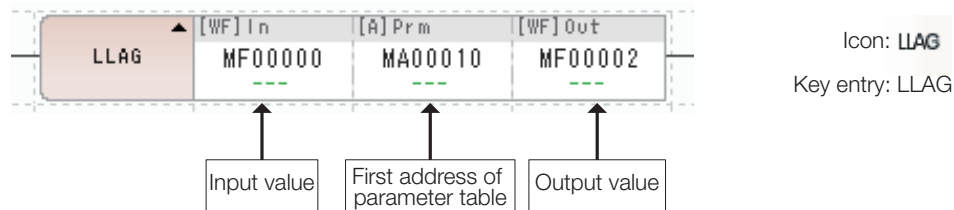
The unit for T_s is the same as the unit for T_1 .

$$Y = \frac{T_1 \times Y' + (T_2 + T_s) \times X - T_2 \times X' + \text{REM}}{T_1 + T_s}$$

Information When IRST (LLAG reset) is ON, Y outputs 0, REM outputs 0, and X outputs 0.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In (Input value)	×	○	×	×	○	×	×	○	○
Prm (First address of parameter table)	×	×	×	×	×	×	○*	○	○
Out (Output value)	×	○*	×	×	○*	×	×	○	×

* C and # registers cannot be used.

◆ Parameter Table for LLAG Instruction with Integers

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RLY	Relay I/O	Relay inputs and relay outputs*	IN/OUT
1	W	T2	Phase lead time constant	Phase lead time constant (ms)	IN
2	W	T1	Phase lag time constant	Phase lag time constant (ms)	IN
3	W	Y	LLAG output	LLAG output (output to Out)	OUT
4	W	REM	Remainder	Remainder storage	OUT
5	W	X	Input value storage	Input value storage	OUT

* The relay inputs and outputs are assigned as given below.

Bit	Symbol	Name	Specification	I/O
0	IRST	LLAG reset bit	Turn ON this input to reset the LLAG operation.	IN
1 to 7	–	(Reserved.)	Spare input relays	IN
8 to F	–	(Reserved.)	Spare output relays	OUT

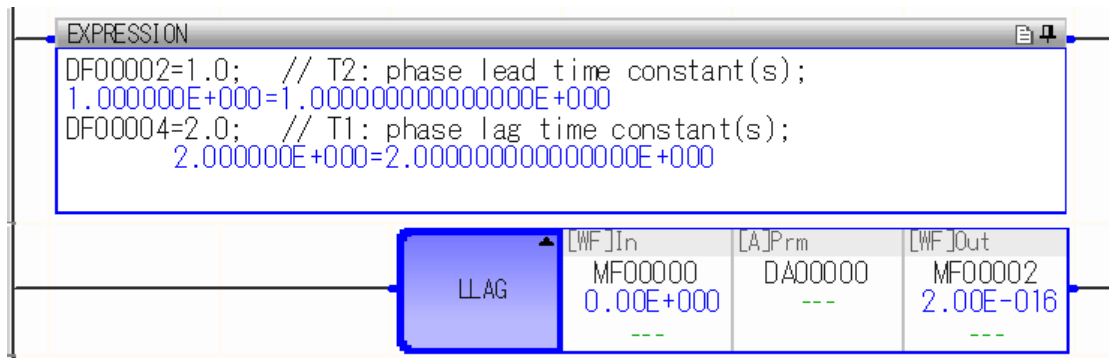
◆ Parameter Table for LAG Instruction with Real Numbers

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RLY	Relay I/O	Relay inputs and relay outputs*	IN/OUT
1	W	-	(Reserved.)	Spare register	-
2	F	T2	Phase lead time constant	Phase lead time constant (s)	IN
4	F	T1	Phase lag time constant	Phase lag time constant (s)	IN
6	F	Y	LLAG output	LLAG output (output to Out)	OUT
8	F	X	Input value storage	Input value storage	OUT

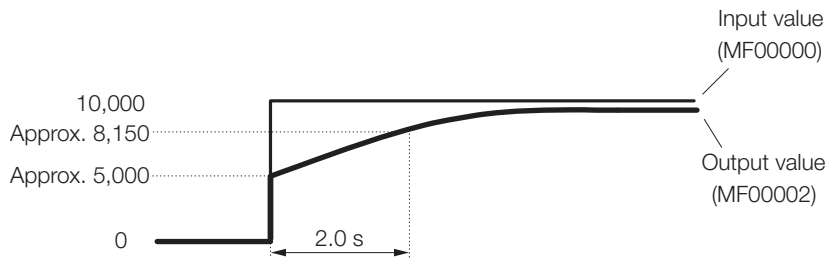
* The relay input and output assignments are the same as for integers.

Programming Example

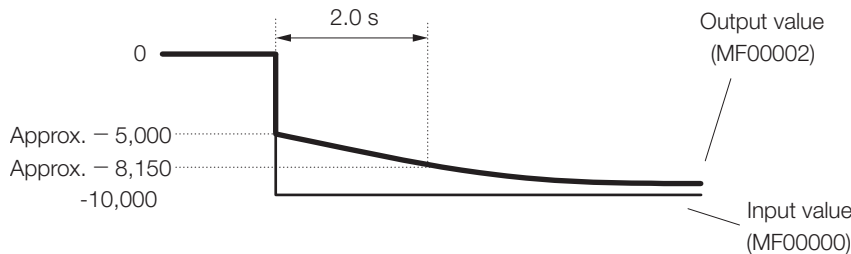
In the following programming example, the LLAG instruction is executed where MF00000 is the input value in the parameter table, MF00002 is the output value, the phase lead time constant is set to 1.0 seconds and the phase lag time constant is set to 2.0 seconds.



MF000002 changes as shown below when MF00000 changes from 0 to 10,000.



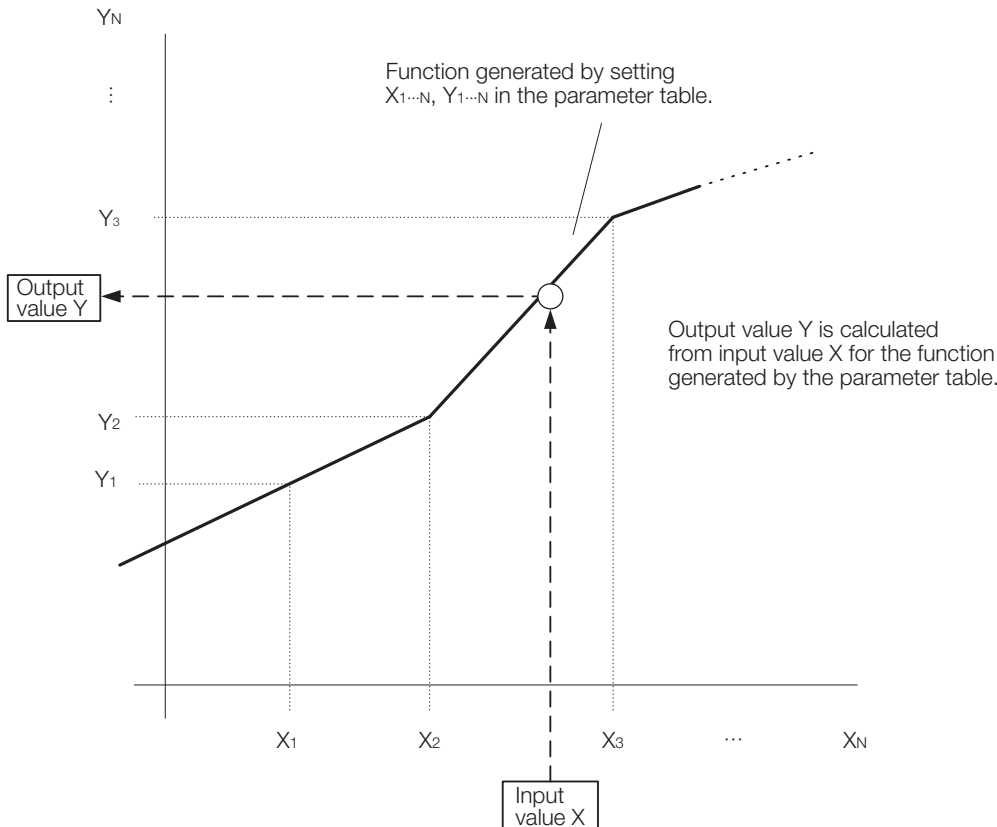
MF000002 changes as shown below when MF00000 changes from 0 to -10,000.



4.8.9 Function Generator (FGN)

A function is generated based on the parameters specified in the parameter table, and it is used to calculate output value Y based on the value of input X.

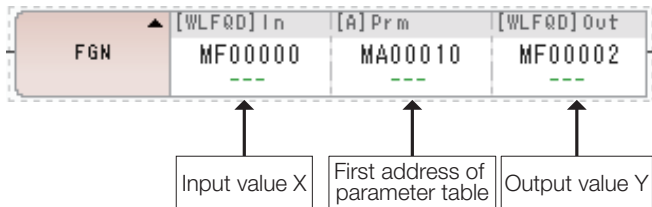
The FGN instruction will be for integers, double-length integers, real numbers, quadruple-length integers, or double-precision real numbers, depending on the data type of input value X. The structure of the parameter table also changes accordingly.



Information Create the parameter table so that $X_1 < X_2 < \dots < X_N$.

Format

The format of this instruction is shown below.



Icon: **FGN**
Key entry: FGN

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In (Input value X)	×	○	○	○	○	○	×	○	○
Prm (First address of parameter table)	×	×	×	×	×	×	○	×	×
Out (Output value Y)	×	○*	○*	○*	○*	○*	×	○	×

* C and # registers cannot be used.

◆ **Parameter Table for FGN Instruction with Integers**

If input value X is an integer, the FGN instruction will be for integers.

Create the parameter table as shown below.

Address	Data Type	Symbol	Name
0	W	N	Number of pairs of X and Y
1	W	X_1	Data X_1
2	W	Y_1	Data Y_1
3	W	X_2	Data X_2
4	W	Y_2	Data Y_2
:	:	:	:
$2N-1$	W	X_N	Data X_N
$2N$	W	Y_N	Data Y_N

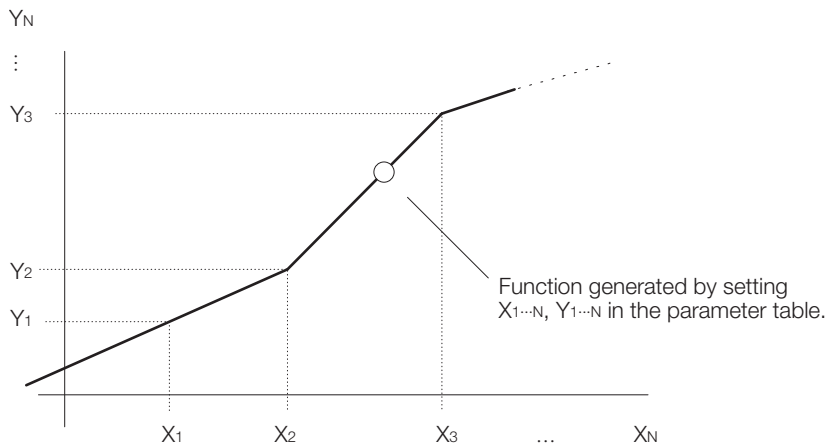
◆ **Parameter Table for FGN Instruction with Double-length Integers or Real Numbers**

If input value X is a double-length integer, the FGN instruction will be for double-length integers.

If input value X is a real number, the FGN instruction will be for real numbers.

Create the parameter table as shown below.

Address	Data Type	Symbol	Name
0	W	N	Number of pairs of X and Y
1	W	-	Reserved.
2	L/F	X_1	Data X_1
4	L/F	Y_1	Data Y_1
6	L/F	X_2	Data X_2
8	L/F	Y_2	Data Y_2
:	:	:	:
$4N-2$	L/F	X_N	Data X_N
$4N$	L/F	Y_N	Data Y_N



◆ Parameter Table for FGN Instruction with Quadruple-length Integers or Double-precision Real Numbers

If input value X is a quadruple-length integer, the FGN instruction will be for quadruple-length integers. If input value X is a double-precision real number, the FGN instruction will be for double-precision real numbers.

Create the parameter table as shown below.

Address	Data Type	Symbol	Name
0	W	N	Number of pairs of X and Y
1	W	-	Reserved.
2	L	-	Reserved.
4	Q/D	X ₁	Data X ₁
8	Q/D	Y ₁	Data Y ₁
12	Q/D	X ₂	Data X ₂
16	Q/D	Y ₂	Data Y ₂
:	:	:	:
8N-4	Q/D	X _N	Data X _N
8N	Q/D	Y _N	Data Y _N

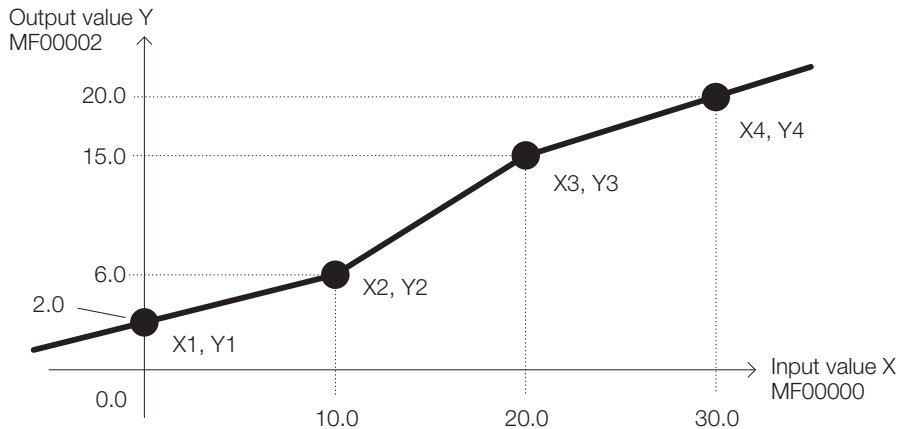
Information Make sure to set the data so that $X_1 < X_2 < \dots < X_N$, regardless of whether the parameter table is for integer data, double-length integer data, real number data, quadruple-length integer data, or double-precision real number data.

Programming Example

In the following programming example, the function is generated using the FGN instruction for real numbers with the parameter table given below.

Number of Pairs	4
X1, Y1	0.0, 2.0
X2, Y2	10.0, 6.0
X3, X4	20.0, 15.0
X4, Y4	30.0, 20.0

The following figure shows the relationship between input value X in MF00000 and output value Y in MF00002.



Additional Information

Output value Y is calculated as shown below.

- If the pair X_n and Y_n where $X_n \leq \text{Input } X \leq X_{n+1}$ exists,

$$\text{Output value } Y = Y_n + \frac{Y_{n+1} - Y_n}{X_{n+1} - X_n} \times (\text{Input value } X - X_n) \quad (1 \leq n \leq N - 1)$$

- If the pair X_n and Y_n where $X_n \leq \text{Input } X \leq X_{n+1}$ does not exist,
If Input value $X < X_1$,

$$\text{Output value } Y = Y_1 + \frac{Y_2 - Y_1}{X_2 - X_1} \times (\text{Input value } X - X_1)$$

If Input value $X > X_N$,

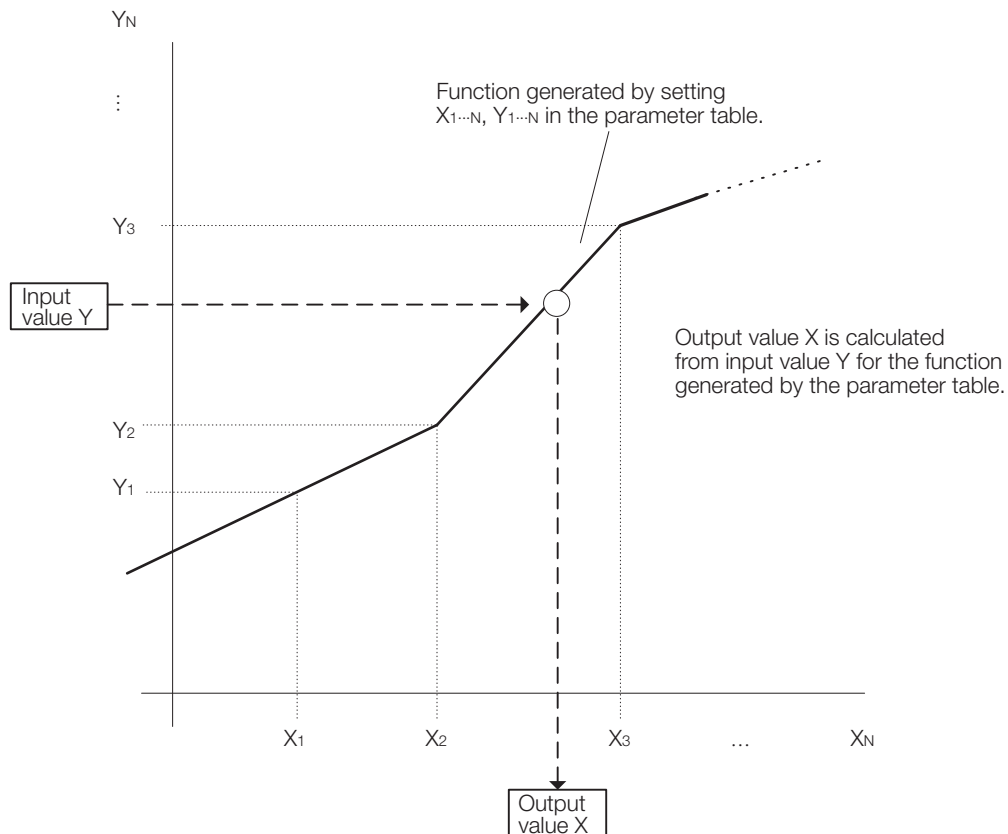
$$\text{Output value } Y = Y_N + \frac{Y_N - Y_{N-1}}{X_N - X_{N-1}} \times (\text{Input value } X - X_N)$$

4.8.10 Inverse Function Generator (IFGN)

A function is generated based on the parameters specified in the parameter table, and it is used to calculate output value X based on the value of input Y.

The IFGN instruction will be for integers, double-length integers, real numbers, quadruple-length integers, or a double-precision real numbers depending on the data type of input value X.

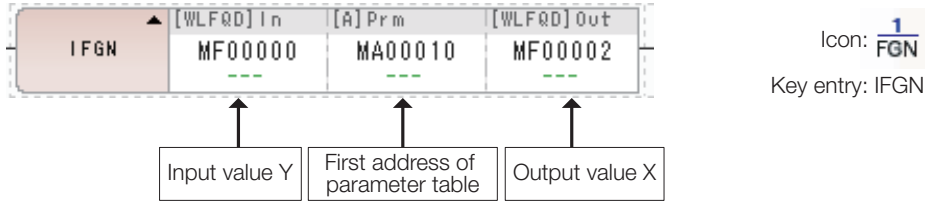
The structure of the parameter table is the same as for the FGN instruction.



Information Set the parameter table so that $Y_1 < Y_2 < \cdots < Y_N$.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In (Input value Y)	x	○	○	○	○	○	x	○	○
Prm (First address of parameter table)	x	x	x	x	x	x	○	x	x
Out (Output value X)	x	○*	○*	○*	○*	○*	x	○	x

* C and # registers cannot be used.

◆ Parameter Table for IFGN Instruction with Integers

If input value X is an integer, the IFGN instruction will be for integers.

Create the parameter table as shown below.

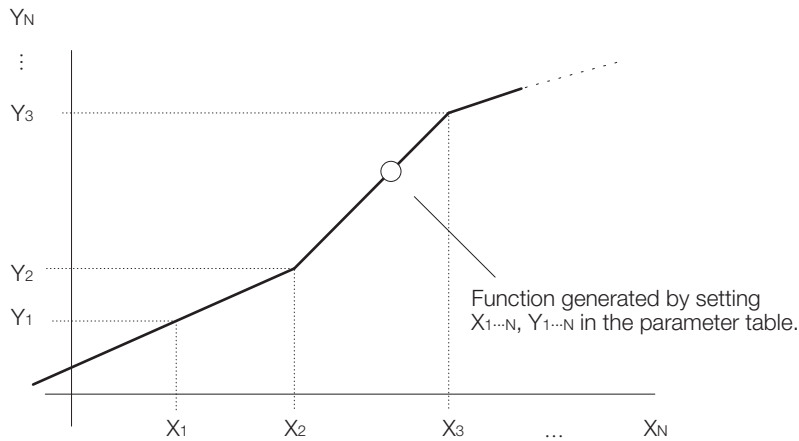
Address	Data Type	Symbol	Name
0	W	N	Number of pairs of X and Y
1	W	X ₁	Data X ₁
2	W	Y ₁	Data Y ₁
3	W	X ₂	Data X ₂
4	W	Y ₂	Data Y ₂
:	:	:	:
2N-1	W	X _N	Data X _N
2N	W	Y _N	Data Y _N

◆ Parameter Table for IFGN Instruction with Double-length Integers or Real Numbers

If input value Y is a double-length integer, the IFGN instruction will be for double-length integers. If input value Y is a real number, the IFGN instruction will be for real numbers.

Create the parameter table as shown below.

Address	Data Type	Symbol	Name
0	W	N	Number of pairs of X and Y
1	W	-	Reserved.
2	L/F	X ₁	Data X ₁
4	L/F	Y ₁	Data Y ₁
6	L/F	X ₂	Data X ₂
8	L/F	Y ₂	Data Y ₂
:	:	:	:
4N-2	L/F	X _N	Data X _N
4N	L/F	Y _N	Data Y _N



◆ Parameter Table for FGN Instructions with Quadruple-length Integers or Double-precision Real Numbers

If input value X is a quadruple-length integer, the FGN instruction will be for quadruple-length integers. If input value X is a double-precision real number, the FGN instruction will be for double-precision real numbers.

Create the parameter table as shown below.

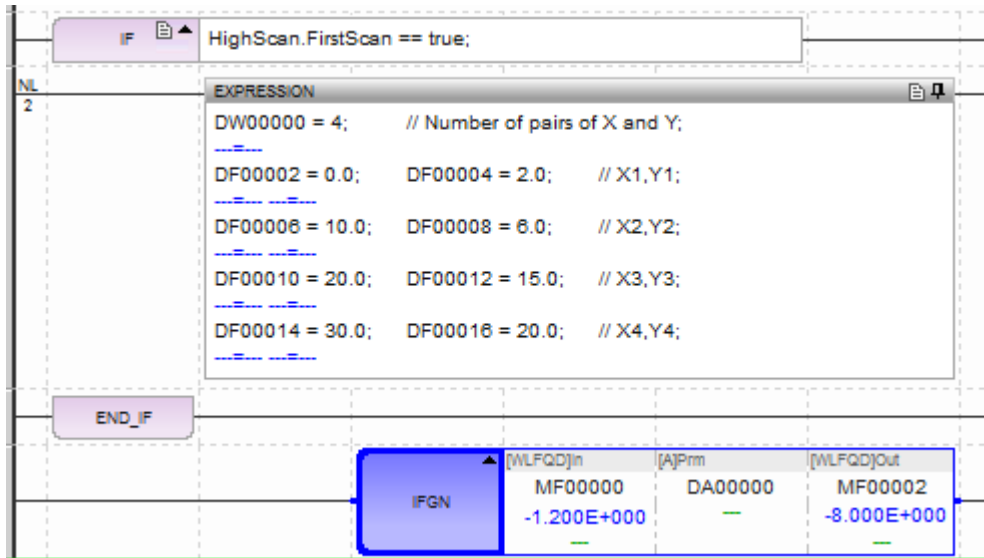
Address	Data Type	Symbol	Name
0	W	N	Number of pairs of X and Y
1	W	–	Reserved.
2	L	–	Reserved.
4	Q/D	X_1	Data X_1
8	Q/D	Y_1	Data Y_1
12	Q/D	X_2	Data X_2
16	Q/D	Y_2	Data Y_2
:	:	:	:
$8N-4$	Q/D	X_N	Data X_N
$8N$	Q/D	Y_N	Data Y_N

Information Make sure to set the data so that $Y_1 < Y_2 < \dots < Y_N$, regardless of whether the parameter table is for integer data, double-length integer data, real number data, quadruple-length integer data, or double-precision real number data.

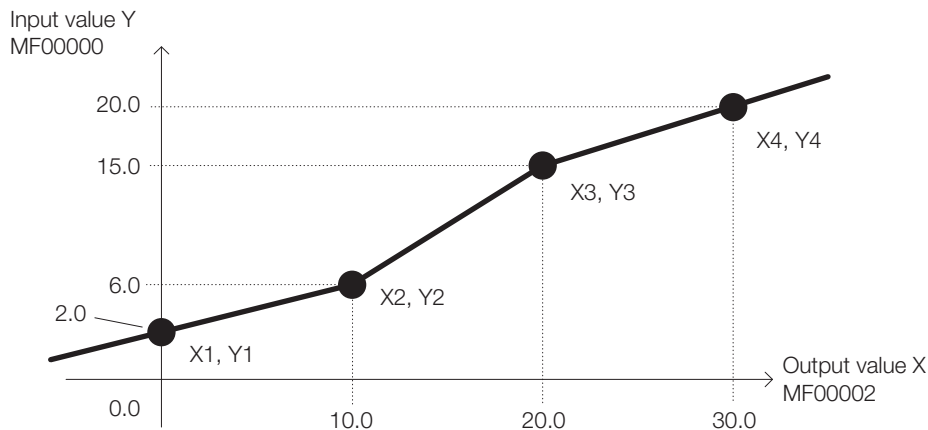
Programming Example

In the following programming example, the function is generated using the IFGN instruction for real numbers with the parameter table given below.

Number of Pairs	4
X1, Y1	0.0, 2.0
X2, Y2	10.0, 6.0
X3, X4	20.0, 15.0
X4, Y4	30.0, 20.0



The following figure shows the relationship between input value Y in MF00000 and output value X in MF00002.



Additional Information

Output value X is calculated as shown below.

- If the pair X_n and Y_n where $Y_n \leq \text{Input } Y \leq Y_{n+1}$ exists,

$$\text{Output value } X = X_n + \frac{X_{n+1} - X_n}{Y_{n+1} - Y_n} \times (\text{Input value } Y - Y_n) \quad (1 \leq n \leq N - 1)$$

- If the pair X_n and Y_n where $Y_n \leq \text{Input } Y \leq Y_{n+1}$ does not exist,
If Input value $Y < Y_1$ then,

$$\text{Output value } X = X_1 + \frac{X_2 - X_1}{Y_2 - Y_1} \times (\text{Input value } Y - Y_1)$$


If Input value $Y > Y_N$ then,

$$\text{Output value } X = X_N + \frac{X_N - X_{N-1}}{Y_N - Y_{N-1}} \times (\text{Input value } Y - Y_N)$$

4.8.11 Linear Accelerator/Decelerator 1 (LAU)

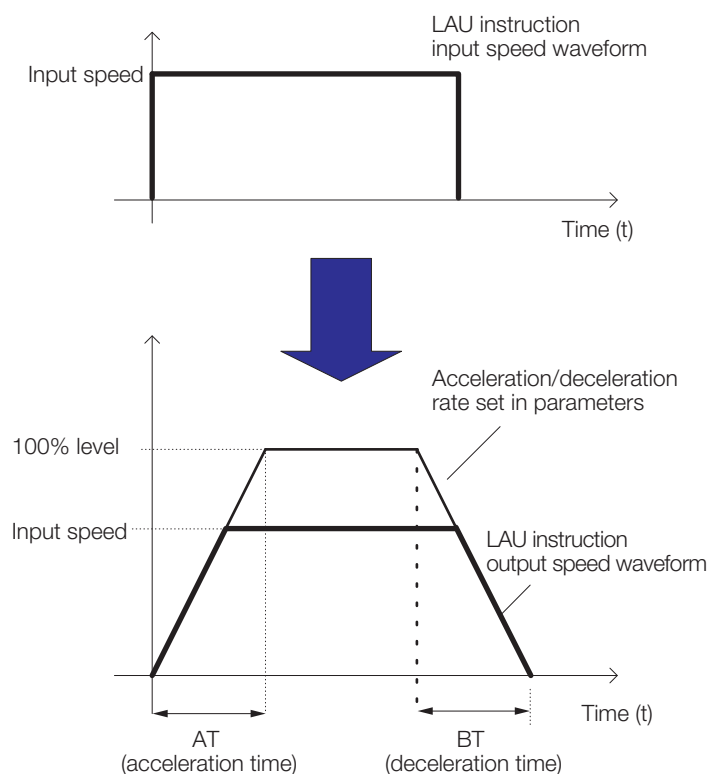
The speed that results from applying a constant acceleration or deceleration rate to the input speed is output. The acceleration or deceleration rate is applied according to predefined parameters in a parameter table. The input value to the LAU instruction can be an integer or a real number. Double-length integers, quadruple-length integers, and double-precision real numbers cannot be used.

The structure of the parameter table is different for integers and real numbers.



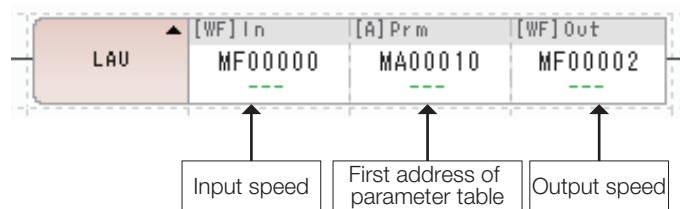
If using an integer, set an integral multiple of 1 ms for the scan time.


Important



Format

The format of this instruction is shown below.



Icon: 
Key entry: LAU

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In (Input speed)	×	○	×	×	○	×	×	○	○
Prm (First address of parameter table)	×	×	×	×	×	×	○	×	×
Out (Output speed)	×	○*	×	×	○*	×	×	○	×

* C and # registers cannot be used.

◆ Parameter Table for LAU Instruction with Integers

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RLY	Relay I/O	Relay inputs and relay outputs*	IN/OUT
1	W	LV	100% level of input	Scale for 100% input	IN
2	W	AT	Acceleration time	Time to accelerate from 0% to 100% (0.1 s)	IN
3	W	BT	Deceleration time	Time to decelerate from 100% to 0% (0.1 s)	IN
4	W	QT	Quick stop time	Time to make a quick stop from 100% to 0% (0.1 s)	IN
5	W	V	Current speed	LAU output (output to Out)	OUT
6	W	DVDT	Current acceleration/ deceleration rate	Scaling with the normal acceleration rate set to 5,000	OUT
7	W	–	(Reserved.)	Spare register	–
8	W	VIM	Previous speed reference	For storage of the previous speed reference input value	OUT
9	W	DVDTK	DVDT coefficient	Scaling factor for DVDT (Current Acceleration Rate)	IN
10	L	REM	Remainder	Remainder of the acceleration/deceleration rate	OUT

* The relay inputs and outputs are assigned as given below.

Bit	Symbol	Name	Specification	I/O
0	RN	Line running	The line is running when this input is ON.	IN
1	QS	Quick stop	A quick stop is performed if this input is turned OFF.	IN
2	DVDTF	Skip execution of DVDT operation	Execution of the DVDT operation is skipped when this input is ON.	IN
3	DVDT S	DVDT operation selection	Selects the method for calculating DVDT	IN
4 to 7	–	(Reserved.)	Spare input relays	IN
8	ARY	Accelerating	ON is output during acceleration.	OUT
9	BRY	Decelerating	ON is output during deceleration.	OUT
A	LSP	Zero speed	ON is output during zero speed.	OUT
B	EQU	Equal	ON is output when the input speed equals the output speed.	OUT
C to F	–	(Reserved.)	Spare output relays	OUT


Note: If QS (quick stop) is OFF, QT (quick stop time) is used as the acceleration/deceleration time.

◆ Parameter Table for LAU Instruction with Real Numbers

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RLY	Relay I/O	Relay inputs and relay outputs* ²	IN/OUT
1	W	–	(Reserved.)	Spare register	–
2	F	LV* ¹	100% level of input	Scale for 100% input	IN
4	F	AT	Acceleration time	Time to accelerate from 0% to 100% (s)	IN
6	F	BT	Deceleration time	Time to decelerate from 100% to 0% (s)	IN
8	F	QT	Quick stop time	Time to make a quick stop from 100% to 0% (s)	IN
10	F	V	Current speed	LAU output (output to Out)	OUT
12	F	DVDT	Current acceleration/ deceleration rate	The current acceleration or deceleration rate is output.	OUT

*1. The ratio between the set value for LV (input at 100% level) and the input speed determines the actual acceleration/deceleration time.

Refer to the following section for details on the processing that is performed internally by the LAU instruction.

 **Additional Information on page 4-158**

*2. The relay inputs and outputs are assigned as given below.

Bit	Symbol	Name	Specification	I/O
0	RN	Line running	The line is running when this input is ON.	IN
1	QS	Quick stop	A quick stop is performed if this input is turned OFF.	IN
2 to 7	–	(Reserved.)	Spare input relays	IN
8	ARY	Accelerating	ON is output during acceleration.	OUT
9	BRY	Decelerating	ON is output during deceleration.	OUT
A	LSP	Zero speed	ON is output during zero speed.	OUT
B	EQU	Equal	ON is output when the input speed equals the output speed.	OUT
C to F	–	(Reserved.)	Spare output relays	OUT

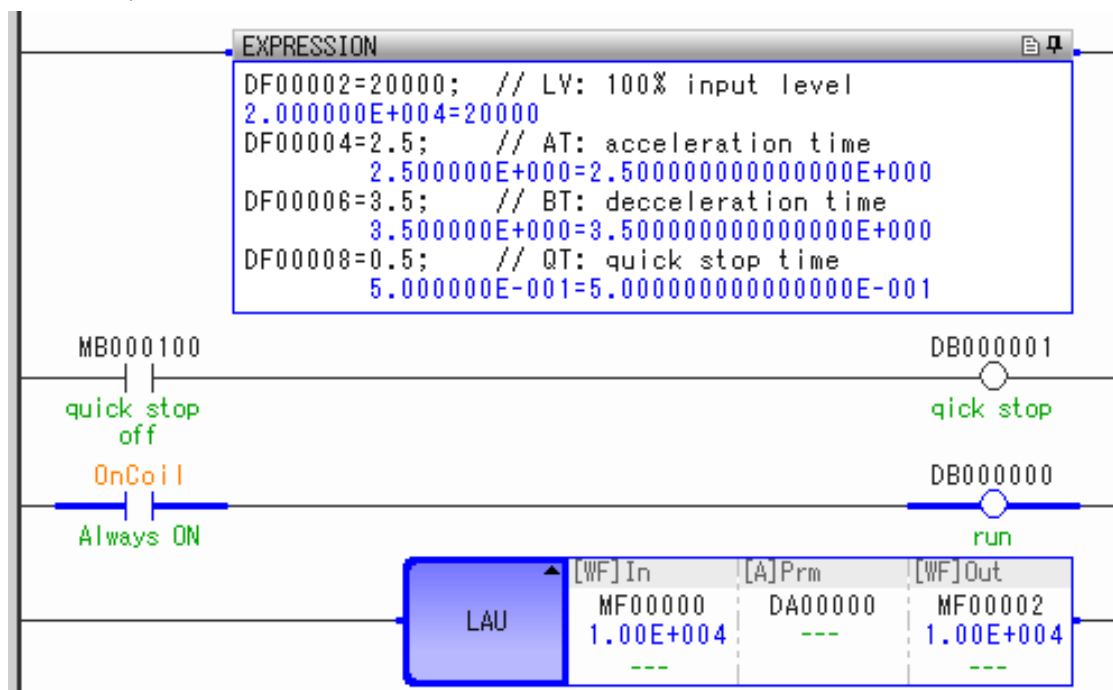
Information When QS (quick stop) is turned OFF, the acceleration/deceleration time is set to the QT (quick stop time).
To execute a quick stop, turn QS (quick stop) OFF and set the input speed to 0 at the same time.

Programming Example

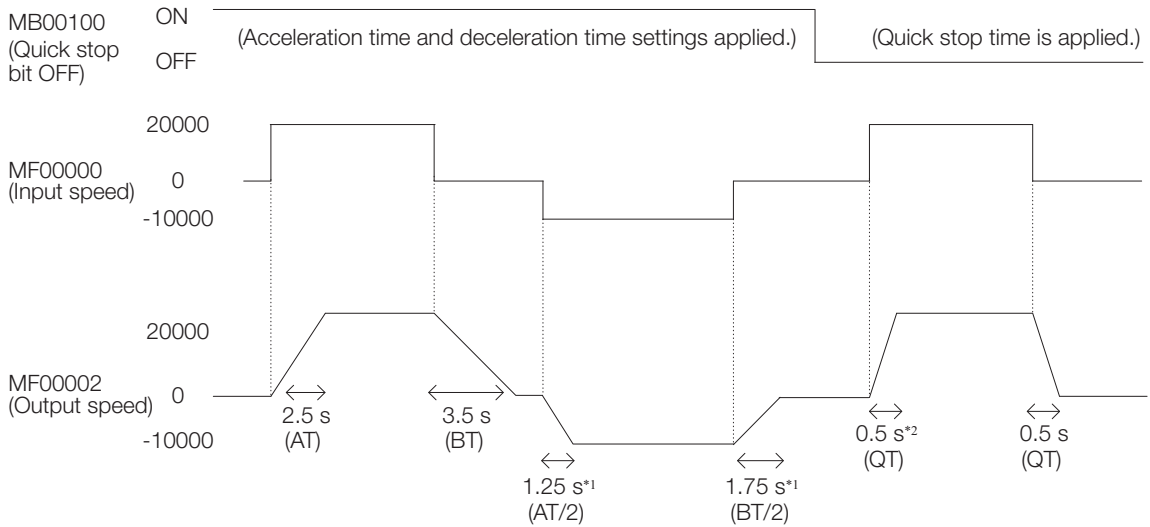
In the following programming example, the LAU instruction for real numbers is executed with the specified acceleration and deceleration rates where MF00000 is the input speed and MF00002 is the output speed.

The following parameters are set for the acceleration or deceleration rate.

- 100% level of acceleration/deceleration rate input = 20,000
- Acceleration time = 2.5 s
- Deceleration time = 3.5 s
- Quick stop time = 0.5 s



The following table shows how each register operates.



- *1. The acceleration time is applied when moving away from 0, and the deceleration time is applied when moving toward 0.
- *2. The quick stop time is also applied as the acceleration time.

Additional Information

◆ Formulas for Calculating the Speed Output Value and Current Acceleration/Deceleration Rate

This section describes the formulas for calculating the speed output values during acceleration, deceleration, and quick stops, and the current acceleration or deceleration rates.

■ LAU Instruction for Integers

The LAU instruction for integers calculates the speed output value during acceleration, deceleration, and quick stops, and the current acceleration or deceleration rates using the formula shown below based on predefined parameters.

In this formula, V is the speed output value, V' is the previous speed output value, VI is the input value for the speed reference, and Ts is the scan time set value.

• Speed Output Value during Acceleration

The speed output value during acceleration is calculated with the following formula.

Positive Side VI > V' (V' ≥ 0)	Negative Side VI < V' (V' < 0)
V = V' + ADV	V = V' - ADV
$ADV \text{ (acceleration rate)} = \frac{LV \times Ts \text{ (0.1 ms)} + REM}{AT \text{ (0.1 s)} \times 1,000}$	

• Speed Output Value during Deceleration

The speed output value during deceleration is calculated with the following formula.

Positive Side VI < V' (V' ≥ 0)	Negative Side VI > V' (V' < 0)
V = V' - BDV	V = V' + BDV
$BDV \text{ (deceleration rate)} = \frac{LV \times Ts \text{ (0.1 ms)} + REM}{BT \text{ (0.1 s)} \times 1,000}$	

- **Speed Output Value during a Quick Stop**

The speed output value during a quick stop is calculated with the following formula.

Positive Side $VI < V'$ ($V' \geq 0$)	Negative Side $VI > V'$ ($V' < 0$)
$V = V' - QDV$	$V = V' + QDV$
QDV (quick stop rate) = $\frac{LV \times Ts (0.1 \text{ ms}) + REM}{QT (0.1 \text{ s}) \times 1,000}$	

- **Current acceleration/deceleration rate**

If DVDTF (skip execution of DVDT operation) is ON, DVDT (current acceleration/deceleration rate) will be calculated according to the setting of DVDTs (DVDT operation selection) using one of the following formulas. If DVDTF is OFF, DVDT is set to 0.

DVDTs = ON	DVDTs = OFF
$DVDT = \frac{(V - V') \times 5,000}{ADV}$	$DVDT = (V - V') \times DVDTK$

Information

1. ARY (accelerating) turns ON at the following times:
When $V' \geq 0$ and $ADV > 0$, or when $V' \leq 0$ and $ADV < 0$
2. BRV (decelerating) turns ON at the following times:
 - When $V' < 0$ and $BDV > 0$, or when $V' > 0$ and $BDV < 0$
 - When $V' < 0$ and $QDV > 0$, or when $V' > 0$ and $QDV < 0$
3. LSP (zero speed) turns ON when V equals 0.
4. EQU (equal) turns ON when VI equals V .
5. If RN (line running) is OFF, the outputs for V , DVDT, and REM are set to 0.

- **LAU Instruction for Real Numbers**

The LAU instruction for real numbers calculates the speed output value during acceleration, deceleration, and quick stops, and the current acceleration or deceleration rates using the formula shown below based on predefined parameters.

In this formula, V is the speed output value, V' is the previous speed output value, VI is the input value for the speed reference, and Ts is the scan time set value.

- **Speed Output Value during Acceleration**

The speed output value during acceleration is calculated with the following formula.

Positive Side $VI > V'$ ($V' \geq 0$)	Negative Side $VI < V'$ ($V' < 0$)
$V = V' + ADV$	$V = V' - ADV$
ADV (acceleration rate) = $\frac{LV \times Ts (0.1 \text{ ms})}{AT (s) \times 10,000}$	

- **Speed Output Value during Deceleration**

The speed output value during deceleration is calculated with the following formula.

Positive Side $VI < V'$ ($V' \geq 0$)	Negative Side $VI > V'$ ($V' < 0$)
$V = V' - BDV$	$V = V' + BDV$
BDV (deceleration rate) = $\frac{LV \times Ts (0.1 \text{ ms})}{BT (s) \times 10,000}$	

- **Speed Output Value during a Quick Stop**

The speed output value during a quick stop is calculated with the following formula.

Positive Side $VI < V'$ ($V' \geq 0$)	Negative Side $VI > V'$ ($V' < 0$)
$V = V' - QDV$	$V = V' + QDV$
QDV (quick stop rate) = $\frac{LV \times Ts (0.1 \text{ ms})}{QT (s) \times 10,000}$	

• **Current acceleration/deceleration rate**

The DVDT (current acceleration/deceleration rate) is calculated with the following formula after V (speed output) has been calculated.

$$DVDT = V - V'$$

Information

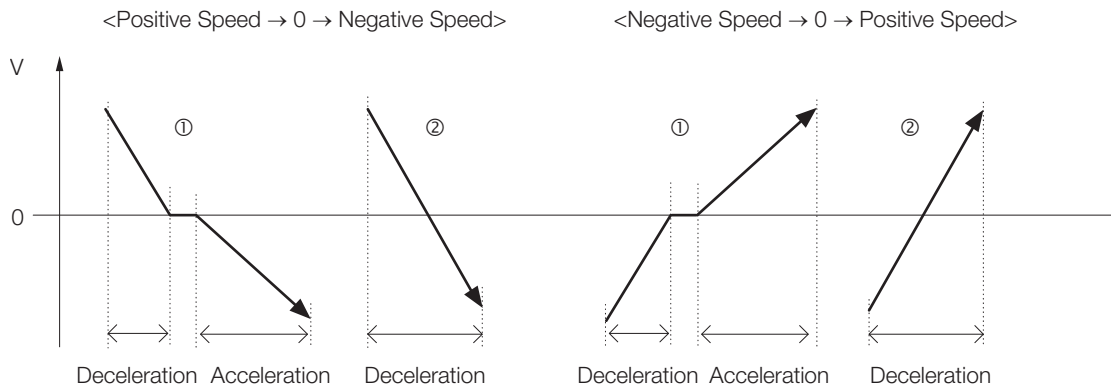
1. ARY (accelerating) turns ON at the following times:
When $V' \geq 0$ and $ADV > 0$, or when $V' \leq 0$ and $ADV < 0$
2. BRV (decelerating) turns ON at the following times:
 - When $V' < 0$ and $BDV > 0$, or when $V' > 0$ and $BDV < 0$
 - When $V' < 0$ and $QDV > 0$, or when $V' > 0$ and $QDV < 0$
3. LSP (zero speed) turns ON when V equals 0.
4. EQR (equal) turns ON when VI equals V.
5. ARY (accelerating) turns ON at the following times:
 $V \neq V'$, and DVDT and V have the same sign.
6. BRV (decelerating) turns ON at the following times:
 $V \neq V'$, and DVDT and V do not have the same sign.
7. If RN (line running) is OFF, the outputs for V and DVDT are set to 0.

◆ **Acceleration and Deceleration When Input Speed Is Changed**

This section describes acceleration and deceleration when the input speed is changed.

After the axis stops when the speed is set to 0, acceleration and deceleration are controlled by the predefined deceleration time and acceleration time. (See ①.)

If the speed reference crosses the point where speed equals 0, acceleration and deceleration are controlled by the deceleration time to keep the speed from fluctuating. (See ②.)




4.8.12 Linear Accelerator/Decelerator 2 (SLAU)

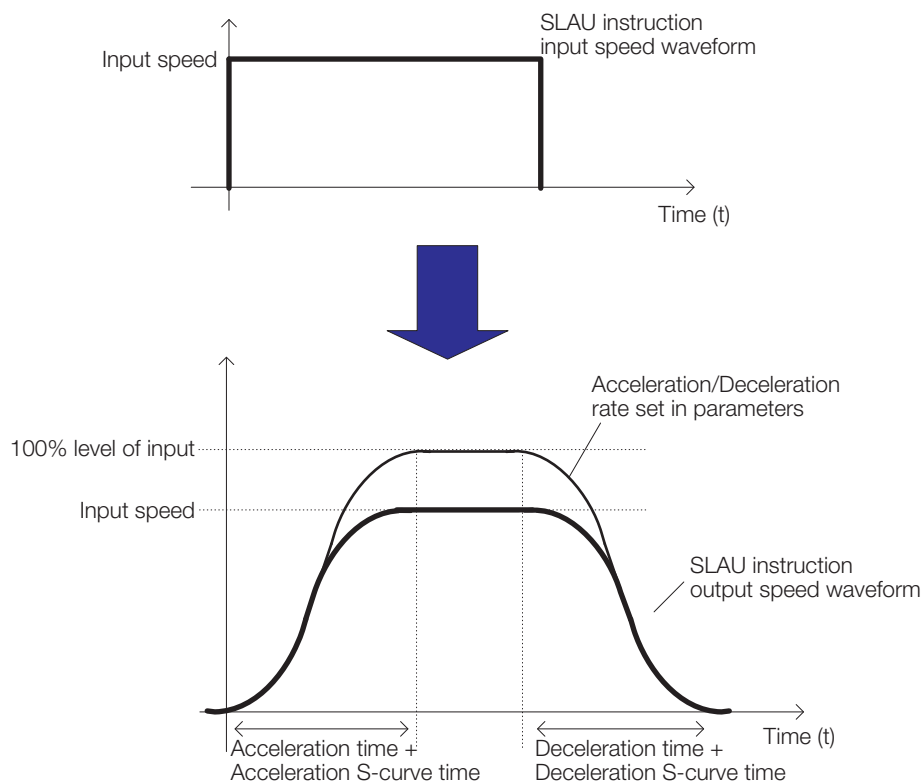
The speed that results from applying a variable acceleration or deceleration rate to the input speed is output. The acceleration or deceleration rate is applied as an S curve according to predefined parameters in a parameter table.

The input value to the SLAU instruction can be an integer, double-length integer, or a real number. Quadruple-length integers and double-precision real numbers cannot be used.

The structure of the parameter table is different for integers and real numbers.

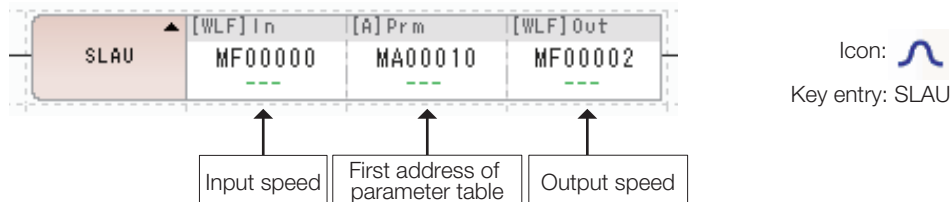


Important If using an integer, set an integral multiple of 1 ms for the scan time.



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In (Input speed)	×	○	○	×	○	×	×	○	○
Prm (First address of parameter table)	×	×	×	×	×	×	○*	○*	×
Out (Output speed)	×	○*	○*	×	○*	×	×	○	×

* C and # registers cannot be used.

◆ Parameter Table for SLAU Instruction with Integers

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RLY	Relay I/O	Relay inputs and relay outputs*	IN/OUT
1	W	LV	100% level of input	Scale for 100% input	IN
2	W	AT	Acceleration time	Time to accelerate from 0% to 100% (0.1 s)	IN
3	W	BT	Deceleration time	Time to decelerate from 100% to 0% (0.1 s)	IN
4	W	QT	Quick stop time	Time to make a quick stop from 100% to 0% (0.1 s)	IN
5	W	AAT	Acceleration S-curve time	Acceleration S-curve region time (0.01 to 32.00 s)	IN
6	W	BBT	Deceleration S-curve time	Deceleration S-curve region time (0.01 to 32.00 s)	IN
7	W	V	Current speed	SLAU output (output to Out)	OUT
8	W	DVDT1	Current acceleration/ deceleration rate 1	Scaling with the normal acceleration rate set to 5,000	OUT
9	W	–	(Reserved.)	Spare register	–
10	W	ABMD	Speed increase when holding	Amount of speed change until the speed stabilizes after the hold command is executed	OUT
11	W	REM1	Remainder	Remainder of the acceleration/deceleration rate	OUT
12	W	–	(Reserved.)	Spare register	–
13	W	VIM	Previous speed reference	For storage of the previous speed reference input value	OUT
14	L	DVDT2	Current acceleration/ deceleration rate 2	1,000 times the actual acceleration/deceleration	OUT
16	L	DVDT3	Current acceleration/ deceleration rate 3	Current acceleration/deceleration rate (= DVDT2/1,000)	OUT
18	L	REM2	Remainder	Remainder of the S-curve region acceleration/deceleration rate	OUT
20	W	REM3	Remainder	Remainder of the current speed	OUT
21	W	DVDTK	DVDT1 coefficient	Scaling factor for DVDT (Current Acceleration Rate 1) (–32768 to 32767)	OUT

* The relay inputs and outputs are assigned as given below.

Bit	Symbol	Name	Specification	I/O
0	RN	Line running	The line is running when this input is ON.	IN
1	QS	Quick stop	A quick stop is performed if this input is turned OFF.	IN
2	DVDTF	Skip execution of DVDT1 operation	Execution of the DVDT operation is skipped when this input is ON.	IN
3	DVDTS	DVDT1 operation selection	Selects the method for calculating DVDT	IN
4 to 7	–	(Reserved.)	Spare input relays	IN
8	ARY	Accelerating	ON is output during acceleration.	OUT
9	BRY	Decelerating	ON is output during deceleration.	OUT
A	LSP	Zero speed	ON is output during zero speed.	OUT
B	EQU	Equal	ON is output when the input speed equals the output speed.	OUT
C	–	(Reserved.)	Spare output relays	OUT
D	CCF	Work relay	System internal work relay	OUT
E	BBF	Work relay	System internal work relay	OUT
F	AAF	Work relay	System internal work relay	OUT

Note: If QS (quick stop) is OFF, QT (quick stop time) is used as the deceleration time.

◆ Parameter Table for SLAU Instruction with Double-length Integers

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RLY	Relay I/O	Relay inputs and relay outputs*2	IN/OUT
1	W	–	(Reserved.)	–	–
2	L	LV	100% level of input	Scale for 100% of input value	IN
4	L	AT	Acceleration time	Time to accelerate from 0% to 100% (0.1 s)	IN
6	L	BT	Deceleration time	Time to decelerate from 100% to 0% (0.1 s)	IN
8	L	QT	Quick stop time	Time to make a quick stop from 100% to 0% (0.1 s)	IN
10	L	AAT	Acceleration S-curve time	Acceleration S-curve region time (0.01 s)	IN
12	L	BBT	Deceleration S-curve time	Deceleration S-curve region time (0.01 s)	IN
14	L	V	Current speed	SLAU output (also the output to the A register)	OUT
16	L	DVDT	Current acceleration/ deceleration rate	The current acceleration/deceleration rate (truncated below the decimal point) is output.	OUT
18	L	ABMD	Speed increase when holding	Amount of speed change until the speed stabilizes after the hold command is executed	OUT
20	D*1	V_D	Current speed	SLAU output for system use (double-precision real number)	IN/OUT
24	D*1	DVDT_D	Current acceleration/ deceleration rate	Current acceleration or deceleration rate for system use (double-precision real number)	IN/OUT

*1. D is a double-precision real number expressed in 4 words. The MPE720 cannot display this value as a real number.

*2. The relay inputs and outputs are assigned as given below.

Bit	Symbol	Name	Specification	I/O
0	RN	Line running	The line is running when this input is ON.	IN
1	QS	Quick stop	A quick stop is performed if this input is turned OFF.	IN
2	DVDTF	Acceleration/ deceleration rate flag	When the input turns ON, DVDT (current acceleration/ deceleration rate) is multiplied by 1,000 and then output.	IN
3 to 7	–	(Reserved.)	Spare input relays	IN
8	ARY	Accelerating	ON is output during acceleration.	OUT
9	BRY	Decelerating	ON is output during deceleration.	OUT
A	LSP	Zero speed	ON is output during zero speed.	OUT
B	EQU	Equal	ON is output when the input value equals the output value.	OUT
C to F	–	Work relay	System internal work relay	IN/OUT

Note: If QS (quick stop) is OFF, QT (quick stop time) is used as the deceleration time.

◆ Parameter Table for SLAU Instruction with Real Numbers

Address	Data Type	Symbol	Name	Specification	I/O
0	W	RLY	Relay I/O	Relay inputs and relay outputs*	IN/OUT
1	W	–	(Reserved.)	Spare register	–
2	F	LV	100% level of input	Scale for 100% input	IN
4	F	AT	Acceleration time	Time to accelerate from 0% to 100% (s)	IN
6	F	BT	Deceleration time	Time to decelerate from 100% to 0% (s)	IN
8	F	QT	Quick stop time	Time to make a quick stop from 100% to 0% (s)	IN
10	F	AAT	Acceleration S-curve time	Acceleration S-curve region time (s)	IN
12	F	BBT	Deceleration S-curve time	Deceleration S-curve region time (s)	IN

Continued on next page.

Continued from previous page.

Address	Data Type	Symbol	Name	Specification	I/O
14	F	V	Current speed	SLAU output (output to Out)	OUT
16	F	DVDT1	Current acceleration/ deceleration rate 1	The actual acceleration or deceleration rate is output.	OUT
18	F	ABMD	Speed increase when holding	Amount of speed change until the speed stabilizes after the hold command is executed	OUT

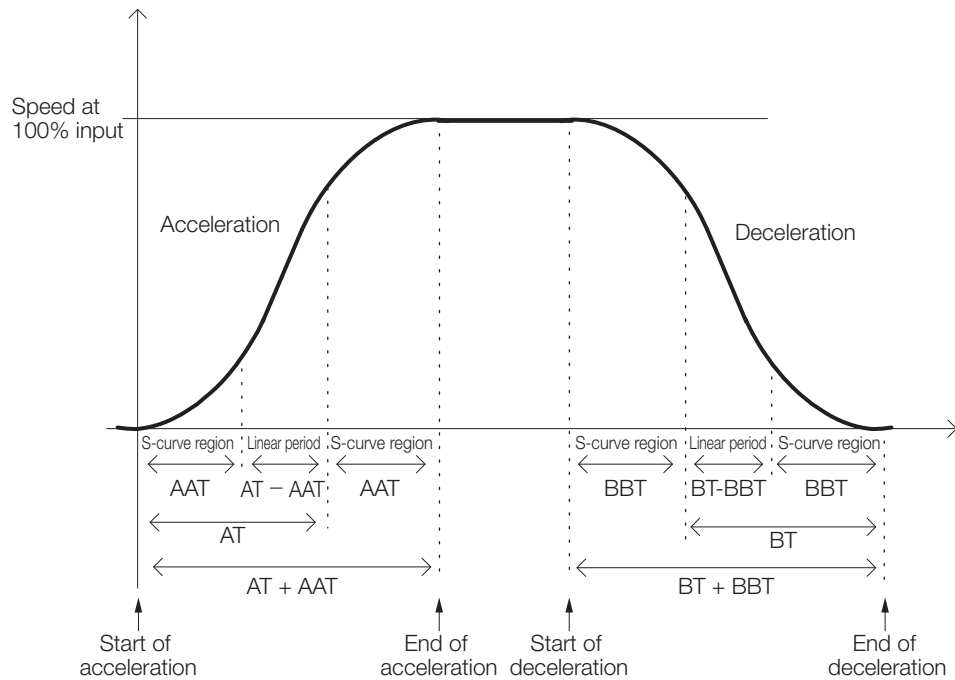
* The relay inputs and outputs are assigned as given below.

Bit	Symbol	Name	Specification	I/O
0	RN	Line running	The line is running when this input is ON.	IN
1	QS	Quick stop	A quick stop is performed if this input is turned OFF.	IN
2 to 7	-	(Reserved.)	Spare input relays	IN
8	ARY	Accelerating	ON is output during acceleration.	OUT
9	BRY	Decelerating	ON is output during deceleration.	OUT
A	LSP	Zero speed	ON is output during zero speed.	OUT
B	EQU	Equal	ON is output when the input speed equals the output speed.	OUT
C to F	-	(Reserved.)	Spare output relays	OUT

Note: If QS (quick stop) is OFF, QT (quick stop time) is used as the deceleration time.

Example

The following figure shows how the parameters are used in the actual instruction.



Note: Refer to the following section for details on the processing that is performed internally by the SLAU instruction.

Additional Information on page 4-166

Information

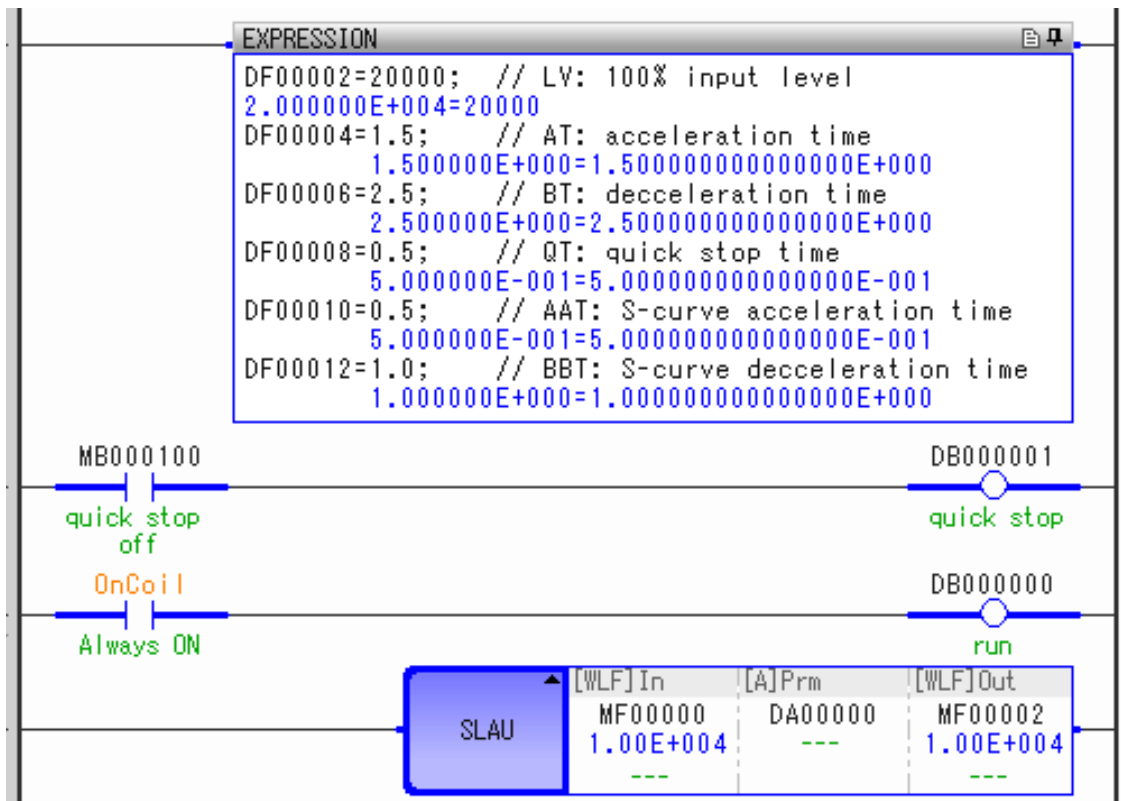
When QS (quick stop) is turned OFF, the output decelerates at QT (quick stop time) and the output speed is set to 0. It is not necessary to set the input speed to 0. For a quick stop, the speed is decelerated linearly without applying the S-curve. Set the parameters so that AT or BT (linear acceleration or deceleration time) is greater than or equal to AAT or BBT (S-curve acceleration or deceleration time).

Programming Example

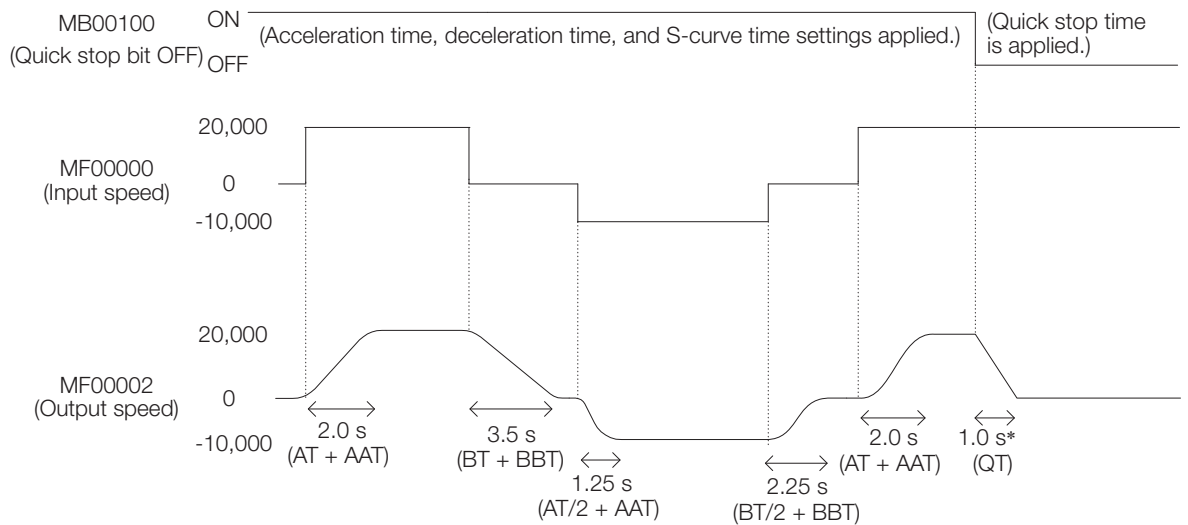
In the following programming example, the SLAU instruction for real numbers is executed with the specified acceleration and deceleration rates where MF00000 is the input speed and MF00002 is the output speed.

The following parameters are set for the acceleration or deceleration rate.

- Speed when input level of acceleration or deceleration rate is 100% = 20,000
- Acceleration time = 1.5 s
- Deceleration time = 2.5 s
- Quick stop time = 0.5 s
- Acceleration S-curve time = 0.5 s
- Deceleration S-curve time = 1.0 s



The following table shows how each register operates.



* If the quick stop bit is turned OFF, the speed is decelerated to a stop using the quick stop time, regardless of the S-curve time and input speed.

Additional Information

◆ Formulas for Calculating the Speed Output Value and Current Acceleration/Deceleration Rate

This section describes the formulas for calculating the speed output values during acceleration, deceleration, quick stops, S-curve acceleration, S-curve deceleration, and the current acceleration or deceleration rates.

■ Operation of the SLAU Instruction for Integers

The SLAU instruction for integers calculates the speed output value during acceleration, deceleration, quick stops, S-curve acceleration, S-curve deceleration, and the current acceleration or deceleration rates using the formulas shown below based on predefined parameters.

In this formula, V is the speed output value, V' is the previous speed output value, VI is the input value for the speed reference, and Ts is the scan time set value.

• Speed Output Value during Acceleration

The speed output value during acceleration is calculated with the following formula.

Outside the S-curve region (ADVS > ADV)	
Positive Side VI > V' (V' ≥ 0)	Negative Side VI < V' (V' < 0)
V = V' + ADV	V = V' - ADV
$ADV \text{ (acceleration rate)} = \frac{LV \times Ts \text{ (0.1 ms)} + REM1}{AT \text{ (0.1 s)} \times 1,000}$	

• Speed Output Value during Deceleration

The speed output value during deceleration is calculated with the following formula.

Outside the S-curve region (BDVS > BDV)	
Positive Side VI < V' (V' ≥ 0)	Negative Side VI > V' (V' < 0)
V = V' - BDV	V = V' + BDV
$BDV \text{ (deceleration rate)} = \frac{LV \times Ts \text{ (0.1ms)} + REM1}{BT \text{ (0.1 s)} \times 1,000}$	

• Speed Output Value during a Quick Stop

The speed output value during a quick stop is calculated with the following formula.

QS = OFF	
Positive Side VI < V' (V' ≥ 0)	Negative Side VI > V' (V' < 0)
V = V' - QDV	V = V' + QDV
$QDV \text{ (quick stop rate)} = \frac{LV \times Ts \text{ (0.1ms)} + REM1}{QT \text{ (0.1 s)} \times 1,000}$	

Information For a quick stop, the speed is decelerated linearly without applying the S-curve.

• Speed Output Value during S-Curve Acceleration

The speed output value during S-curve acceleration is calculated with the following formula.

Inside the S-curve region (ADVS < ADV)	
Positive Side VI > V' (V' ≥ 0)	Negative Side VI < V' (V' < 0)
V = V' + ADVS	V = V' - ADVS
$ADVS \text{ (S-curve region acceleration rate)} = ADVS' \pm AADVS$ $AADVS = \frac{ADV \times Ts \text{ (0.1 ms)} + REM2}{AAT \text{ (0.01 s)} \times 100}$	

- **Speed Output Value during S-Curve Deceleration**

The speed output value during S-curve deceleration is calculated with the following formula.

Inside the S-curve region (BDVS < BDV)	
Positive Side VI < V' (V' ≥ 0)	Negative Side VI > V' (V' < 0)
V = V' - BDVS	V = V' + BDVS

BDVS (S-curve region deceleration rate) = BDVS' ± BBDVS

$$BBDVS = \frac{BDV \times Ts (0.1 \text{ ms}) + REM2}{BBT (0.01 \text{ s}) \times 100}$$

- **Current acceleration/deceleration rate**

If DVDTF (skip execution of DVDT1 operation) is ON, DVDT1 (current acceleration/deceleration rate) will be calculated according to the setting of DVDTs (DVDT1 operation selection) using one of the following formulas. If DVDTF is OFF, DVDT1 is set to 0.

$$\text{If DVDTs} = \text{ON, DVDT1} = \frac{(V - V') \times 5,000}{ADV}$$

$$\text{If DVDTs is OFF, DVDT1} = (V - V') \times DVDTK$$

The value for DVDT2 (current acceleration/deceleration rate 2) is calculated as follows:

During acceleration: Inside the S-curve region: DVDT2 = ±ADV

Outside the S-curve region: DVDT2 = ±ADV

During deceleration: Inside the S-curve region: DVDT2 = ±BDVS

Outside the S-curve region: DVDT2 = ±BDV

During a quick stop: DVDT = ±QDV

The result of ABMD (speed increase upon holding) is output after the following operation is performed.

$$ABMD = \frac{DVDT2' \times DVDT2'}{2 \times AADVS (BBDVS)}$$

DVDT2' : Previous value of DVDT2 (current acceleration/deceleration rate 2)

Information

1. ARY (accelerating) turns ON at the following times:
 - When V' ≥ 0 and ADV > 0, or when V' ≤ 0 and ADV < 0
 - If V' ≥ 0 and ADVS > 0 inside an S-curve region, or if V' ≤ 0 and ADVS < 0 inside an S-curve region
2. BRV (decelerating) turns ON at the following times:
 - When V' < 0 and BDV > 0, or when V' > 0 and BDV < 0
 - When V' < 0 and QDV > 0, or when V' > 0 and QDV < 0
 - If V' < 0 and BDVS > 0 inside an S-curve region, or if V' > 0 and BDVS < 0 inside an S-curve region
3. LSP (zero speed) turns ON when V equals 0.
4. EQU (equal) turns ON when VI equals V.
5. If RN (line running) is OFF, the outputs for V, DVDT1, DVDT2, DVDT3, REM1, REM2, and REM3 are set to 0.

■ Operation of the SLAU Instruction for Double-length Integers or Real Numbers

The SLAU instruction for double-length integers or real numbers calculates the speed output value during acceleration, deceleration, quick stops, S-curve acceleration, S-curve deceleration, and the current acceleration or deceleration rates using the formulas shown below.

In this formula, V is the speed output value, V' is the previous speed output value, VI is the input value for the speed reference, Ts is the scan time set value, ADVS' is the previous ADVS value, and BDVS' is the previous BDVS value.

• Speed Output Value during Acceleration

The speed output value during acceleration is calculated with the following formula.

Outside the S-curve region (ADVS > ADV)	
Positive Side VI > V' (V' ≥ 0)	Negative Side VI < V' (V' < 0)
V = V' + ADV	V = V' - ADV
$ADV \text{ (acceleration rate)} = \frac{LV \times Ts \text{ (0.1 ms)}}{AT \text{ (s)} \times 10,000}$	

• Speed Output Value during Deceleration

The speed output value during deceleration is calculated with the following formula.

Outside the S-curve region (BDVS > BDV)	
Positive Side VI < V' (V' ≥ 0)	Negative Side VI > V' (V' < 0)
V = V' - BDV	V = V' + BDV
$BDV \text{ (deceleration rate)} = \frac{-LV \times Ts \text{ (0.1 ms)}}{BT \text{ (s)} \times 10,000}$	

• Speed Output Value during a Quick Stop

The speed output value during a quick stop is calculated with the following formula.

QS = OFF	
Positive Side VI < V' (V' ≥ 0)	Negative Side VI > V' (V' < 0)
V = V' - QDV	V = V' + QDV
$QDV \text{ (quick stop rate)} = \frac{LV \times Ts \text{ (0.1 ms)}}{QT \text{ (s)} \times 10,000}$	

Information For a quick stop, the speed is decelerated linearly without applying the S-curve.

• Speed Output Value during S-Curve Acceleration

The speed output value during S-curve acceleration is calculated with the following formula.

Inside the S-curve region (ADVS < ADV)	
Positive Side VI > V' (V' ≥ 0)	Negative Side VI < V' (V' < 0)
V = V' + ADVS	V = V' - ADVS
$ADVS \text{ (S-curve region acceleration rate)} = ADVS' \pm AADVS$ $AADVS = \frac{ADV \times Ts \text{ (0.1 ms)}}{AAT \text{ (s)} \times 10,000}$	

• Speed Output Value during S-Curve Deceleration

The speed output value during S-curve deceleration is calculated with the following formula.

Inside the S-curve region (BDVS < BDV)	
Positive Side VI < V' (V' ≥ 0)	Negative Side VI > V' (V' < 0)
V = V' - BDVS	V = V' + BDVS
$BDVS \text{ (S-curve region deceleration rate)} = BDVS' \pm BBDVS$ $BBDVS = \frac{BDV \times Ts \text{ (0.1 ms)}}{BBT \text{ (s)} \times 10,000}$	

- **Current acceleration/deceleration rate**

The value of DVDT1 (current acceleration/deceleration rate 1) is output after the following operation is performed:

During acceleration: Inside the S-curve region: $DVDT = ADVS$
 Outside the S-curve region: $DVDT = ADV$
 During deceleration: Inside the S-curve region: $DVDT = BDVS$
 Outside the S-curve region: $DVDT = BDV$
 During a quick stop: $DVDT = QDV$

The result of ABMD (speed increase upon holding) is output after the following operation is performed.

$$ABMD = \frac{DVDT \times DVDT}{2 \times AADVS (BBDVS)}$$

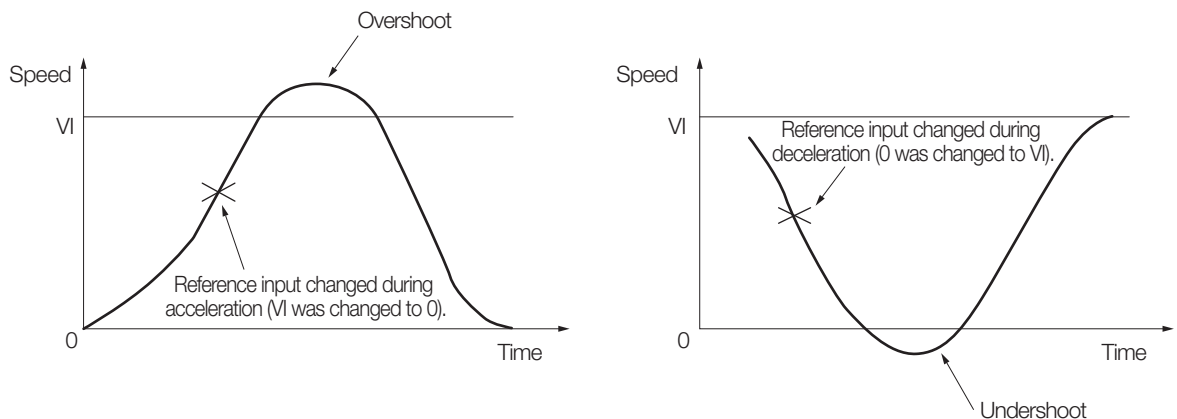
Information

1. LSP (zero speed) turns ON when V equals 0.
2. EQU (equal) turns ON when VI equals V.
3. If RN (line running) is OFF, the outputs for V, DVDT, and AVMD are set to 0.

◆ Precautions in Using the SLAU Instruction

■ Changing the Input Value for VI (Input Speed) during Acceleration or Deceleration

If VI (input value) is to be changed while accelerating or decelerating, do not use the SLAU instruction for integers. Otherwise, overshooting or undershooting may occur as shown in the following figures.

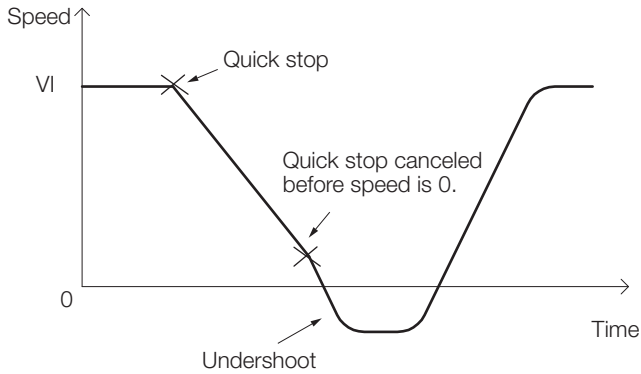


If VI (input value) is to be changed while accelerating or decelerating, take one of the following measures in your application program.

- Use the SLAU instruction for real numbers.
- Use the SLAU instruction for integers in conjunction with the LIMIT instruction. Specifically, use the output value of the SLAU instruction for integers as the input value to the LIMIT instruction to prevent overshooting or undershooting.

■ **Cancelling a Quick Stop While Decelerating during a Quick Stop**

When decelerating for a quick stop, do not cancel the quick stop before the output speed reaches 0. Otherwise, undershooting may occur while approaching the input speed.



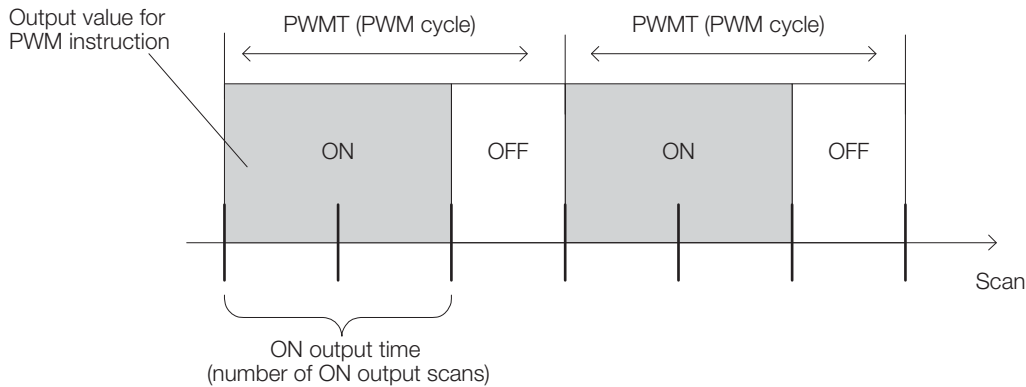
If you must reset the quick stop before the output speed reaches 0, use the LIMIT instruction on the output speed to prevent undershooting.

4.8.13 Pulse Width Modulation (PWM)

The input value (from -100.00% to 100.00%) is converted using pulse-width modulation and the result is output to the output value and parameter table. The input value can be used only with integer data, and the output value can be used only with bit data. Double-length integers and real numbers cannot be used.

Important

If using an integer, set an integral multiple of 1 ms for the scan time.



The ON output time and number of ON output scans of the PWM instruction can be calculated with the following formula.

X is the input value, PWMT is the PWM cycle (ms), and Ts is the scan time set value (ms).

$$\text{ON output time} = \frac{\text{PWMT} (X + 10,000)}{20,000}$$

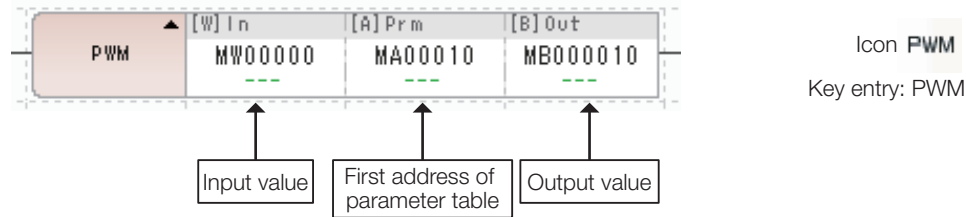
$$\text{Number of ON output scans} = \frac{\text{PWMT} (X + 10,000)}{\text{Ts} \times 20,000}$$

Information

1. The relation between the input value and the PWM output ON ratio is shown below.
 - Input value 100.00% → 100% ON (ON output time = PWMT)
 - Input value 0.00% → 50% ON (ON output time = PWMT/2)
 - Input value -100.00% → 0% ON (ON output time = 0)
2. After turning ON the power supply, turn ON PWMRST (PWM reset) to clear all internal calculations before using the PWM instruction. The PWM operation will start executing from the point when the PWM reset bit was turned ON.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In (Input value)	×	○	×	×	×	×	×	○	○
Prm (First address of parameter table)	×	×	×	×	×	×	○*	○*	×
Out (Output value)	○*	×	×	×	×	×	×	○	×

* C and # registers cannot be used.

◆ Ranges of Input and Output Values

The input value must be between -10,000 and 10,000 in units of 0.01%.

If the input exceeds this range, processing is performed for the upper limit (10,000) and the lower limit (-10,000).

The output value is set to 1 when the PWM output is ON, or to 0 when the PWM output is OFF.

◆ Parameter Table

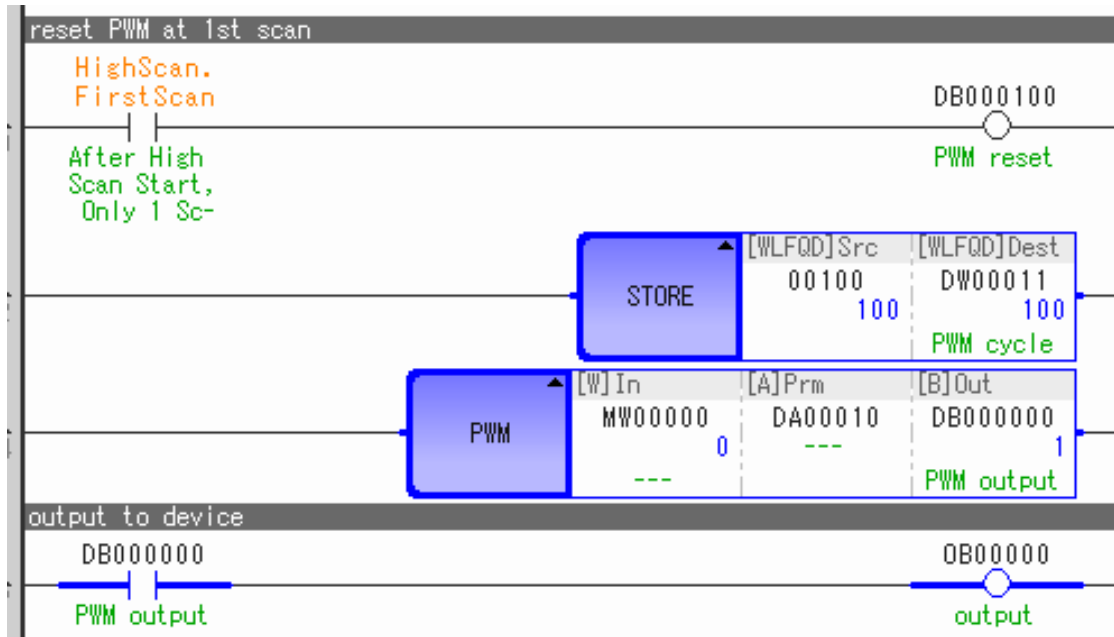
Address	Data Type	Symbol	Name	Specification	I/O
0	W	RLY	Relay I/O	Relay inputs and relay outputs*	IN/OUT
1	W	RWMT	PWM cycle	PWM cycle (1 ms) Range: 1 to 32,767 ms	IN
2	W	ONCNT	ON output setting timer	ON output setting timer (1 ms)	OUT
3	W	CVON	ON output counting timer	ON output counting timer (1 ms)	OUT
4	W	CVONREM	ON output counting timer remainder	ON output counting timer remainder (0.1 ms)	OUT
5	W	OFFCNT	OFF output setting timer	OFF output setting timer (1 ms)	OUT
6	W	CVOFF	OFF output counting timer	OFF output counting timer (1 ms)	OUT
7	W	CVOFFREM	OFF output counting timer remainder	OFF output counting timer remainder (0.1 ms)	OUT

* The relay inputs and outputs are assigned as given below.

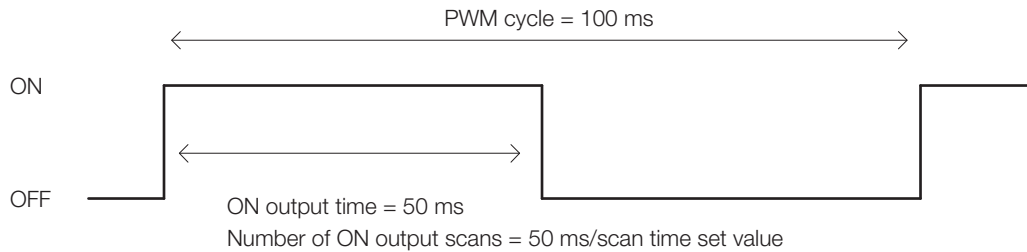
Bit	Symbol	Name	Specification	I/O
0	PWMRST	PWM reset bit	Turn ON this input to reset the PWM operation.	IN
2 to 7	–	(Reserved.)	Spare input relays	IN
8	PWMOUT	PWM output	PWM Output (The output value is set to 1 when the output is ON, or to 0 when the output is OFF.)	OUT
9 to F	–	(Reserved.)	Spare output relays	OUT

Programming Example

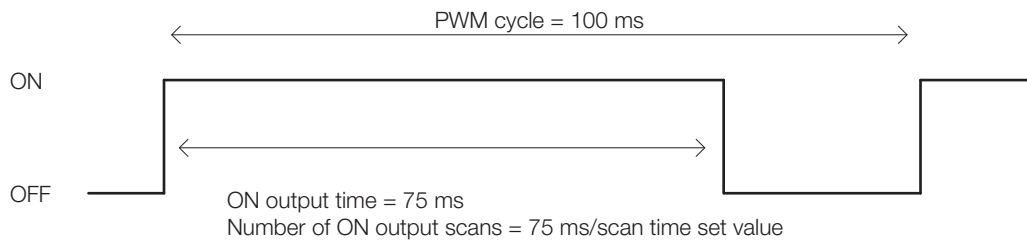
In the following programming example, the PWM output for the input value in MW00000 is stored in OB000000 where the PWM cycle is 100 ms.



This figure shows the output of OB000000 when MW00000 is 0 (0%: ON output time is 1/2 of the PWM cycle).



This figure shows the output of OB000000 when MW00000 is 5,000 (50%: ON output time is 3/4 of the PWM cycle).



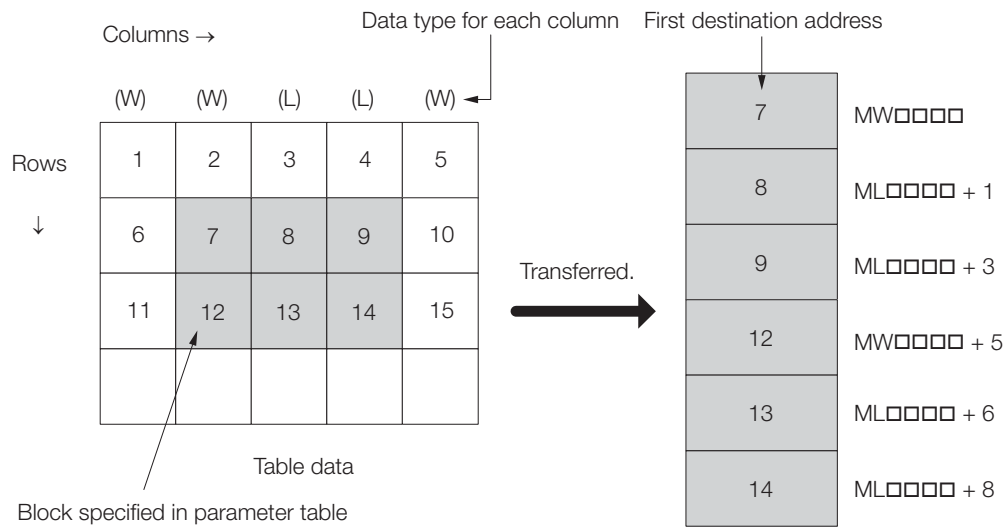
4.9 Table Manipulation Instructions

4.9.1 Read Table Block (TBLBR/TBLBRE)

A block of table data that is specified by the table name, row number, and column number is moved to a continuous area that starts at the first destination address. The data is stored in the destination area according to the data type of the elements that were read.

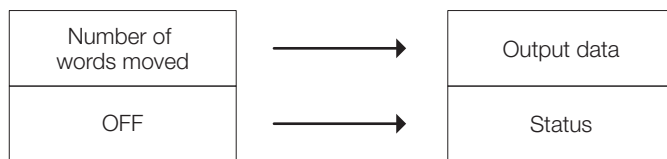
If an error occurs when accessing the table, such as data that is out of range or not enough data length at the destination, an error is output and no data is read. The contents in the destination area will remain unchanged.

If the instruction ends normally, the number of words that were moved is output, and the status is turned OFF. If an error occurs, an error code is output and the status is turned ON.

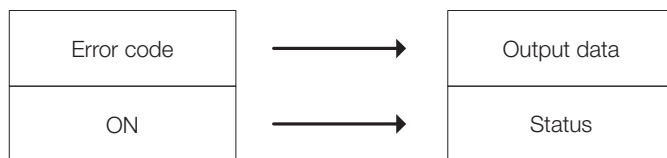


Data is stored according to the data type of the table data.

• If the Move Succeeds



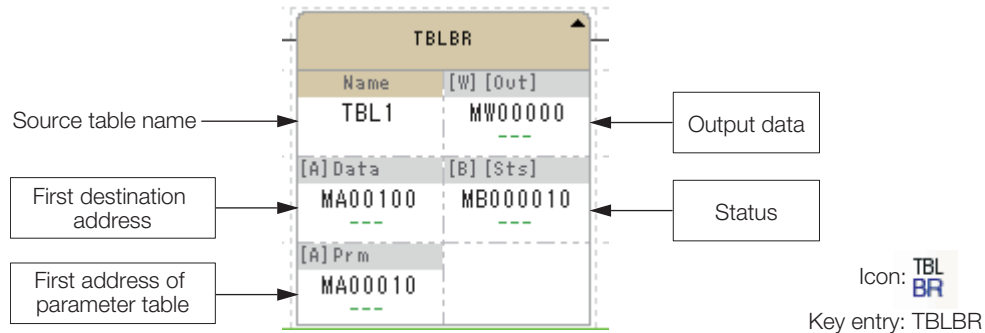
• If the Move Fails



Information If the move fails, the destination area will retain the contents from before the instruction was executed.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Name (Table name)*1*2	×	×	×	×	×	×	○	×	×
Data (First destination address)	×	×	×	×	×	×	○*4	×	×
Prm (First address of parameter table)	×	×	×	×	×	×	○	×	×
Out (Output data)*3	×	○*4	×	×	×	×	×	○	×
Sts (Status)*3	○*4	×	×	×	×	×	×	×	×

- *1. Specify the registers in which the text string for the table name (8 bytes + NULL character max.) has been stored.
- *2. G, M, D, or C register only.
- *3. Optional.
- *4. C and # registers cannot be used.

Details on I/O Items

Item	Name	I/O	Description
Name	Table name	IN	For the TBLBR instruction, directly enter the table name. For the TBLBRE instruction, indirectly designate the table name in registers.
Data	First destination address	IN	Specify the first address of the destination.
Prm	First address of parameter table	IN/OUT	Specify the first address of the table data.
Out	Output data	OUT	Specify the destination address of the output data.
Sts	Status	OUT	Specify the address for checking the status of this instruction.

Note the following precautions regarding Name.

- Always add a NULL character to Name. If a NULL character is not added to Name, a fixed length of 8 bytes (4 words) of data is read and handled as the table name. There is a risk that an unintended table name may be referenced because of this.
- When characters have been set with ASCII instructions, the NULL character is not set at the end of the text string. Use the STRSET instruction to set Name.
- An operation error will occur if the size from the first register to the maximum range of registers is less than 8 bytes (4 words).

◆ Parameter Table

Address	Data Type	Symbol	Name	Specification	I/O
0	L	ROW1	First row number of table elements	First row number of table elements to move (1 to 65,535)	IN
2	L	COL1	First column number of table elements	First column number of table elements to move (1 to 32,767)	IN
4	W	RLEN	Number of row elements	Number of row elements (1 to 32,767)	IN
5	W	CLEN	Number of column elements	Number of column elements (1 to 32,767)	IN

◆ Error Codes

Error Code	Error Name	Meaning
0001 hex	Table undefined	The target table is undefined.
0002 hex	Outside range of row numbers	The row number of the table element is outside the target table.
0003 hex	Outside range of column numbers	The column number of the table element is outside the target table.
0004 hex	Incorrect number of elements	The number of target elements is invalid.
0005 hex	Insufficient storage area	The storage area is insufficient.
0006 hex	Insufficient element type	The data type specified for the element is wrong.
0007 hex	Queue buffer error	An attempt was made to read from an empty queue buffer, or to write to a full queue buffer by advancing the pointer.
0008 hex	Queue table error	The specified table is not a queue table.
0009 hex	System error	An unexpected error was detected in the system during instruction execution.

Note: The error codes apply to all table manipulation instructions.

Programming Example

In the following programming example, the specified block in record table data TBL1 is moved to a continuous area that starts at the first address of the parameter table (MW00100) when switch 1 (DB000100) turns ON.

The parameter table is set as shown in the following table.

Register	Data	Remarks
DL00000	2	First row number of table elements
DL00002	2	First column number of table elements
DW00004	3	Number of row elements
DW00005	3	Number of column elements

The contents of table data TBL1 are given below.

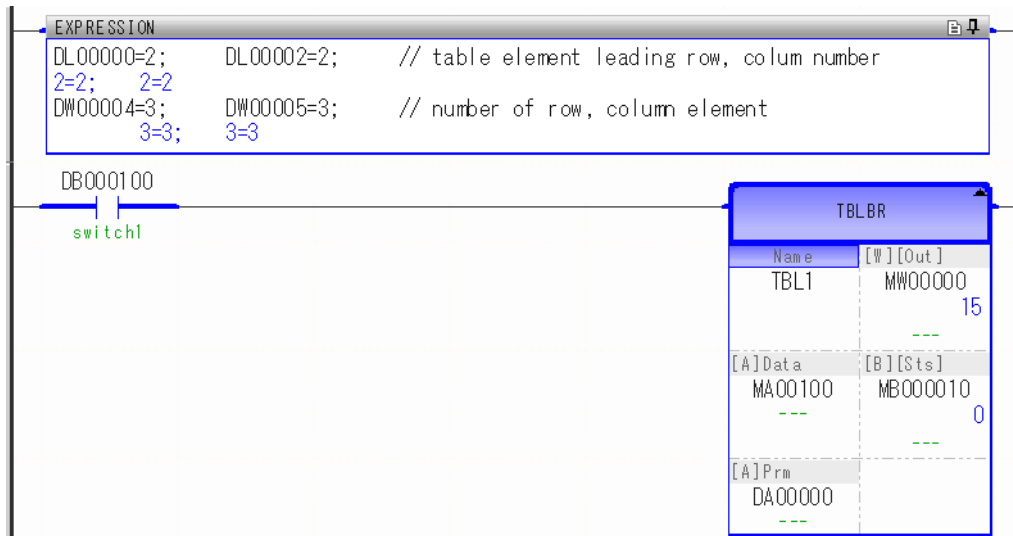
Row	Column				
	1 (W)*	2 (W)*	3 (L)*	4 (L)*	5 (F)*
1	1000	1001	10000	10001	1.1
2	2000	2002	20000	20002	1.2
3	3000	3003	30000	30003	1.3
4	4000	4004	40000	40004	1.4
5	5000	5005	50000	50005	1.5

Area to move

* Indicates the data type.

4.9 Table Manipulation Instructions

4.9.1 Read Table Block (TBLBR/TBLBRE)



After the instruction is executed, the data is moved to a continuous area that starts from MW00100 as shown below.

The number of words that was moved is set to 15 in MW00000 (output data), and MB000010 (status) is set to 0 (move successful).

Register	Data	Register	Data	Register	Data
MW00100	2002	ML00101	20000	ML00103	20002
MW00105	3003	ML00106	30000	ML00108	30003
MW00110	4004	ML00111	40000	ML00113	40004

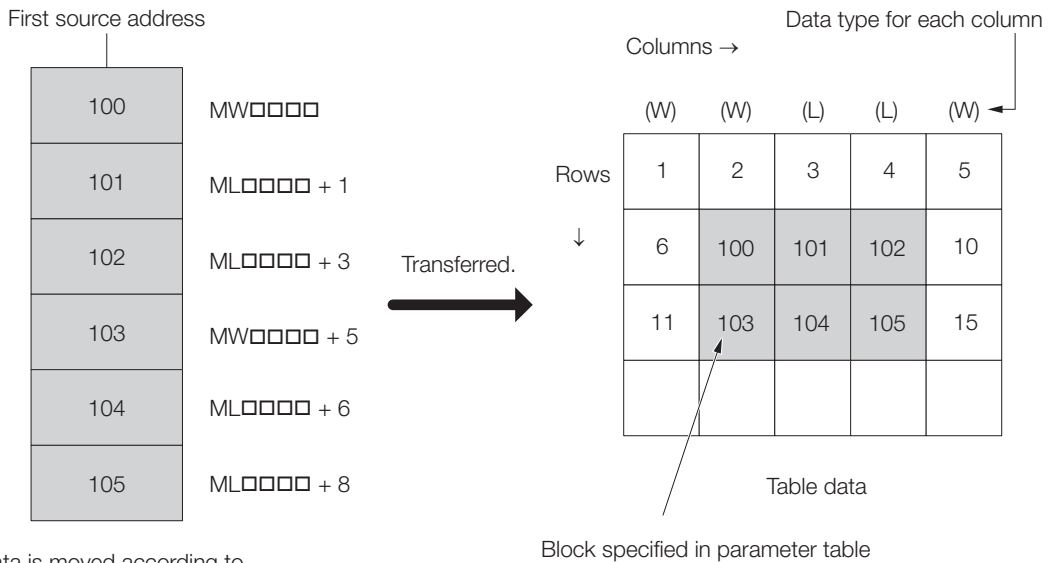
Note: The registers are assigned as shown in the above table.

4.9.2 Write Table Block (TBLBW/TBLBWE)

Data from a continuous area that starts at the first source address is moved to a block of table data that is specified by the table name, row number, and column number. The data is moved under the assumption that the data type of the source area and each element in the table data match.

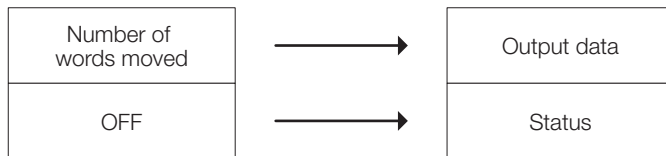
If an error occurs when accessing the table, such as data that is out of range or not enough data length at the source, an error is output and no data is written. The contents in the destination area will remain unchanged.

If the instruction ends normally, the number of words that were moved is output, and the status is turned OFF. If an error occurs, an error code is output and the status is turned ON.

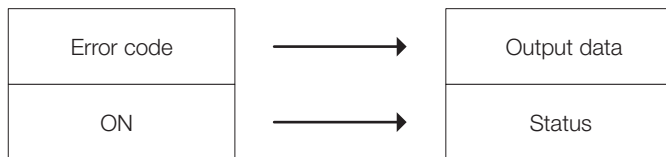


Data is moved according to the data type of the table data.

• **If the Move Succeeds**



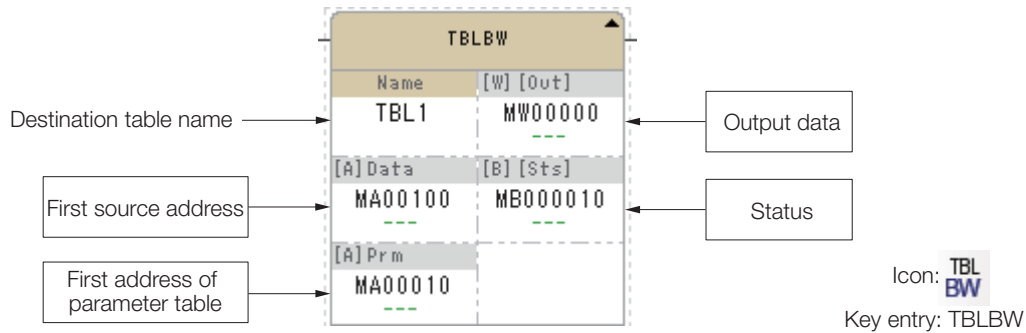
• **If the Move Fails**



Information If the move fails, the table data at the destination will retain the contents from before the instruction was executed.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Name (Table name)*1*2	×	×	×	×	×	×	○	×	×
Data (First source address)	×	×	×	×	×	×	○*4	×	×
Prm (First address of parameter table)	×	×	×	×	×	×	○	×	×
Out (Output data)*3	×	○*4	×	×	×	×	×	○	×
Sts (Status)*3	○*4	×	×	×	×	×	×	×	×

*1. Specify the registers in which the text string for the table name (8 bytes + NULL character max.) has been stored.

*2. G, M, D, or C register only.

*3. Optional.

*4. C and # registers cannot be used.

Details on I/O Items

Item	Name	I/O	Description
Name	Table name	IN	For the TBLBW instruction, directly enter the table name. For the TBLBWE instruction, indirectly designate the table name in registers.
Data	First source address	IN	Specify the first address of the source.
Prm	First address of parameter table	IN/OUT	Specify the first address of the table data.
Out	Output data	OUT	Specify the first address of the table data.
Sts	Status	OUT	Specify the address for checking the status of this instruction.



Important

Note the following precautions regarding Name.

- Always add a NULL character to Name. If a NULL character is not added to Name, a fixed length of 8 bytes (4 words) of data is read and handled as the table name. This is a risk that an unintended table name may be referenced because of this.
- When characters have been set with ASCII instructions, the NULL character is not set at the end of the text string. Use the STRSET instruction to set Name.
- An operation error will occur if the size from the first register to the maximum range of registers is less than 8 bytes (4 words).

◆ Parameter Table

Address	Data Type	Symbol	Name	Specification	I/O
0	L	ROW1	First row number of table elements	First row number of table elements to move (1 to 65,535)	IN
2	L	COL1	First column number of table elements	First column number of table elements to move (1 to 32,767)	IN
4	W	RLEN	Number of row elements	Number of row elements (1 to 32,767)	IN
5	W	CLEN	Number of column elements	Number of column elements (1 to 32,767)	IN

◆ Error Codes

Error Code	Error Name	Meaning
0001 hex	Table undefined	The target table is undefined.
0002 hex	Outside range of row numbers	The row number of the table element is outside the target table.
0003 hex	Outside range of column numbers	The column number of the table element is outside the target table.
0004 hex	Incorrect number of elements	The number of target elements is invalid.
0005 hex	Insufficient storage area	The storage area is insufficient.
0006 hex	Insufficient element type	The data type specified for the element is wrong.
0007 hex	Queue buffer error	An attempt was made to read from an empty queue buffer, or to write to a full queue buffer by advancing the pointer.
0008 hex	Queue table error	The specified table is not a queue table.
0009 hex	System error	An unexpected error was detected in the system during instruction execution.

Note: The error codes apply to all table manipulation instructions.

Programming Example

In the following programming example, a continuous area of data from the first address of the parameter table at MW00100 is moved to a specified block in record table data TBL1 when switch 1 (DB000100) turns ON.

The parameter table is set as shown in the following table.

Register	Data	Remarks
DL00000	2	First row number of table elements
DL00002	2	First column number of table elements
DW00004	3	Number of row elements
DW00005	3	Number of column elements

The data to move is given below.

Register	Data	Register	Data	Register	Data
MW00100	1	ML00101	2	ML00103	3
MW00105	4	ML00106	5	ML00108	6
MW00110	7	ML00111	8	ML00113	9

This table shows the contents of table data TBL1 after the instruction is executed.

The number of words that were moved is set to 15 in MW00000 (output data) and MB000010 (status) is set to 0 (move successful).

Row	Column				
	1 (W)*	2 (W)*	3 (L)*	4 (L)*	5 (F)*
1					
2		1	2	3	
3		4	5	6	
4		7	8	9	
5					

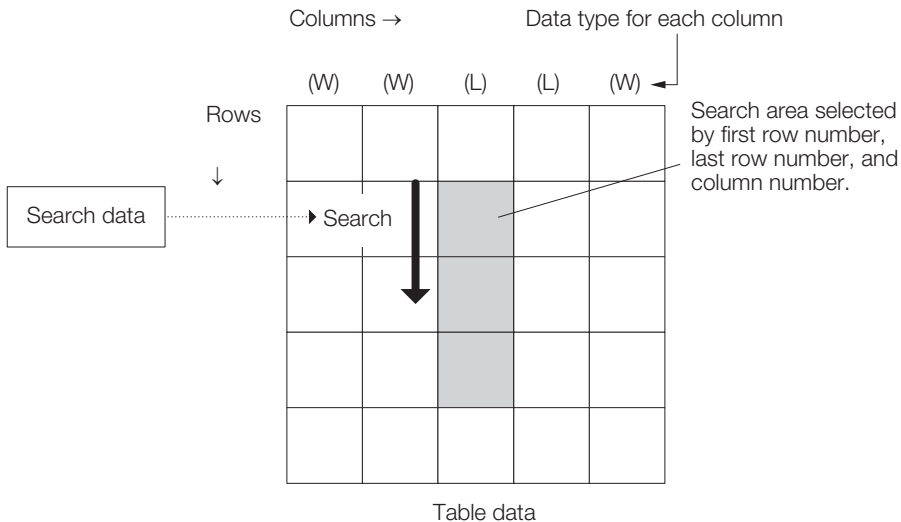
Moved area

* Indicates the data type.

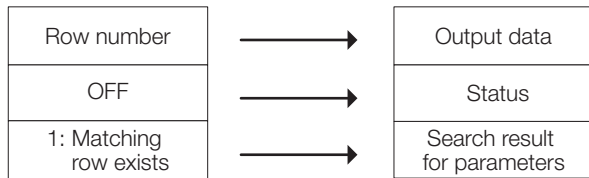
4.9.3 Search for Table Row (TBL SRL/TBL SRLE)

A search is made for the search data in column elements of the table data that is specified by the table name, row number, and column number. The search result is output as the row number of the data that matches the search data. The type of the data to be searched is automatically determined by the data type of the specified column elements.

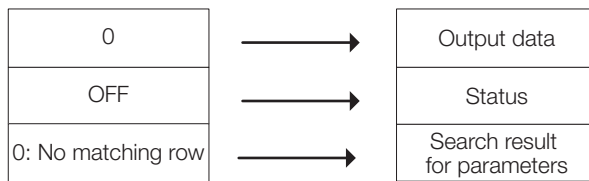
If the instruction ends normally and the search data is found, the search result in the input parameter table is set to 1, the output data is set to the row number, and the status is turned OFF. If the search data is not found, the search result and output data are set to 0. If an error occurs, an error code is set in the output data and the status is turned ON.



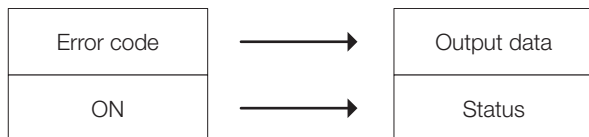
• **Search Data Found**



• **Search Data Not Found**

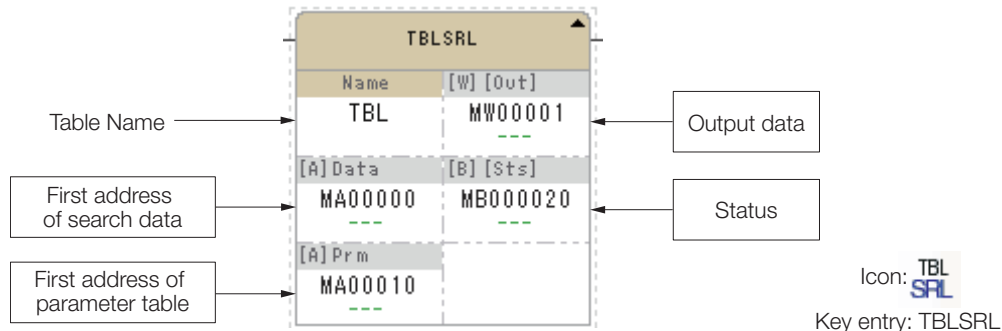


• **Search Error**



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Name (Table name)*1*2	×	×	×	×	×	×	○	×	×
Data (First address of search data)	×	×	×	×	×	×	○	×	×
Prm (First address of parameter table)	×	×	×	×	×	×	○	×	×
Out (Output data)*3	×	○*4	×	×	×	×	×	○	×
Sts (W) (Status)*3	○*4	×	×	×	×	×	×	×	×

- *1. Specify the registers in which the text string for the table name (8 bytes + NULL character max.) has been stored.
- *2. G, M, D, or C register only.
- *3. Optional.
- *4. C and # registers cannot be used.

Details on I/O Items

Item	Name	I/O	Description
Name	Table name	IN	For the TBL SRL instruction, directly enter the table name. For the TBL SRLE instruction, indirectly designate the table name in registers.
Data	First destination address	IN	Specify the first address of the destination.
Prm	First address of parameter table	IN/OUT	Specify the first address of the table data.
Out	Output data	OUT	Specify the first address of the table data.
Sts	Status	OUT	Specify the address for checking the status of this instruction.

Note the following precautions regarding Name.

- Always add a NULL character to Name. If a NULL character is not added to Name, a fixed length of 8 bytes (4 words) of data is read and handled as the table name. This is a risk that an unintended table name may be referenced because of this.
- When characters have been set with ASCII instructions, the NULL character is not set at the end of the text string. Use the STRSET instruction to set Name.
- An operation error will occur if the size from the first register to the maximum range of registers is less than 8 bytes (4 words).

◆ Parameter Table

Address	Data Type	Symbol	Name	Specification	I/O
0	L	ROW1	First row number of table elements	First row number of table elements to search (1 to 65,535)	IN
2	L	ROW2	Last row number of table elements	Last row number of table elements to search (1 to 65,535)	IN
4	L	COLUMN	Column number of table elements	Column number of table elements to search (1 to 32,767)	IN
6	W	FIND	Search result	Search result 0: No matching row 1: Matching row exists	OUT

◆ Error Codes

Error Code	Error Name	Meaning
0001 hex	Table undefined	The target table is undefined.
0002 hex	Outside range of row numbers	The row number of the table element is outside the target table.
0003 hex	Outside range of column numbers	The column number of the table element is outside the target table.
0004 hex	Incorrect number of elements	The number of target elements is invalid.
0005 hex	Insufficient storage area	The storage area is insufficient.
0006 hex	Insufficient element type	The data type specified for the element is wrong.
0007 hex	Queue buffer error	An attempt was made to read from an empty queue buffer, or to write to a full queue buffer by advancing the pointer.
0008 hex	Queue table error	The specified table is not a queue table.
0009 hex	System error	An unexpected error was detected in the system during instruction execution.

Programming Example

In the following programming example, a search is made by row for search data 32 in MW00000 from array table data TBL1.

The parameter table is set as shown in the following table.

Register	Data	Remarks
DL00010	2	First row number of table elements
DL00012	5	Last row number of table elements
DL00014	2	Column number of table elements

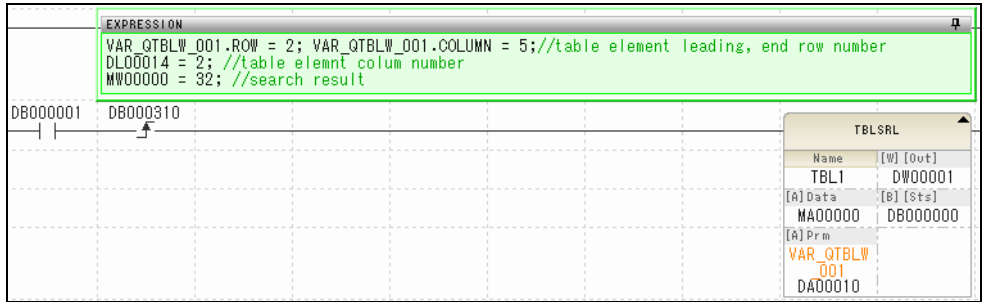
The contents of table data TBL1 are given below. (Table elements are integer data.)

Row	Column				
	1 (W)*	2 (W)*	3 (W)*	4 (W)*	5 (W)*
1	11	12	13	14	15
2	21	22	23	24	25
3	31	32	33	34	35
4	41	42	43	44	45
5	51	52	53	54	55

* Indicates the data type.

Area to search

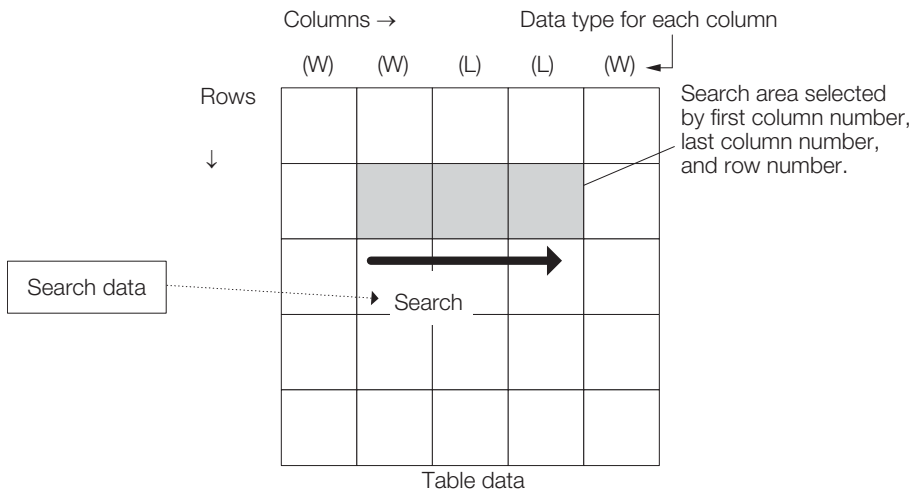
A match for 32 was found in row number 3 in the search area, so DW00001 (output data) is set to 3.



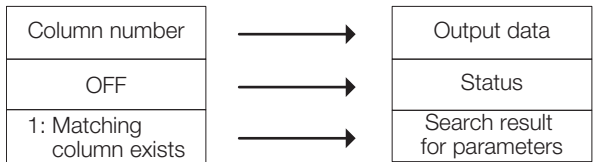
4.9.4 Search for Table Column (TBLSRC/TBLSRCE)

A search is made for the search data in row elements of the table data that is specified by the table name, row number, and column number. The search result is output as the column number of the data that matches the search data. The type of the data to be searched is automatically determined by the data type of the specified row elements.

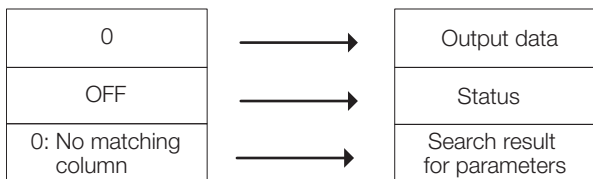
If the instruction ends normally and the search data is found, the search result in the input parameter table is set to 1, the output data is set to the column number, and the status is turned OFF. If the search data is not found, the search result and output data are set to 0. If an error occurs, an error code is set in the output data and the status is turned ON.



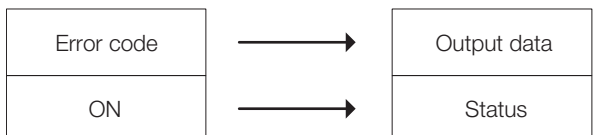
• Search Data Found



• Search Data Not Found

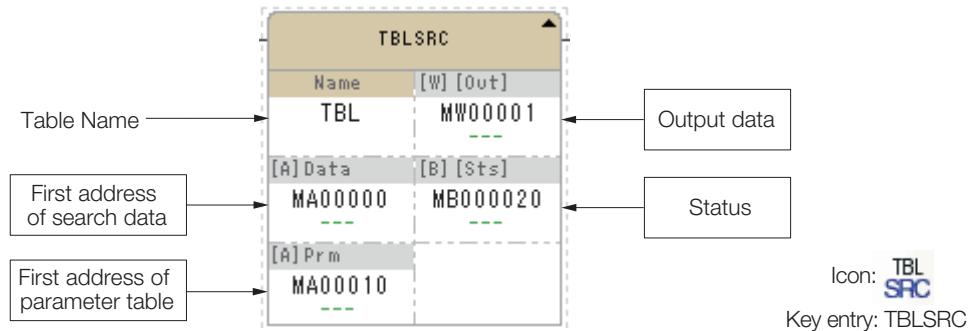


• Search Error



Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Name (Table name) ^{*1*2}	×	×	×	×	×	×	○	×	×
Data (First address of search data)	×	×	×	×	×	×	○	×	×
Prm (First address of parameter table)	×	×	×	×	×	×	○	×	×
Out (Output data) ^{*3}	×	○ ^{*4}	×	×	×	×	×	○	×
Sts (Status) ^{*3}	○ ^{*4}	×	×	×	×	×	×	×	×

*1. Specify the registers in which the text string for the table name (8 bytes + NULL character max.) has been stored.

*2. G, M, D, or C register only.

*3. Optional.

*4. C and # registers cannot be used.

Details on I/O Items

Item	Name	I/O	Description
Name	Table name	IN	For the TBLSRC instruction, directly enter the table name. For the TBLSRCE instruction, indirectly designate the table name in registers.
Data	First destination address	IN	Specify the first address of the destination.
Prm	First address of parameter table	IN/OUT	Specify the first address of the table data.
Out	Output data	OUT	Specify the first address of the table data.
Sts	Status	OUT	Specify the address for checking the status of this instruction.



Important

Note the following precautions regarding Name.

- Always add a NULL character to Name. If a NULL character is not added to Name, a fixed length of 8 bytes (4 words) of data is read and handled as the table name. This is a risk that an unintended table name may be referenced because of this.
- When characters have been set with ASCII instructions, the NULL character is not set at the end of the text string. Use the STRSET instruction to set Name.
- An operation error will occur if the size from the first register to the maximum range of registers is less than 8 bytes (4 words).

◆ Parameter Table

Address	Data Type	Symbol	Name	Specification	I/O
0	L	ROW1	Row number of table elements	Row number of table elements to search (1 to 65,535)	IN
2	L	COLUMN1	First column number of table elements	First column number of table elements to search (1 to 32,767)	IN
4	L	COLUMN2	Last column number of table elements	Last column number of table elements to search (1 to 32,767)	IN
6	W	FIND	Search result	Search result 0: No matching column 1: Matching column exists	OUT

◆ Error Codes

Error Code	Error Name	Meaning
0001 hex	Table undefined	The target table is undefined.
0002 hex	Outside range of row numbers	The row number of the table element is outside the target table.
0003 hex	Outside range of column numbers	The column number of the table element is outside the target table.
0004 hex	Incorrect number of elements	The number of target elements is invalid.
0005 hex	Insufficient storage area	The storage area is insufficient.
0006 hex	Unexpected element type	The data type specified for the element is wrong.
0007 hex	Queue buffer error	An attempt was made to read from an empty queue buffer, or to write to a full queue buffer by advancing the pointer.
0008 hex	Queue table error	The specified table is not a queue table.
0009 hex	System error	An unexpected error was detected in the system during instruction execution.

Note: The error codes apply to all table manipulation instructions.

Programming Example

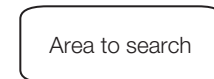
In the following programming example, a search is made by column for search data 34 in MW00000 from array table data TBL1.

The parameter table is set as shown in the following table.

Register	Data	Remarks
DL00010	3	Row number of table elements
DL00012	2	First column number of table elements
DL00014	5	Last column number of table elements

The contents of table data TBL1 are given below. (Table elements are integer data.)

Row	Column				
	1 (W)*	2 (W)*	3 (W)*	4 (W)*	5 (W)*
1	11	12	13	14	15
2	21	22	23	24	25
3	31	32	33	34	35
4	41	42	43	44	45
5	51	52	53	54	55



* Indicates the data type.

A match for 34 was found in column number 4 in the search area, so DW00001 (output data) is set to 4.

The EXPRESSION window contains the following code:

```

DL00010=3;           // table element row number;;
3=3
DL00012=2;   DL00014=5; // table element leading, end column number
                2=2;     5=5
MW00000=34;      // search result
                34=34
    
```

The TBLSRC window shows the following table structure:

TBLSRC	
Name	[W] [Out]
TBL1	DW00001
	4
[A] Data	[B] [Sts]
MA00000	DB000000
	0
[A] Pfm	
DA00010	

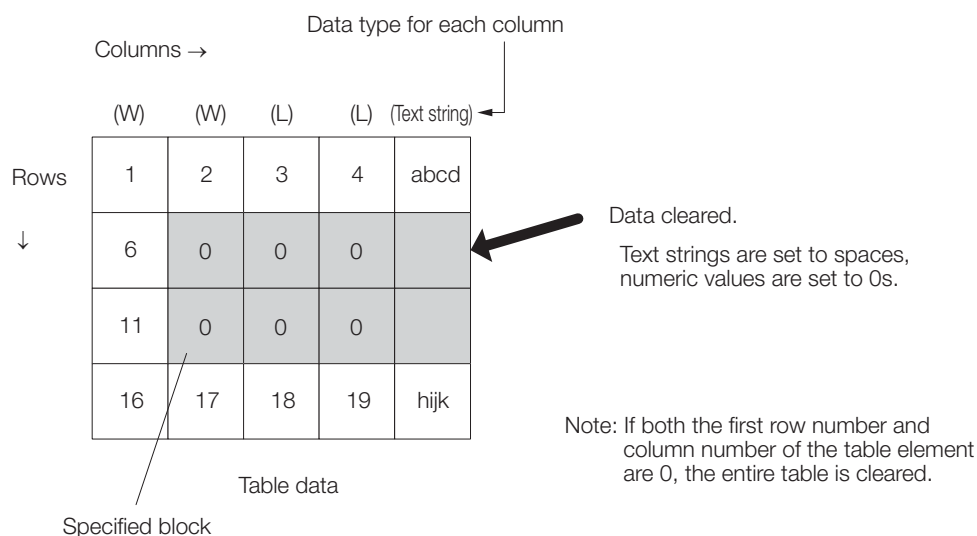
4.9.5 Clear Table Block (TBLCL/TBLCLE)

A block of data in the table data that is specified by the table name, row number, and column number is cleared. The table elements are filled with spaces if the data type is for text strings, and 0s if the data type is for numeric values.

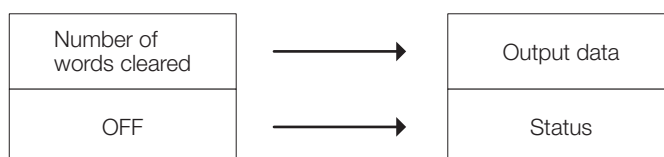
If both the first row number and the first column number of the table element are 0, the entire table will be cleared.

If an error occurs when accessing the table, such as data that is out of range or not enough data length at the destination, an error is output and no data is written.

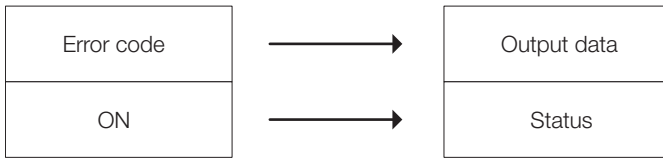
If the instruction ends normally, the number of words that were cleared is output and the status is turned OFF. If an error occurs, an error code is set in the output data and the status is turned ON.



• If the Clear Succeeds



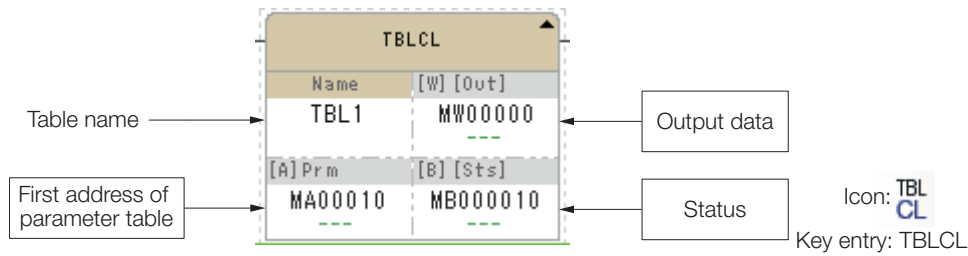
• If the Clear Fails



Information If the clear fails, the table data will retain the contents from before the instruction was executed.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Name (Table name) ^{*1*2}	×	×	×	×	×	×	○	×	×
Prm (First address of parameter table)	×	×	×	×	×	×	○	×	×
Out (Output data) ^{*3}	×	○ ^{*4}	×	×	×	×	×	○	×
Sts (Status) ^{*3}	○ ^{*4}	×	×	×	×	×	×	×	×

*1. Specify the registers in which the text string for the table name (8 bytes + NULL character max.) has been stored.

*2. G, M, D, or C register only.

*3. Optional.

*4. C and # registers cannot be used.

Details on I/O Items

Item	Name	I/O	Description
Name	Table name	IN	For the TBLCL instruction, directly enter the table name. For the TBLCLE instruction, indirectly designate the table name in registers.
Prm	First address of parameter table	IN/OUT	Specify the first address of the table data.
Out	Output data	OUT	Specify the first address of the table data.
Sts	Status	OUT	Specify the address for checking the status of this instruction.

Important

Note the following precautions regarding Name.

- Always add a NULL character to Name. If a NULL character is not added to Name, a fixed length of 8 bytes (4 words) of data is read and handled as the table name. This is a risk that an unintended table name may be referenced because of this.
- When characters have been set with ASCII instructions, the NULL character is not set at the end of the text string. Use the STRSET instruction to set Name.
- An operation error will occur if the size from the first register to the maximum range of registers is less than 8 bytes (4 words).

◆ Parameter Table

Address	Data Type	Symbol	Name	Specification	I/O
0	L	ROW	First row number of table elements	First row number of table elements to move (1 to 65,535)	IN
2	L	COL	First column number of table elements	First column number of table elements to move (1 to 32,767)	IN
4	W	RLEN	Number of row elements	Number of row elements (1 to 32,767)	IN
5	W	CLEN	Number of column elements	Number of column elements (1 to 32,767)	IN

◆ Error Codes

Error Code	Error Name	Meaning
0001 hex	Table undefined	The target table is undefined.
0002 hex	Outside range of row numbers	The row number of the table element is outside the target table.
0003 hex	Outside range of column numbers	The column number of the table element is outside the target table.
0004 hex	Incorrect number of elements	The number of target elements is invalid.
0005 hex	Insufficient storage area	The storage area is insufficient.
0006 hex	Insufficient element type	The data type specified for the element is wrong.
0007 hex	Queue buffer error	An attempt was made to read from an empty queue buffer, or to write to a full queue buffer by advancing the pointer.
0008 hex	Queue table error	The specified table is not a queue table.
0009 hex	System error	An unexpected error was detected in the system during instruction execution.

Note: The error codes apply to all table manipulation instructions.

Programming Example

In the following programming example, the specified block is cleared from record table data TBL1 when switch 1 (DB000100) turns ON.

The parameter table is set as shown in the following table.

Register	Data	Remarks
DL00000	2	First row number of table elements
DL00002	2	First column number of table elements
DW00004	3	Number of row elements
DW00005	3	Number of column elements

The contents of table data TBL1 are given below.

Row	Column				
	1 (W)*	2 (W)*	3 (L)*	4 (Text string)*	5 (F)*
1	1000	1001	10000	ABCD	1.1
2	2000	2002	20000	BCDE	1.2
3	3000	3003	30000	CDEF	1.3
4	4000	4004	40000	DEFG	1.4
5	5000	5005	50000	EFGH	1.5

Area to clear

* Indicates the data type.

4.9 Table Manipulation Instructions

4.9.6 Move Table Block (TBLMV/TBLMVE)

The EXPRESSION window contains the following code:

```
DL00000=2; DL00002=2; // table element leading row, column number
2=2; 2=2
DW00004=3; // table element row number
3=3
DW00005=3; // table element column number
3=3
```

The TBLCL window shows the following data:

Name	[W]	[Out]
TBL1	MW00000	15
[A]Prm	DA00000	[B] [Sts]
		MB000010
		0

The data is cleared after the instruction is executed as shown below.

Row	Column				
	1 (W)*	2 (W)*	3 (L)*	4 (Text string)*	5 (F)*
1	1000	1001	10000	ABCD	1.1
2	2000	0	0		1.2
3	3000	0	0		1.3
4	4000	0	0		1.4
5	5000	5005	50000	EFGH	1.5

A red box highlights the area from row 2 to row 4, columns 2 to 4. A callout bubble points to this area with the text "Area cleared".

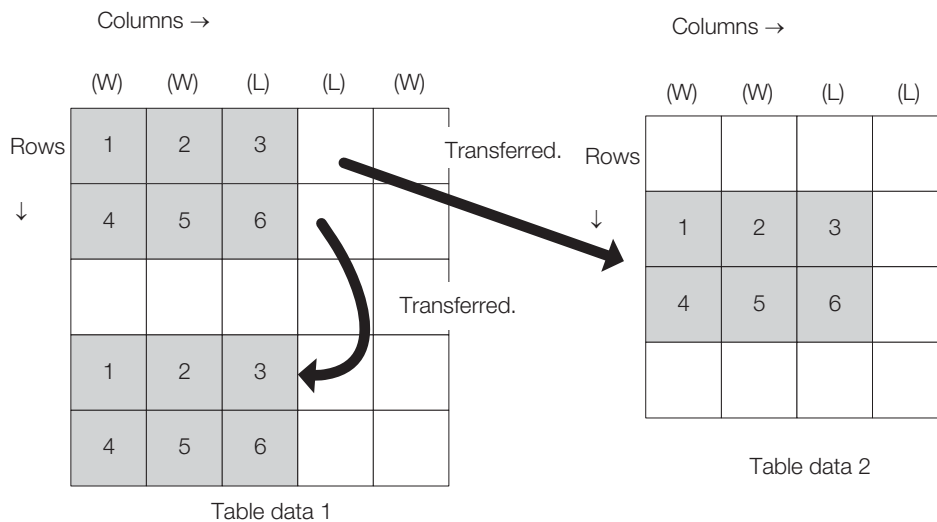
* Indicates the data type.

4.9.6 Move Table Block (TBLMV/TBLMVE)

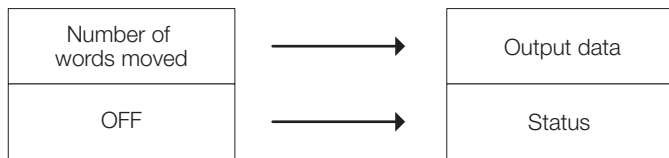
A block of data in the table data that is specified by the table name, row number, and column number is moved to a different block in the table. The block can be moved between different tables or within the same table.

If the data type of the column elements in the source and destination do not match, an error is output and no data is moved.

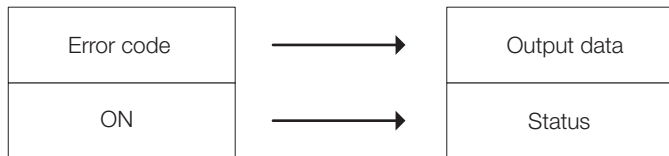
If the instruction ends normally, the number of words that were moved is output, and the status is turned OFF. If an error occurs, an error code is output and the status is turned ON.



• If the Move Succeeds



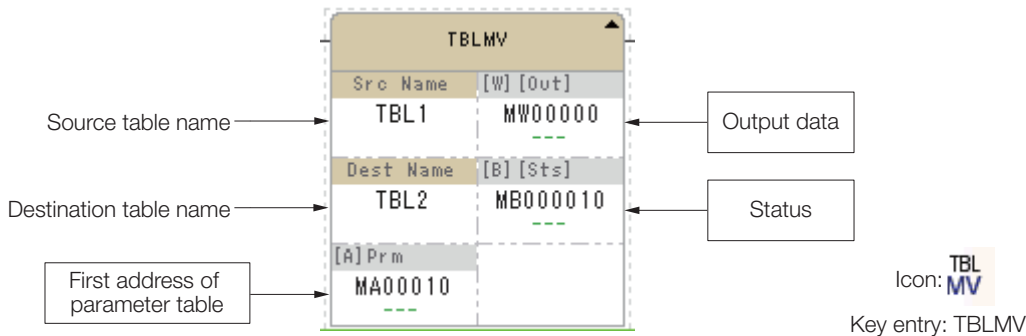
• If the Move Fails



Information If the move fails, the table data will retain the contents from before the instruction was executed.

Format

The format of this instruction is shown below.




I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src Name*1*2	×	×	×	×	×	×	○	×	×
Dest Name*1*2	×	×	×	×	×	×	○	×	×
Prm (First address of parameter table)	×	×	×	×	×	×	○	×	×
Out (Output data)*3	×	○*4	×	×	×	×	×	○	×
Sts (Status)*3	○*4	×	×	×	×	×	×	×	×

*1. Specify the registers in which the text string for the table name (8 bytes + NULL character max.) has been stored.
 *2. G, M, D, or C register only.
 *3. Optional.
 *4. C and # registers cannot be used.

Details on I/O Items

Item	Name	I/O	Description
Src Name	Table name	IN	For the TBLMV instruction, directly enter the table name. For the TBLMVE instruction, indirectly designate the table name in registers.
Dest Name			
Prm	First address of parameter table	IN/OUT	Specify the first address of the table data.
Out	Output data	OUT	Specify the first address of the table data.
Sts	Status	OUT	Specify the address for checking the status of this instruction.



Note the following precautions regarding Name.

- Always add a NULL character to Name. If a NULL character is not added to Name, a fixed length of 8 bytes (4 words) of data is read and handled as the table name. This is a risk that an unintended table name may be referenced because of this.
- When characters have been set with ASCII instructions, the NULL character is not set at the end of the text string. Use the STRSET instruction to set Name.
- An operation error will occur if the size from the first register to the maximum range of registers is less than 8 bytes (4 words).

◆ Parameter Table

Address	Data Type	Symbol	Name	Specification	I/O
0	L	ROW1	First row number of table elements	First row number of table elements at source to move (1 to 65,535)	IN
2	L	COLUMN1	First column number of table elements	First column number of table elements at source to move (1 to 32,767)	IN
4	W	RLEN	Number of row elements	Number of row elements (1 to 32,767)	IN
5	W	CLEN	Number of column elements	Number of column elements (1 to 32,767)	IN
6	L	ROW2	First row number of table elements	First row number of table elements at destination to move (1 to 65,535)	IN
8	L	COLUMN2	First column number of table elements	First column number of table elements at destination to move (1 to 32,767)	IN

◆ Error Codes

Error Code	Error Name	Meaning
0001 hex	Table undefined	The target table is undefined.
0002 hex	Outside range of row numbers	The row number of the table element is outside the target table.
0003 hex	Outside range of column numbers	The column number of the table element is outside the target table.
0004 hex	Incorrect number of elements	The number of target elements is invalid.
0005 hex	Insufficient storage area	The storage area is insufficient.
0006 hex	Insufficient element type	The data type specified for the element is wrong.
0007 hex	Queue buffer error	An attempt was made to read from an empty queue buffer, or to write to a full queue buffer by advancing the pointer.
0008 hex	Queue table error	The specified table is not a queue table.
0009 hex	System error	An unexpected error was detected in the system during instruction execution.


Note: The error codes apply to all table manipulation instructions.

Programming Example

In the following programming example, the specified block in record table data TBL1 is moved to the specified block in table data TBL2 when switch 1 (DB000100) turns ON.

The contents of table data TBL1 are given below.

Row	Column		
	1 (W)*	2 (W)*	3 (L)*
1	1000	1001	10000
2	2000	2002	20000
3	3000	3003	30000
4	4000	4004	40000
5	5000	5005	50000



Area to move

* Indicates the data type.

The parameter table is set as shown in the following table.

Register	Data	Remarks
DL00000	2	First row number at source
DL00002	1	First column number at source
DW00004	3	Number of row elements
DW00005	3	Number of column elements
DL00006	2	First row number at destination
DL00008	2	First column number at destination

The contents of table data TBL2 are given below.

Row	Column				
	1 (W)*	2 (W)*	3 (W)*	4 (W)*	5 (W)*
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

Area to move

* Indicates the data type.

The screenshot shows a programming environment with two windows. The top window, titled 'EXPRESSION', contains the following code:

```

DL00000=2;    DL00002=1;    // table element leading row, column number
2=2;    1=1;
DW00004=3;    // table element row number;;
3=3;
DW00005=3;    // table element column number;;
3=3;
DL00006=2;    DL00008=2;    // destination table element leading row, column number
2=2;    2=2;
    
```

The bottom window shows a 'switch1' instruction and a 'TBLMV' instruction window. The TBLMV window displays the following parameters:

Src Name	[W][Out]
TBL1	MM00012
Dest Name	[B][Sts]
TBL2	MB00010
[A]Prm	DA00000

This table shows the contents of table data TBL2 after the instruction is executed.

Row	Column				
	1 (W)*	2 (W)*	3 (W)*	4 (W)*	5 (W)*
1	0	0	0	0	0
2	0	2000	2002	20000	0
3	0	3000	3003	30000	0
4	0	4000	4004	40000	0
5	0	0	0	0	0

Moved area

* Indicates the data type.

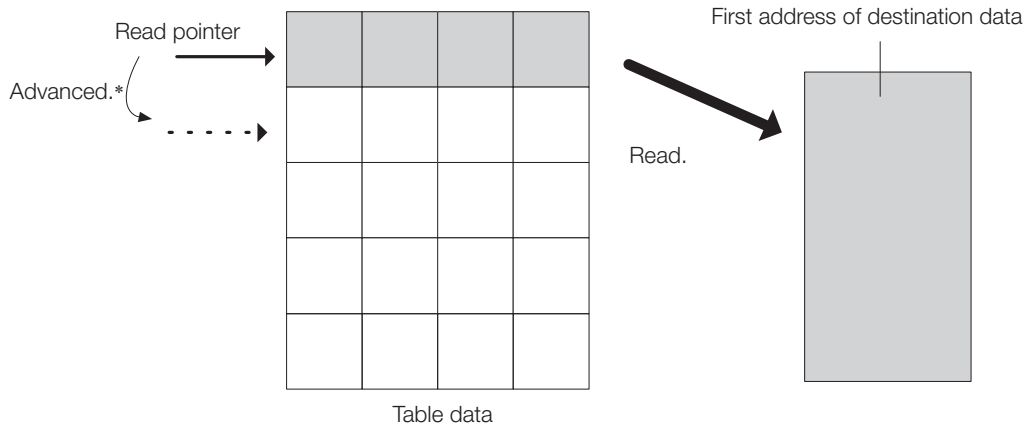
4.9.7 Read Queue Table (QTBLR/QTBLRE and QTBLRI/QTBLRIE)

Column elements of table data that are specified by the table name, row number, and column number are continuously read and stored in a continuous area that starts at a specified register. The data type of the elements read is automatically determined by the table that is specified. The data type of the destination register is ignored and the data is stored according to the data type of the table without any conversion.

The QTBLR/QTBLRE instruction does not change the queue table read pointer. The QTBLRI/QTBLRIE instruction advances the queue table read pointer by one row.

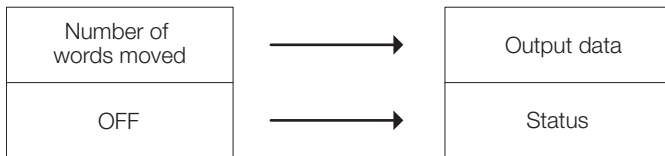
If an error occurs when accessing the table, such as a table name error, an out of range row number, or an empty queue buffer, an error is output, no data is read, and the pointer is not advanced. The contents of the destination register will be retained.

If the instruction ends normally, the number of words that were moved is output, and the status is turned OFF. If an error occurs, an error code is output and the status is turned ON.

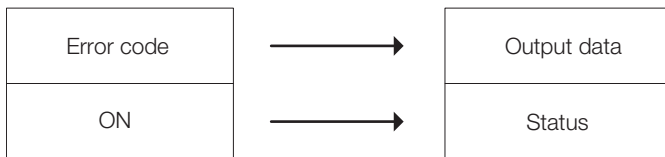


* The pointer is not advanced after executing the QTBLR/QTBLRE instruction. The pointer is advanced after executing the QTBLRI/QTBLRIE instruction.

• **If the Read Succeeds**



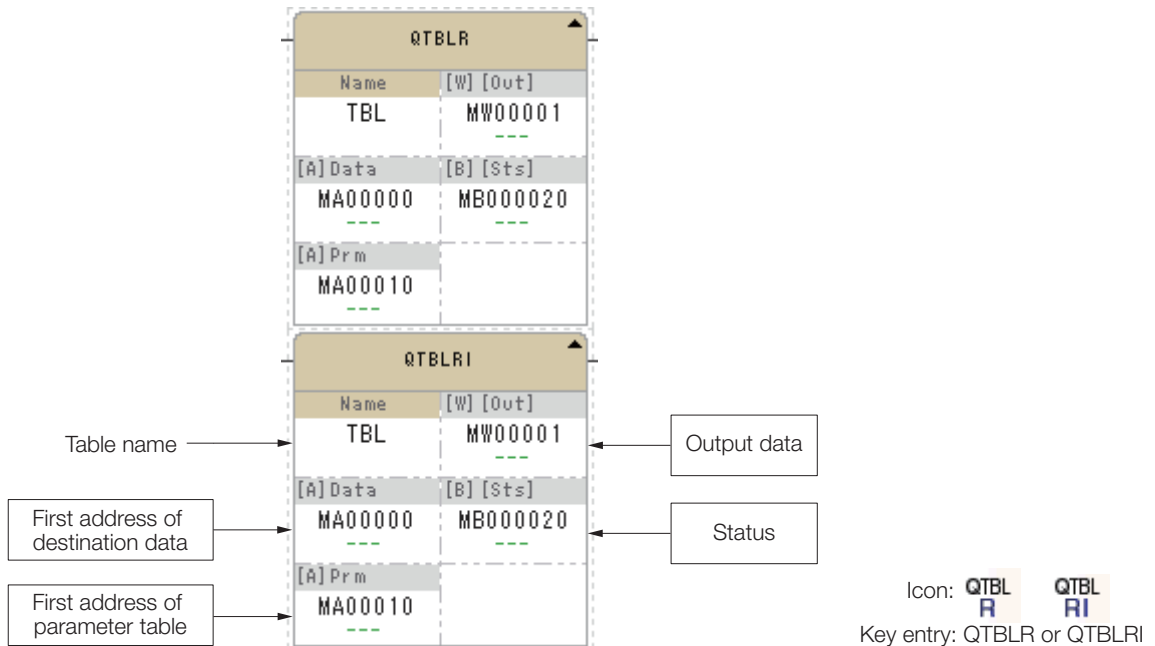
• **If the Read Fails**



Information If the read fails, the data at the destination will retain the contents from before the instruction was executed.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Name (Table name)*1*2	×	×	×	×	×	×	○	×	×
Data (First address of destination data)	×	×	×	×	×	×	○	×	×
Prm (First address of parameter table)	×	×	×	×	×	×	○*4	×	×
Out (Output data)*3	×	○*4	×	×	×	×	×	○	×
Sts (Status)*3	○*4	×	×	×	×	×	×	×	×

*1. Specify the registers in which the text string for the table name (8 bytes + NULL character max.) has been stored.

*2. G, M, D, or C register only.

*3. Optional.

*4. C and # registers cannot be used.

Details on I/O Items

Item	Name	I/O	Description
Name	Table name	IN	For the QTBLR and QTBLRI instructions, directly enter the table name. For the QTBLRE and QTBLRIE instructions, indirectly designate the table name in registers.
Data	First destination address	IN	Specify the first address of the destination.
Prm	First address of parameter table	IN/OUT	Specify the first address of the table data.
Out	Output data	OUT	Specify the first address of the table data.
Sts	Status	OUT	Specify the address for checking the status of this instruction.



Important

Note the following precautions regarding Name.

- Always add a NULL character to Name. If a NULL character is not added to Name, a fixed length of 8 bytes (4 words) of data is read and handled as the table name. This is a risk that an unintended table name may be referenced because of this.
- When characters have been set with ASCII instructions, the NULL character is not set at the end of the text string. Use the STRSET instruction to set Name.
- An operation error will occur if the size from the first register to the maximum range of registers is less than 8 bytes (4 words).

◆ Parameter Table

Address	Data Type	Symbol	Name	Specification	I/O
0	L	ROW	Relative row number of table elements	Relative row number of table elements at source to move (1 to 65,535)	IN
2	L	COLUMN	First column number of table elements	First column number of table elements at source to move (1 to 32,767)	IN
4	W	CLEN	Number of column elements	Number of column elements to move (1 to 32,767)	IN
5	W	Reserved.			
6	L	RPTR	Read pointer	Read pointer of the queue after execution	OUT
8	L	WPTR	Write pointer	Write pointer of the queue after execution	OUT

◆ Error Codes

Error Code	Error Name	Meaning
0001 hex	Table undefined	The target table is undefined.
0002 hex	Outside range of row numbers	The row number of the table element is outside the target table.
0003 hex	Outside range of column numbers	The column number of the table element is outside the target table.
0004 hex	Incorrect number of elements	The number of target elements is invalid.
0005 hex	Insufficient storage area	The storage area is insufficient.
0006 hex	Insufficient element type	The data type specified for the element is wrong.
0007 hex	Queue buffer error	An attempt was made to read from an empty queue buffer, or to write to a full queue buffer by advancing the pointer.
0008 hex	Queue table error	The specified table is not a queue table.
0009 hex	System error	An unexpected error was detected in the system during instruction execution.

Note: The error codes apply to all table manipulation instructions.


◆ Setting the Relative Row Number of Table Elements

Relative Row Number	Read Row	Remarks
0	Read pointer row	The pointer advances only for the QTBLRI instruction.
1	Read pointer row	Pointer is not advanced.
2	Read pointer row - 1	Pointer is not advanced.
3	Read pointer row - 2	Pointer is not advanced.
:	:	:
n	Read pointer row - (n - 1)	Pointer is not advanced.

Programming Example

In the following programming example, the specified column elements in array table data TBL1 are read from the MW00010 area to the MW00012 area when switch 2 (DB000002) turns ON.

Before switch 2 is turned ON, the table data is set as shown below by turning ON switch 1 three times while the value is changed from MW00010 to MW00012. Refer to the following section for details.

 4.9.8 Write Queue Table (QTBLW/QTBLWE and QTBLWI/QTBLWIE) on page 4-198 – Programming Example on page 4-201

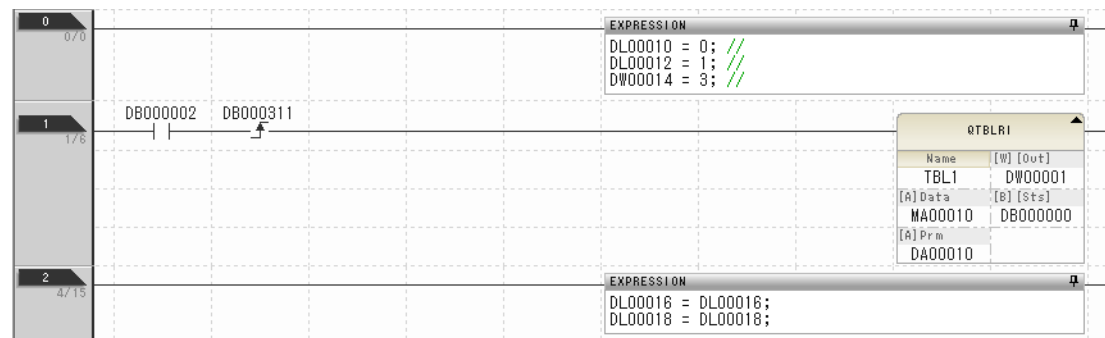
The contents of table data TBL1 are given below.

Row	Column		
	1 (W)*	2 (W)*	3 (W)*
1	11	12	13
2	21	22	23
3	31	32	33

* Indicates the data type.

The parameter table is set as shown in the following table.

Register	Data	Remarks
DL00010	0	Relative row number of table elements
DL00012	1	First column number of table elements
DW00014	3	Number of column elements



The data that is read changes each time switch 2 (DB000002) turns ON, from the first time to the third time, as shown below.

Register	1st Data	2nd Data	3rd Data
MW00010	11	21	31
MW00011	12	22	32
MW00012	13	23	33

The read pointer is advanced each time the instruction is executed starting at the first row on the first pass, the second row on the second pass, and so on, therefore resulting in the table shown above.

Information When the power is turned ON, the data pointed to by the read pointer and write pointer is undefined. Always execute the QTBLCL/QTBLCLE instruction before using the QTBLR/QTBLRE, QTBLRI/QTBLRIE, QTBLW/QTBLWE, or QTBLWI/QTBLWIE instruction. An operation error may occur if the QTBLR/QTBLRE, QTBLRI/QTBLRIE, QTBLW/QTBLWE, or QTBLWI/QTBLWIE instruction is executed without executing the QTBLCL/QTBLCLE instruction first.

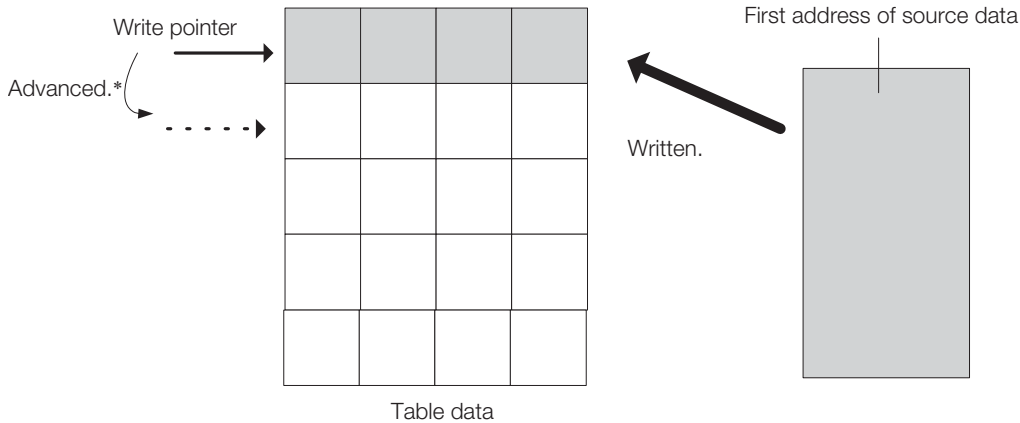
4.9.8 Write Queue Table (QTBLW/QTBLWE and QTBLWI/QTBLWIE)

Data in a continuous area that starts at a specified register is continuously written to columns in a specified table data. The instruction is processed under the assumption that the data type of the source and destination are the same.

The QTBLW/QTBLWE instruction does not change the queue table write pointer. The QTBLWI/QTBLWIE instruction advances the queue table write pointer by one row.

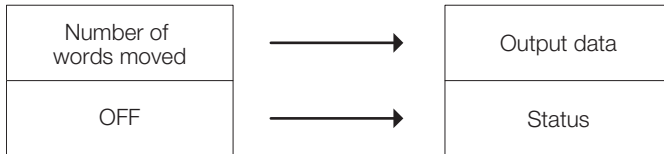
If an error occurs when accessing the table, such as a table name error, an out of range row number, or a full queue buffer, an error is output, no data is written, and the pointer is not advanced. The contents of the destination register will be retained.

If the instruction ends normally, the number of words that were moved is output, and the status is turned OFF. If an error occurs, an error code is output and the status is turned ON.

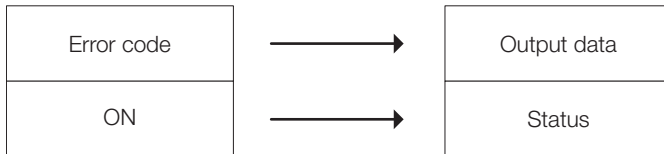


* The pointer is not advanced after executing the QTBLW/QTBLWE instruction. The pointer is advanced after executing the QTBLWI/QTBLWIE instruction.

• **If the Write Succeeds**



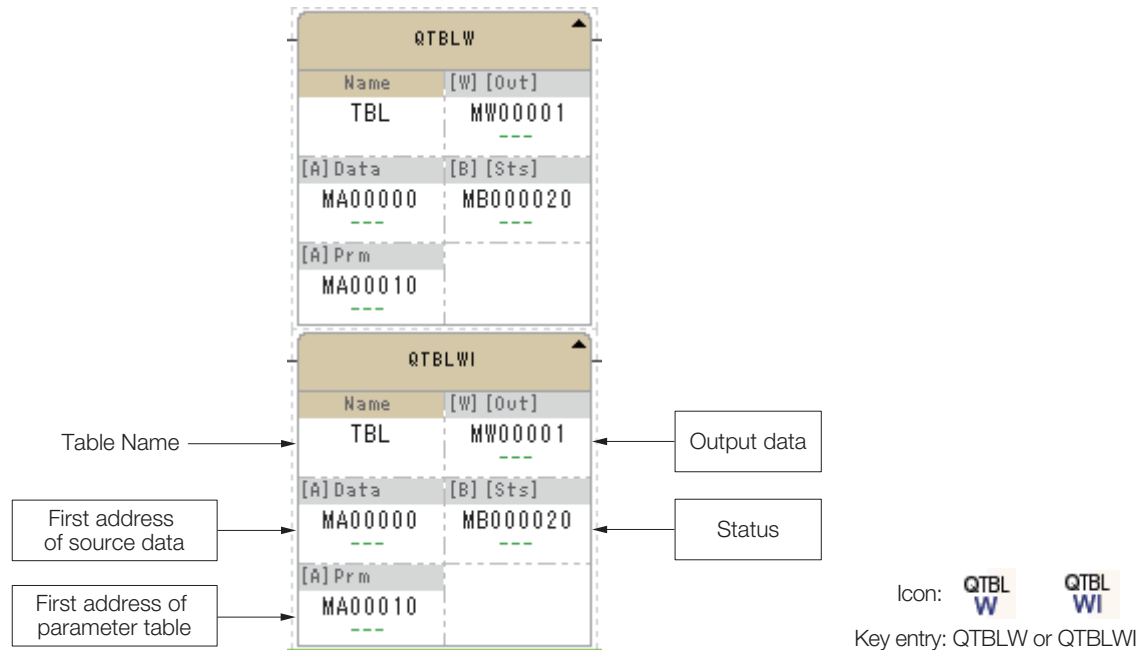
• **If the Write Fails**



Information If the write fails, the table data will retain the contents from before the instruction was executed.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Name (Table name) ^{*1*2}	×	×	×	×	×	×	○	×	×
Data (First address of source data)	×	×	×	×	×	×	○	×	×
Prm (First address of parameter table)	×	×	×	×	×	×	○ ^{*4}	×	×
Out (Output data) ^{*3}	×	○ ^{*4}	×	×	×	×	×	○	×
Sts (Status) ^{*3}	○ ^{*4}	×	×	×	×	×	×	×	×

*1. Specify the registers in which the text string for the table name (8 bytes + NULL character max.) has been stored.

*2. G, M, D, or C register only.

*3. Optional.

*4. C and # registers cannot be used.

Details on I/O Items

Item	Name	I/O	Description
Name	Table name	IN	For the QTBLW and QTBLWI instructions, directly enter the table name. For the QTBLWE and QTBLWIE instructions, indirectly designate the table name in registers.
Data	First destination address	IN	Specify the first address of the destination.
Prm	First address of parameter table	IN/OUT	Specify the first address of the table data.
Out	Output data	OUT	Specify the first address of the table data.
Sts	Status	OUT	Specify the address for checking the status of this instruction.



Note the following precautions regarding Name.

- Always add a NULL character to Name. If a NULL character is not added to Name, a fixed length of 8 bytes (4 words) of data is read and handled as the table name. This is a risk that an unintended table name may be referenced because of this.
- When characters have been set with ASCII instructions, the NULL character is not set at the end of the text string. Use the STRSET instruction to set Name.
- An operation error will occur if the size from the first register to the maximum range of registers is less than 8 bytes (4 words).

◆ Parameter Table

Address	Data Type	Symbol	Name	Specification	I/O
0	L	ROW	Relative row number of table elements	Relative row number of table elements at destination (1 to 65,535)	IN
2	L	COLUMN	First column number of table elements	First column number of table elements at destination (1 to 32,767)	IN
4	W	CLEN	Number of column elements	Number of column elements to move (1 to 32,767)	IN
5	W	Reserved.			
6	L	RPTR	Read pointer	Read pointer of the queue after execution	OUT
8	L	WPTR	Write pointer	Write pointer of the queue after execution	OUT

◆ Error Codes

Error Code	Error Name	Meaning
0001 hex	Table undefined	The target table is undefined.
0002 hex	Outside range of row numbers	The row number of the table element is outside the target table.
0003 hex	Outside range of column numbers	The column number of the table element is outside the target table.
0004 hex	Incorrect number of elements	The number of target elements is invalid.
0005 hex	Insufficient storage area	The storage area is insufficient.
0006 hex	Insufficient element type	The data type specified for the element is wrong.
0007 hex	Queue buffer error	An attempt was made to read from an empty queue buffer, or to write to a full queue buffer by advancing the pointer.
0008 hex	Queue table error	The specified table is not a queue table.
0009 hex	System error	An unexpected error was detected in the system during instruction execution.

Note: The error codes apply to all table manipulation instructions.

◆ Setting the Relative Row Number of Table Elements

Relative Row Number	Row Write	Remarks
0	Write pointer row	The pointer advances only for the QTBLWI instruction.
1	Write pointer row	Pointer is not advanced.
2	Write pointer row - 1	Pointer is not advanced.
3	Write pointer row - 2	Pointer is not advanced.
:	:	:
n	Write pointer row - (n - 1)	Pointer is not advanced.

Programming Example

In the following programming example, the data from MW00010 to MW00012 is written to the specified column elements in array table data TBL1 when switch 1 (DB000001) turns ON.

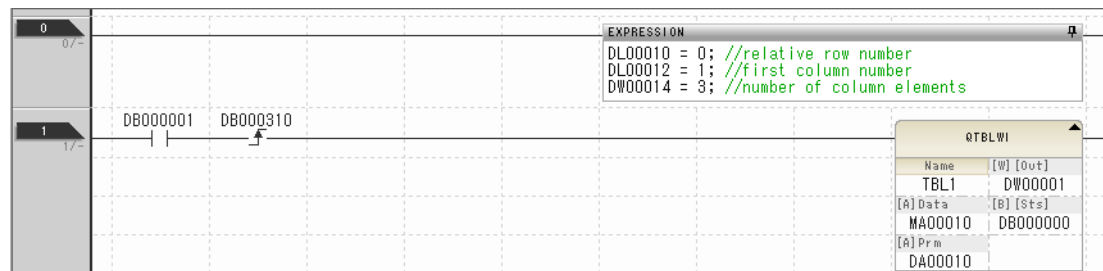
Initialize table data TBL1 before executing this type of programming.

Row	Column		
	1 (W)*	2 (W)*	3 (W)*
1	0	0	0
2	0	0	0
3	0	0	0

* Indicates the data type.

The parameter table is set as shown in the following table.

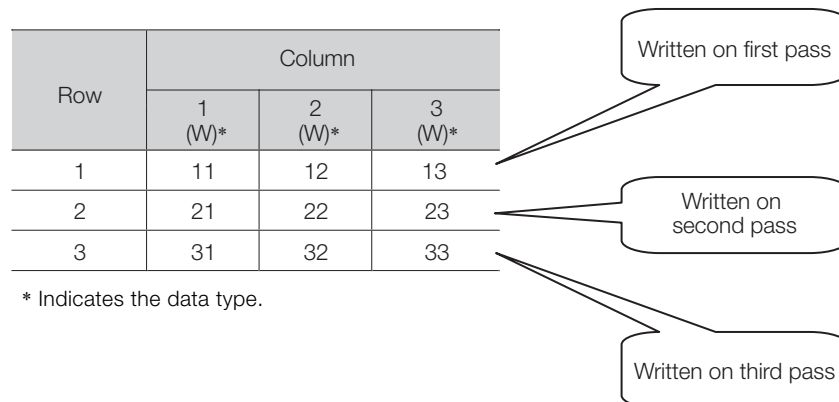
Register	Data	Remarks
DL00010	0	Relative row number of table elements
DL00012	1	First column number of table elements
DW00014	3	Number of column elements



Switch 1 (DB000001) is turned ON three time while the data is changed from MW00010 to MW00012, as shown below.

Register	1st Data	2nd Data	3rd Data
MW00010	11	21	31
MW00011	12	22	32
MW00012	13	23	33

The write pointer is advanced each time the instruction is executed starting at the first row on the first pass, the second row on the second pass, and so on. After three executions, TBL1 will be set with data as shown below.



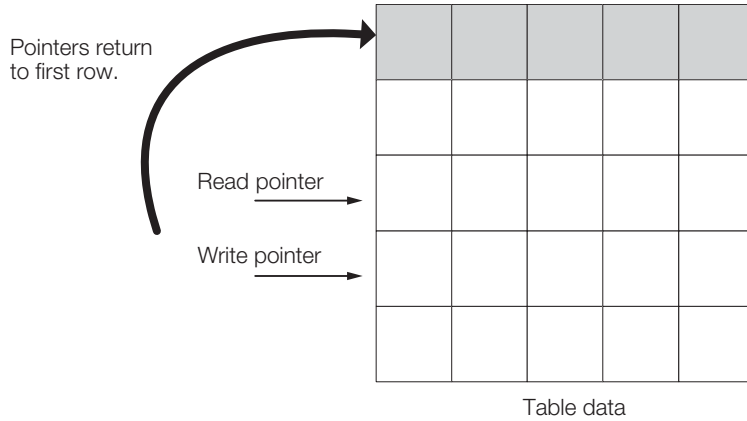
* Indicates the data type.

Information When the power is turned ON, the data pointed to by the read pointer and write pointer is undefined. Always execute the QTBLCL/QTBLCLE instruction before using the QTBLR/QTBLRE, QTBLRI/QTBLRIE, QTBLW/QTBLWE, or QTBLWI/QTBLWIE instruction. An operation error may occur if the QTBLR/QTBLRE, QTBLRI/QTBLRIE, QTBLW/QTBLWE, or QTBLWI/QTBLWIE instruction is executed without executing the QTBLCL/QTBLCLE instruction first.

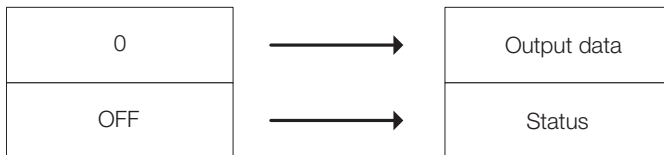
4.9.9 Clear Queue Table Pointer (QTBLCL/QTBLCLE)

The queue read and queue write pointers are returned to their initial state (first row) for the table data that is specified by the table name.

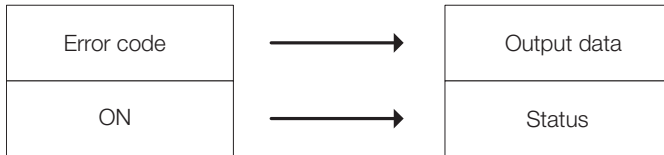
If the instruction ends normally, the output data is set to 0 and the status is turned OFF. If an error occurs, an error code is set in the output data and the status is turned ON.



- If the Queue Clear Succeeds



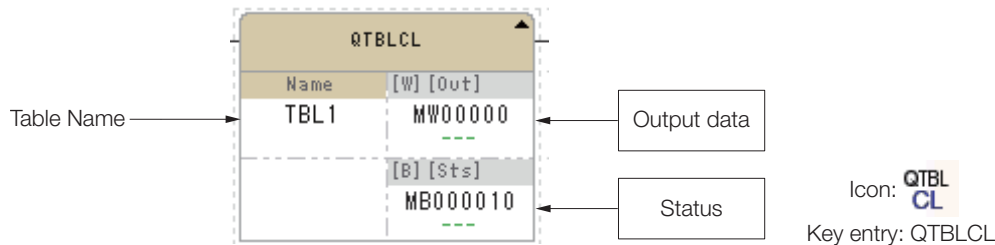
- If the Queue Clear Fails



Information If the clear fails, the queues will retain the contents from before the instruction was executed.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Name (Table name) ^{*1*2}	×	×	×	×	×	×	○	×	×
Out (Output data) ^{*3}	×	○ ^{*4}	×	×	×	×	×	○	×
Sts (Status) ^{*3}	○ ^{*4}	×	×	×	×	×	×	×	×

*1. Specify the registers in which the text string for the table name (8 bytes + NULL character max.) has been stored.
 *2. G, M, D, or C register only.
 *3. Optional.
 *4. C and # registers cannot be used.

Details on I/O Items

Item	Name	I/O	Description
Name	Table name	IN	For the QTBLCL instruction, directly enter the table name. For the QTBLCLE instruction, indirectly designate the table name in registers.
Out	Output data	OUT	Specify the first address of the table data.
Sts	Status	OUT	Specify the address for checking the status of this instruction.



Important

Note the following precautions regarding Name.

- Always add a NULL character to Name. If a NULL character is not added to Name, a fixed length of 8 bytes (4 words) of data is read and handled as the table name. This is a risk that an unintended table name may be referenced because of this.
- When characters have been set with ASCII instructions, the NULL character is not set at the end of the text string. Use the STRSET instruction to set Name.
- An operation error will occur if the size from the first register to the maximum range of registers is less than 8 bytes (4 words).

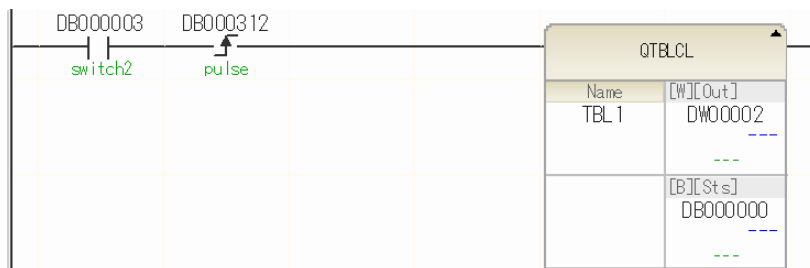
◆ Error Codes

Error Code	Error Name	Meaning
0001 hex	Table undefined	The target table is undefined.
0002 hex	Outside range of row numbers	The row number of the table element is outside the target table.
0003 hex	Outside range of column numbers	The column number of the table element is outside the target table.
0004 hex	Incorrect number of elements	The number of target elements is invalid.
0005 hex	Insufficient storage area	The storage area is insufficient.
0006 hex	Insufficient element type	The data type specified for the element is wrong.
0007 hex	Queue buffer error	An attempt was made to read from an empty queue buffer, or to write to a full queue buffer by advancing the pointer.
0008 hex	Queue table error	The specified table is not a queue table.
0009 hex	System error	An unexpected error was detected in the system during instruction execution.

Note: The error codes apply to all table manipulation instructions.

Programming Example

In the following programming example, the queue pointers for the specified queue table are initialized when switch 2 (DB000003) turns ON.



Information

When the power is turned ON, the data pointed to by the read pointer and write pointer is undefined. Always execute the QTBLCL/QTBLCLE instruction before using the QTBLR/QTBLRE, QTBLRI/QTBLRIE, QTBLW/QTBLWE, or QTBLWI/QTBLWIE instruction. An operation error may occur if the QTBLR/QTBLRE, QTBLRI/QTBLRIE, QTBLW/QTBLWE, or QTBLWI/QTBLWIE instruction is executed without executing the QTBLCL/QTBLCLE instruction first.

4.10 System Function Instructions

4.10.1 Counter (COUNTER)

When the count up or count down command changes from OFF to ON, the current value is incremented or decremented.

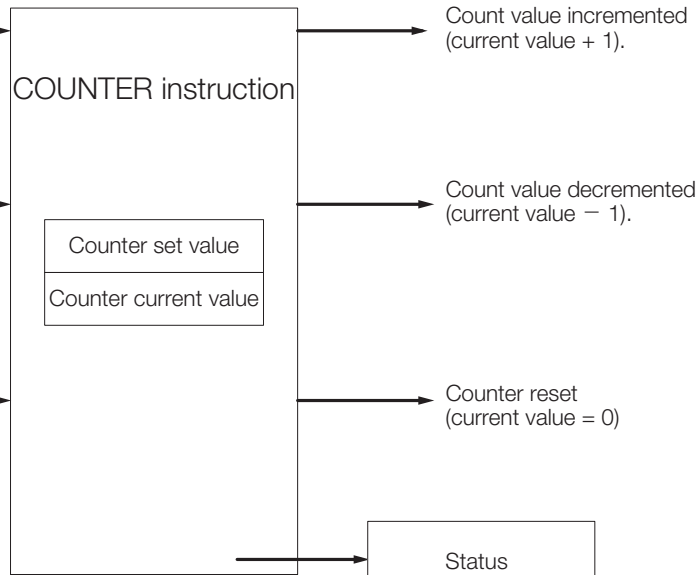
When the counter reset command turns ON, the current value of the counter is set to 0. The current value of the counter is compared against the set value and the result is output.

If a counter error occurs (i.e., if the current value is greater than the set value), the current value will neither be incremented or decremented.

Rising edge of count up command (OFF → ON)

Rising edge of count down command (OFF → ON)

Counter reset command ON

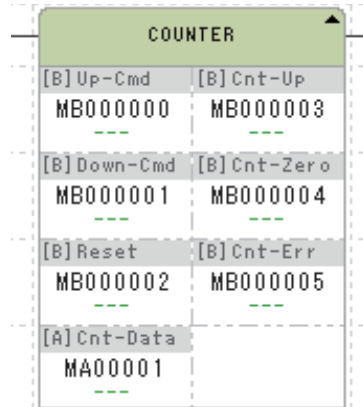


Three status are output as shown below.

- Count matched (current value = set value).
- Count is zero (current value = 0).
- Counter error (Current value > set value or current value < 0)

Format

The format of this instruction is shown below.



Icon: **COUNTER**

Key entry: COUNTER

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Up-Cmd (Count up command)	○	×	×	×	×	×	×	×	×
Down-Cmd (Count down command)	○	×	×	×	×	×	×	×	×
Reset (Counter reset command)	○	×	×	×	×	×	×	×	×
Cnt-Data (First address of counter processing data area)	×	×	×	×	×	×	○*1	×	×
Cnt-Up (Count up)	○*2	×	×	×	×	×	×	×	×
Cnt-Zero (Zero count)	○*2	×	×	×	×	×	×	×	×
Cnt-Err (Count error)	○*2	×	×	×	×	×	×	×	×

*1. M or D register only.

*2. C and # registers cannot be used.

The following table describes each input and output item.

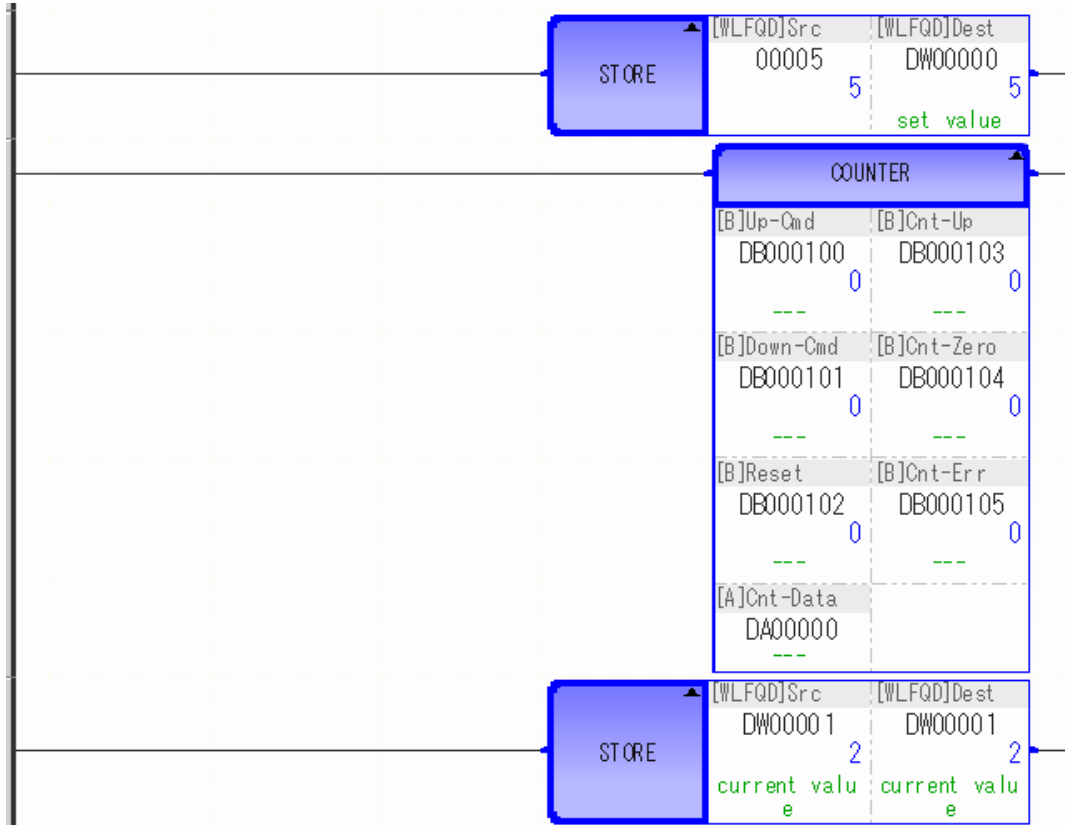
I/O Item	Description	I/O
Up-Cmd (Count up command)	The count value is incremented when this command changes from OFF to ON.*	IN
Down-Cmd (Count down command)	The count value is decremented when this command changes from OFF to ON.*	IN
Reset (Counter reset command)	The current value is reset to 0 when this command is ON.	IN
Cnt-Data (First address of counter processing data area)	+0 word: Set value	IN
	+1 word: Current value	OUT
	+2 word: Work flags	OUT
Cnt-Up (Count up)	Turns ON when the current value equals the set value.	OUT
Cnt-Zero (Zero count)	Turns ON when the current value equals 0.	OUT
Cnt-Err (Count error)	Turns ON when the current value is greater than the set value. Also turns ON when the current value is less than 0.	OUT

* If the count up command and count down command change from OFF to ON at the same time, the current value stays the same.

Programming Example

In the following programming example, the first line sets the counter set value to 5, and the third line monitors the counter current value in DW00001.

When the count up command (DB000100) changes from OFF to ON, DW00001 is incremented, and when the count down command (DB000101) changes from OFF to ON, DW00001 is decremented.

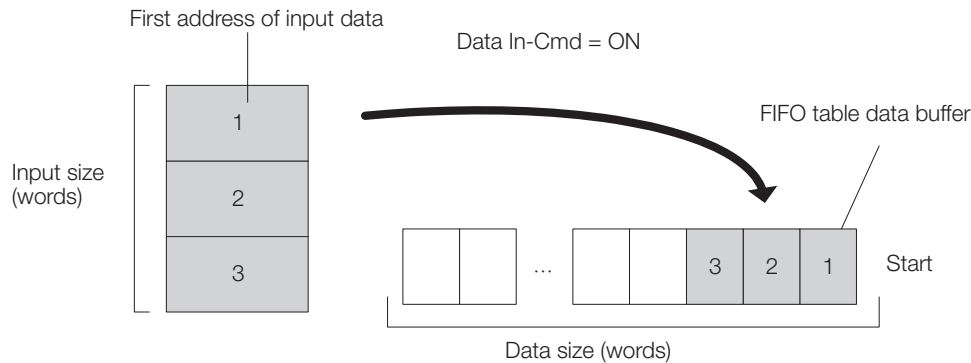


4.10.2 First-in First-out (FINFOUT)

This is a first-in first-out block data transfer function. The FIFO table consists of a 4-word header and a data buffer. Make sure to set the data size, input size, and output size words in the header before calling the FINFOUT instruction.

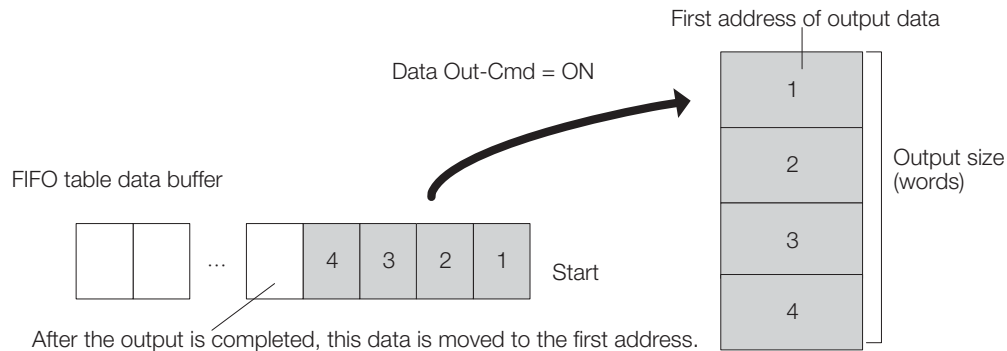
■ If the Data Input Command (In-Cmd) Is ON

When the In-Cmd is ON, the specified number of data items from the specified input data area are stored sequentially in the data area of the FIFO table.



■ If the Data Output Command (Out-Cmd) Is ON

When the Out-Cmd is ON, the specified number of data items are moved from the first address in the data area of the FIFO table to the specified output data area.



■ If the Reset Command (Reset) Is ON

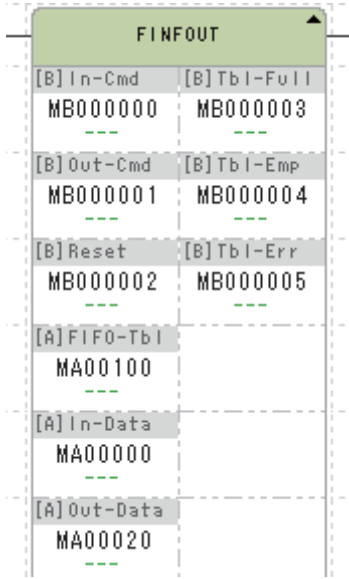
The number of words stored in the FIFO table is set to 0 and Tbl-Emp (FIFO table empty) turns ON.

Note: The contents of the table buffer are retained and not cleared to 0.

Information If the data empty size is less than the input size or if the data size is less than the output size, Tbl-Err (FIFO table error) turns ON.

Format

The format of this instruction is shown below.



Icon: **FIN**
FOUT

Key entry: FINFOUT

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In-Cmd (Data input command)	○	×	×	×	×	×	×	×	×
Out-Cmd (Data output command)	○	×	×	×	×	×	×	×	×
Reset (Reset command)	○	×	×	×	×	×	×	×	×
FIFO-Tbl (First address of FIFO table)	×	×	×	×	×	×	○*1	×	×
In-Data (First address of input data)	×	×	×	×	×	×	○*1	×	×
Out-Data (First address of output data)	×	×	×	×	×	×	○*1	×	×
Tbl-Full (FIFO table full)	○*2	×	×	×	×	×	×	×	×
Tbl-Emp (FIFO table empty)	○*2	×	×	×	×	×	×	×	×
Tbl-Err (FIFO table error)	○*2	×	×	×	×	×	×	×	×

*1. M or D register only.

*2. C and # registers cannot be used.

The following table describes each input and output item.

I/O Item	Description	I/O
In-Cmd (Data input command)	Data is stored in the FIFO table when this command is ON.	IN
Out-Cmd (Data output command)	Data is transferred out of the FIFO table when this command is ON.	IN
Reset (Reset command)	The number of words to store is set to 0 when this command is ON.	IN
FIFO-Tbl (First address of FIFO table)	+0 word: Data size	IN
	+1 word: Input size	IN
	+2 word: Output size	IN
	+3 word: Data storage size	OUT
	+4 word and on: Data	OUT
In-Data (First address of input data)	First address of input data	IN
Out-Data (First address of output data)	First address of output data	IN

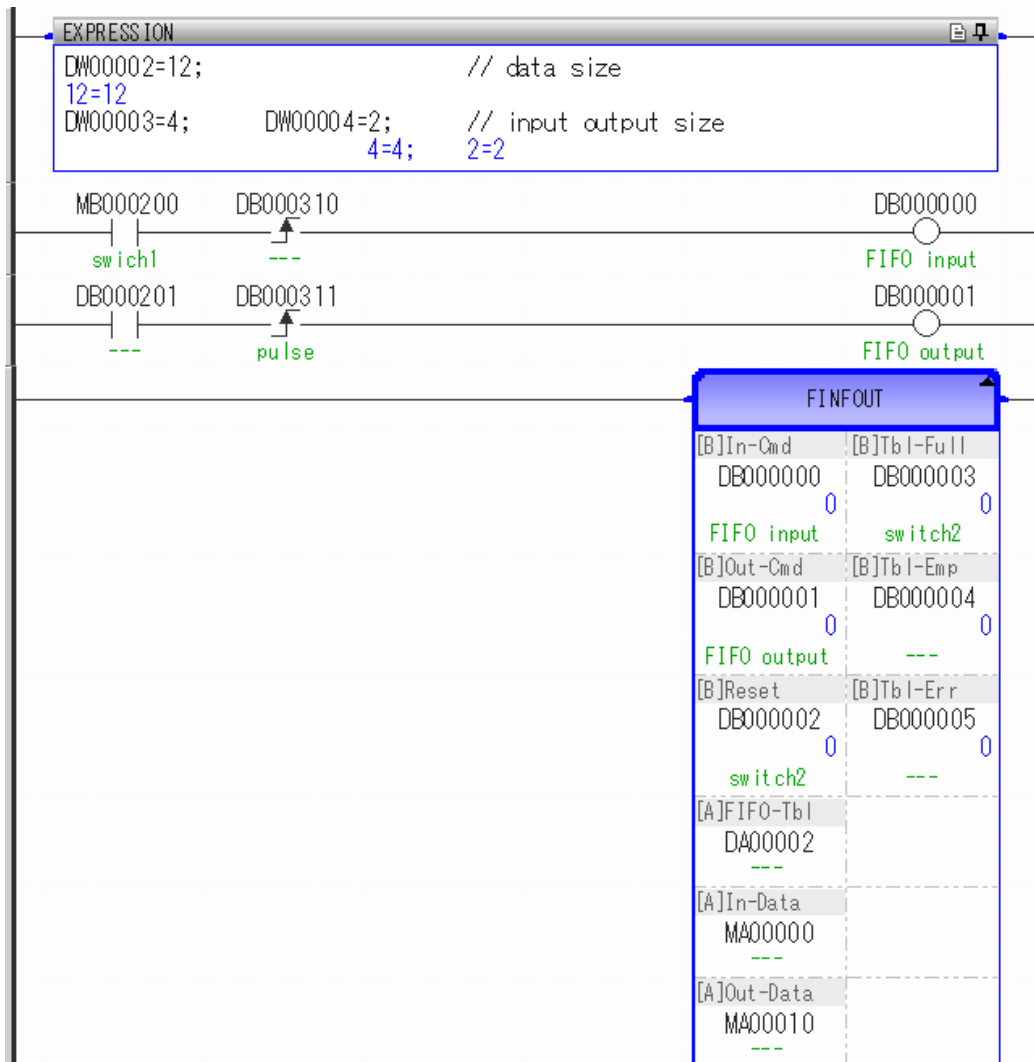
Continued on next page.

Continued from previous page.

I/O Item	Description	I/O
Tbl-Full (FIFO table full)	Turns ON when the FIFO table is full.	OUT
Tbl-Emp (FIFO table empty)	Turns ON when the FIFO table is empty.	OUT
Tbl-Err (FIFO table error)	Turns ON when the FIFO table has an error.	OUT

Programming Example

In the following programming example, a FIFO table is created with a data size of 12 words, input size of 4 words, and an output size of 2 words, and then the FINFOUT instruction is executed.



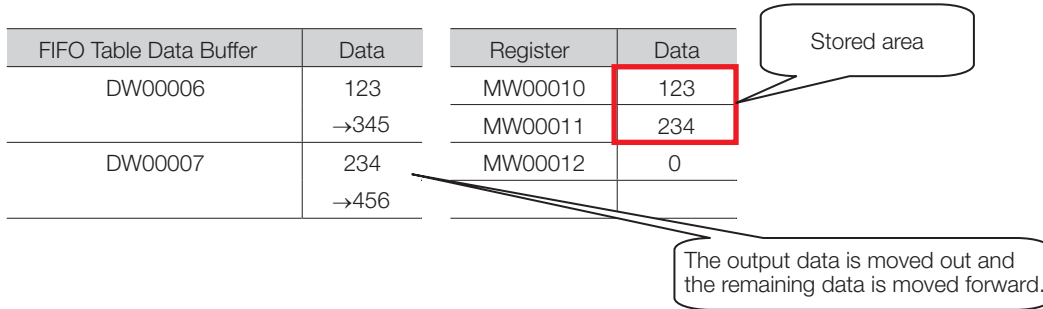
The data from MW00000 to MW00003 is stored in the FIFO table buffer when switch 1 (MB000200) turns ON.

The data storage size in DW00005 is set to 4.

Register	Data	FIFO Table Data Buffer	Data
MW00000	123	DW00006	123
MW00001	234	DW00007	234
MW00002	345	DW00008	345
MW00003	456	DW00009	456
		DW00010	0
		:	:
		DW00017	0

A callout box labeled "Stored area" points to the data values 123, 234, 345, and 456 in the FIFO Table Data Buffer column.

Next, when switch 2 (MB000201) turns ON, two words of data from the first address in the FIFO table buffer are output to the area from MW0010 to MW0011. The data storage size in DW00005 is set to 2.



4.10.3 Trace (TRACE)

This instruction performs trace execution control of the trace data that is specified by the trace group number (1 to 4).

The MP3000 Series is equipped with the following three types of traces and the trace settings are configured in the MPE720.

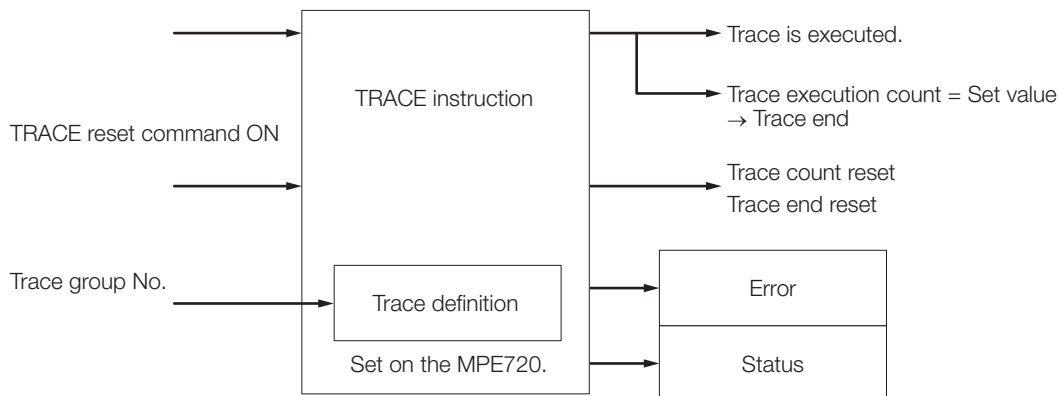
Trace Type	Key Items to Set in the MPE720	Limitations When Executed with the Trace (TRACE) Instruction
Real-Time Trace	<ul style="list-style-type: none"> Register address of trace subject Trace mode setting Sampling setting Trigger setting 	Trc-End (trace end) cannot be used.
Trace Manager	<ul style="list-style-type: none"> Register address of trace subject Sampling Trace count Trace start condition Trace stop condition 	<ul style="list-style-type: none"> Set the trace count to use Trc-End (trace end). The extended specifications on the MP3000 cannot be used (register type/data type/register range/maximum data buffer size).
XY Trace	<ul style="list-style-type: none"> X-Y axis specification Trace subject Trace mode setting Sampling setting Trigger setting 	Trc-End (trace end) cannot be used.

Information The trace definition is set in the Data Trace Definitions in the MPE720. Refer to the following manual for details.

MP3000 Series Machine Controller System Setup Manual (Manual No.: SIEP C880725 00)

- The trace is executed if Execute (trace execution command) is ON.
- The trace counter is reset when Reset (trace reset command) turns ON. This also resets Trc-End (trace end).
- Trc-End (trace end) turns ON when the specified number of traces have been executed.

TRACE execution command ON



Format

The format of this instruction is shown below.

TRACE	
[B] Execute	[B] Trc-End
M000000	M000002
[B] Reset	[B] Error
M000001	M000003
[W] Group-No	[W] Status
M000002	M000001

Icon: **TR
CE**

Key entry: TRACE

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute (Trace execution command)	○	×	×	×	×	×	×	×	×
Reset (Trace reset command)	○	×	×	×	×	×	×	×	×
Group-No (Trace group No.)	×	○	×	×	×	×	×	○	○
Trc-End (Trace end)	○*	×	×	×	×	×	×	×	×
Error	○*	×	×	×	×	×	×	×	×
Status	×	○*	×	×	×	×	×	○	×

* C and # registers cannot be used.

The following table describes each input and output item.

I/O Item	Description	I/O
Execute (Trace execution command)	Trace execution begins when this command turns ON.	IN
Reset (Trace reset command)	Trace execution is reset when this command turns ON.	IN
Group-No (Trace group No.)	Trace group No. (1 to 4)	IN
Trc-End (Trace end)	Turns ON when the trace ends.	OUT
Error	Turns ON when an error occurs.	OUT
Status	Trace execution status.	OUT

The status configuration is shown below.

Bit	Name	Remarks
0	Trace data full	Turns ON after one turn through the data trace memory of the specified group.
1 to 7	Reserved for system.	–
8	No trace definition	The function will not be executed.
9	Group No. error	The function will not be executed.
A to C	Reserved for system.	–
D	Execution timing error	The function will not be executed.
E	Reserved for system.	–
F	Reserved for system.	–

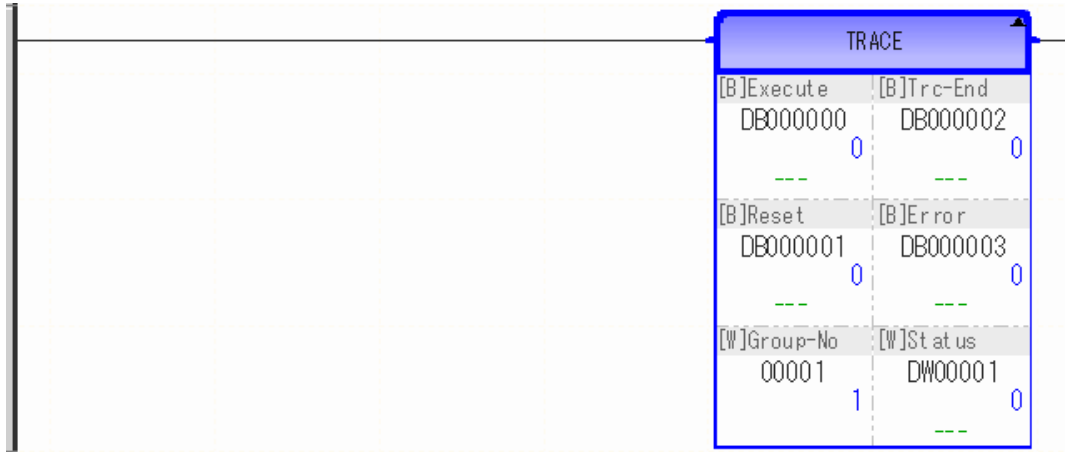
- Information** For MPE720 software version 7.42 or higher and CPU version 1.37 or higher, when there is a trace definition and the trace settings are disabled from the MPE720 or the trace buffer was reset, bit 8 (No trace definition) of Status (Status) is turned ON. Perform the following operations to turn OFF bit 8 (No trace definition) of Status (Status).
- Enable the trace settings.
 - If the trace buffer was reset, save the trace data definition again.

Programming Example

In the following programming example, the definition for trace group No. 1 is used to execute a trace.

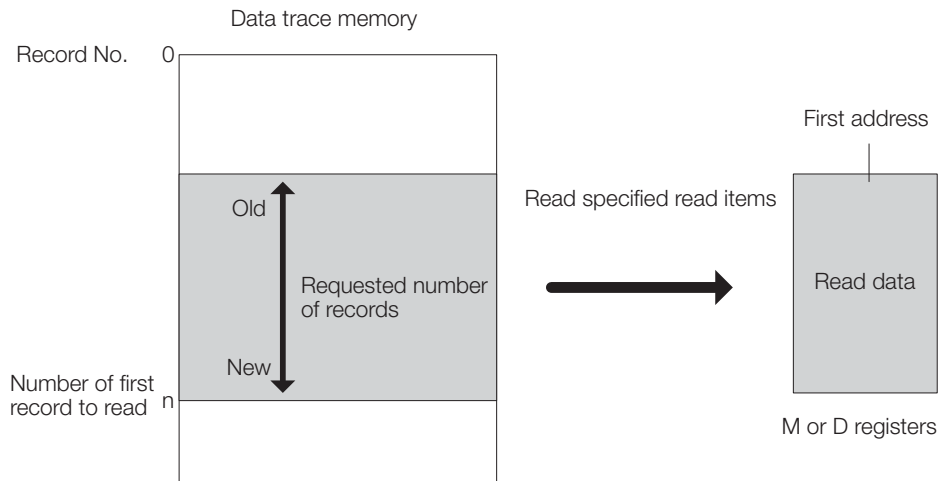
The trace starts when the trace execution command (DB000000) turns ON.

Information Set the data trace definition for trace group No. 1 on the MPE720 in advance. Make sure to set the sampling condition to *Program*.



4.10.4 Read Data Trace (DTRC-RD/DTRC-RDE)



Trace data in the Machine Controller is read and stored in registers. The data in the trace memory can be read by specifying the record number and the number of records. You can designate and read only the required items in a record.



Format

The format of this instruction is shown below.

DTRC-RD	
[B] Execute	[B] Complete
M000000	M000001
[W] Group-No	[B] Error
M000001	M000002
[W] Rec-No	[W] Status
M000002	M000005
[W] Rec-Size	[W] Rec-Size
M000003	M000006
[W] Select	[W] Rec-Len
M000004	M000007
[A] Dat-Adr	
MA00010	

Icon:  Data-Trace Read
 Data-Trace Read Extend

Key entry: DTRC-RD/DTRC-RDE

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute (Trace read execution command)	○	×	×	×	×	×	×	×	×
Group-No (Trace group No.)	×	○	×	×	×	×	×	×	×
Rec-No (Record No.)	×	○ ^{*3}	○ ^{*4}	×	×	×	×	×	×
Rec-Size (Number of records)	×	○ ^{*3}	○ ^{*4}	×	×	×	×	×	×
Select (Item selection)	×	○ ^{*3}	×	○ ^{*4}	×	×	×	×	×
Dat-Adr (First address)	×	×	×	×	×	×	○ ^{*1}	×	×
Complete (Trace completed)	○ ^{*2}	×	×	×	×	×	×	×	×
Error	○ ^{*2}	×	×	×	×	×	×	×	×
Status	×	○ ^{*2}	×	×	×	×	×	×	×
Rec-Size (Number of records read)	×	○ ^{*2*3}	○ ^{*4}	×	×	×	×	×	×
Rec-Len (Length of 1 read record)	×	○ ^{*2*3}	○ ^{*4}	×	×	×	×	×	×

*1. M or D register only.

*2. C and # registers cannot be used.

*3. For the DTRC-RD instruction.

*4. For the DTRC-RDE instruction.

4.10.4 Read Data Trace (DTRC-RD/DTRC-RDE)

The following table describes each input and output item.

<DTRC-RD>

I/O Item	Description	I/O
Execute (Trace read execution command)	Data trace read execution command	IN
Group-No (Trace group No.)	Data trace group No. (1 to 4).	IN
Rec-No (Record No.)	Number of first record to read (0 to maximum records - 1)	IN
Rec-Size (Number of records)	Requested records to read (0 to maximum records - 1)	IN
Select (Item selection)	Items to be read (0001 to FFFF hex) Bits 0 to F correspond to data specifiers 1 to 16 in the trace definition.	IN
Dat-Adr (First address)	Number of first register to read (MA, DA)	IN
Complete (Trace completed)	Turns ON when the trace read ends.	OUT
Error	Turns ON when an error occurs.	OUT
Status	Data trace read execution status	OUT
Rec-Size (Number of records read)	Number of records read	OUT
Rec-Len (Length of 1 read record)	Length of 1 read record (words)	OUT

<DTRC-RDE>

I/O Item	Description	I/O
Execute (Trace read execution command)	Data trace read execution command	IN
Group-No (Trace group No.)	Data trace group No. (1 to 4).	IN
Rec-No (Record No.)	Number of first record to read (0 to maximum records - 1)	IN
Rec-Size (Number of records)	Requested records to read (0 to maximum records - 1)	IN
Select (Item selection)	Items to be read (0000000000000001 to FFFFFFFFFFFFFFFF hex) Bits 0 to 3F correspond to data specifiers 1 to 16 in the trace definition.	IN
Dat-Adr (First address)	Number of first register to read (MA, GA, or DA)	IN
Complete (Trace completed)	Turns ON when the trace read ends.	OUT
Error	Turns ON when an error occurs.	OUT
Status	Data trace read execution status	OUT
Rec-Size (Number of records read)	Number of records read	OUT
Rec-Len (Length of 1 read record)	Length of 1 read record (words)	OUT

The status configuration is shown below.

Bit	Name	Remarks
0 to 7	Reserved for system.	–
8	No trace definition	The function will not be executed.
9	Group No. error	The function will not be executed.
A	Specified record No. error	The function will not be executed.
B	Specified number of records error	The function will not be executed.
C	Data storage error	The function will not be executed.
D	Reserved for system.	–
E	Reserved for system.	–
F	Address input error	The function will not be executed.

Programming Example

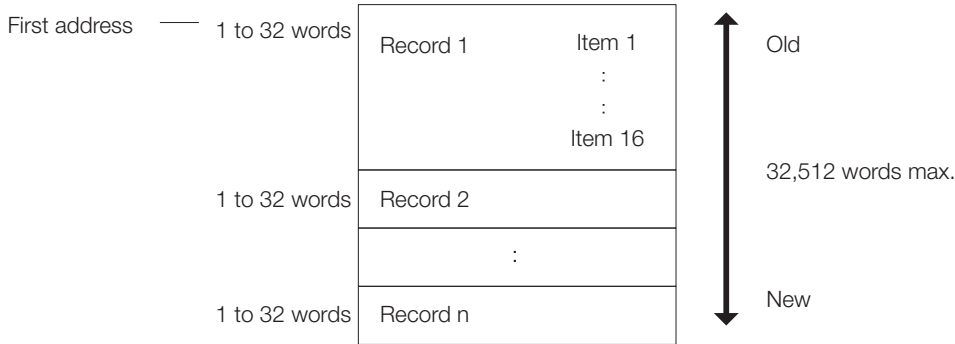
The following programming example reads the data trace for group definition No. 1. The trace data is read when the trace read execution command (DB000000) turns ON.

DTRC-RD	
[B]Execute DB000000 ---	[B]Complete DB000001 ---
[W]Group-No DW000001 ---	[B]Error DB000002 ---
[W]Rec-No DW000002 ---	[W]Status DW000005 ---
[W]Rec-Size DW000003 ---	[W]Rec-Size DW000006 ---
[W]Select DW000004 ---	[W]Rec-Len DW000007 ---
[A]Dat-Adr DA000010 ---	

Additional Information

◆ Structure of Read Data

The length of a record can be from 1 to 32 words, depending on the selected data items. The maximum number of records can be from 1,015 to 32,511 depending on the record length.



◆ Record Lengths

A record consists of the selected data items.

The record length (number of words in a single record) is determined by the selected registers and the number of data items.

- Number of words for 1 record = $B_n \times 1 \text{ word} + W_n \times 1 \text{ word} + L_n \times 2 \text{ words} + F_n \times 2 \text{ words}$
 B_n: Number of selected bit registers
 W_n: Number of selected integer registers
 L_n: Number of selected double-length integer registers
 F_n: Number of selected real number registers
 The maximum total is 16 items.
- Maximum record length = 32 words (with 16 double-length integers or real number registers)
- Minimum record length = 1 word (with 1 record for each bit or integer register)

◆ **Number of Records**

The number of records that can be specified depends on the record length as shown below.

- Number of records with the maximum record length: 0 to 1,015
- Number of records with the minimum record length: 0 to 32,511
(Upper limit: 32,521 divided by the record length - 1)

◆ **Latest record number**

The most recent record number for each trace group is stored in the system registers as shown below.

System Register Address	Description
SW00100	Latest record number in group 1.
SW00101	Latest record number in group 2.
SW00102	Latest record number in group 3.
SW00103	Latest record number in group 4.
SW00104	-
SW00105	-
SW00106	-
SW00107	-

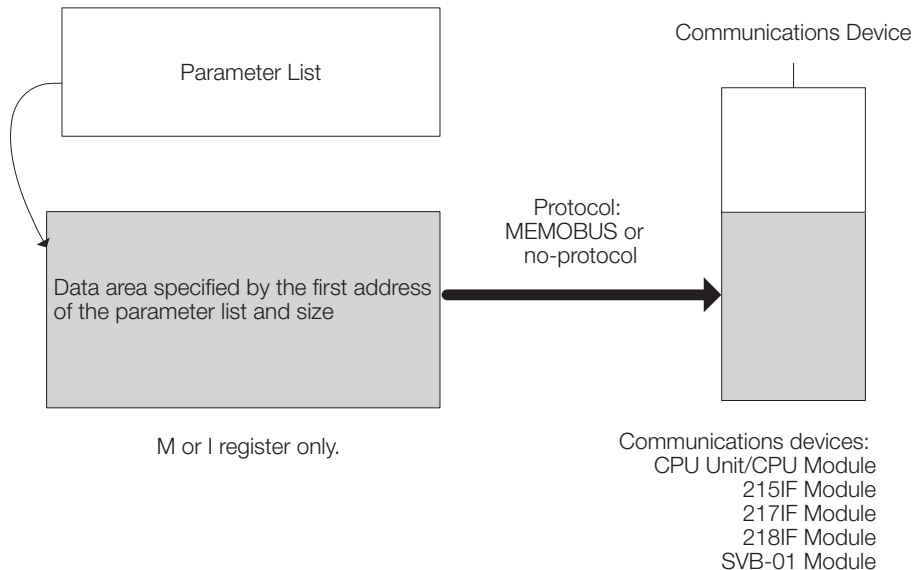
4.10.5 Send Message (MSG-SND)

A message is sent to a remote station on the specified circuit of the communications device type.

This function supports the following communications devices and protocols.

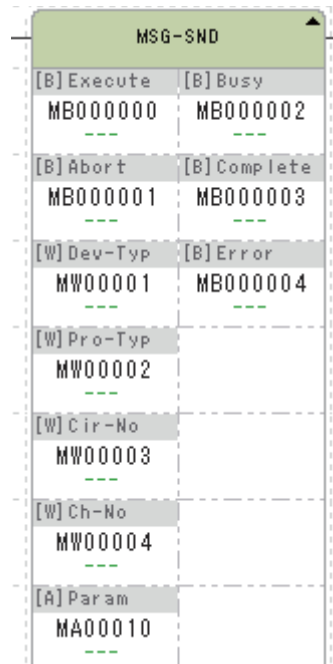
Communications devices: CPU Unit/CPU Module, 215IF Module, 217IF Module, 218IF Module, and SVB-01 Module

Protocol: MEMOBUS communications or no-protocol



Format

The format of this instruction is shown below.



Icon: **MSG**
-SND

Key entry: MSG-SND

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute (Send execution command)	○	×	×	×	×	×	×	×	×
Abort (Send abort command)	○	×	×	×	×	×	×	×	×
Dev-Typ (Communications device type)	×	○	×	×	×	×	×	○	○
Pro-Typ (Communications protocol)	×	○	×	×	×	×	×	○	○
Cir-No (Circuit number)	×	○	×	×	×	×	×	○	○
Ch-No (Communications buffer channel number)	×	○	×	×	×	×	×	○	○
Param (First address of parameter list)	×	×	×	×	×	×	○ ^{*1}	×	×
Busy (Processing)	○ ^{*2}	×	×	×	×	×	×	×	×
Complete (Processing completed)	○ ^{*2}	×	×	×	×	×	×	×	×
Error (Error occurred)	○ ^{*2}	×	×	×	×	×	×	×	×

*1. M, G, or D register only.

*2. C and # registers cannot be used.

Refer to the following manual for details on I/O items, parameters, and programming examples.

📖 MP2000 Series Ladder Programming User's Manual (Manual No.: SIEZ-C887-1.2)

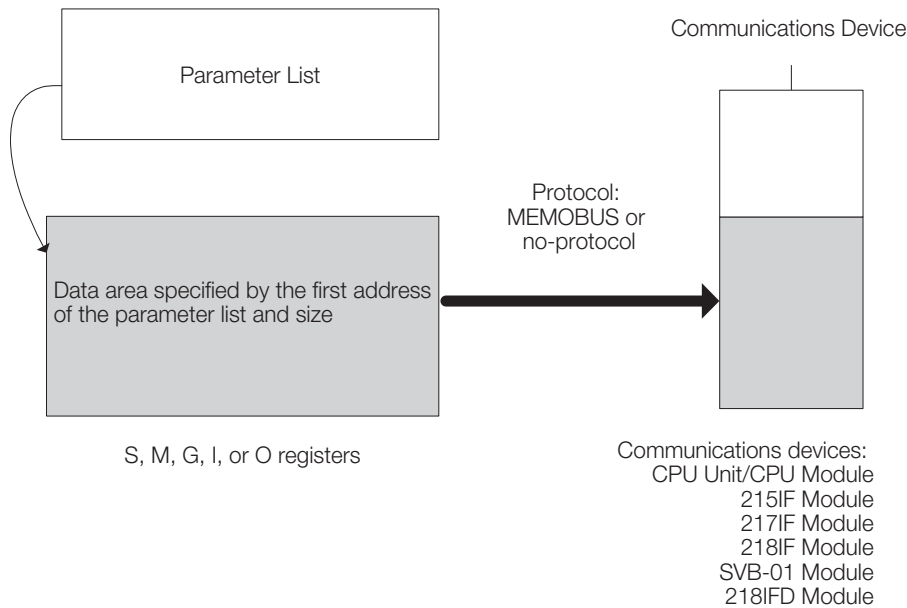
4.10.6 Send Message Extended (MSG-SNDE)

A message is sent to a remote station on the specified circuit of the communications device type.

This function supports the following communications devices and protocols.

Communications devices: CPU Unit/CPU Module, 215IF Module, 217IF Module, 218IF Module, SVB-01 Module, and 218IF Module

Protocol: MEMOBUS communications or no-protocol



The basic operation is the same as for the MSG-SND function. However, normally, you should use the MSG-SNDE function for compatibility with the MP3000-series Machine Controllers.

The MSG-SND function is compatible with the MP2000-series Machine Controllers. The accessible range of registers is different, as shown below.

Name of the register	Access Range for the MSG-SNDE		Access Range for the MSG-SND	
	Address Range	Access	Address Range	Access
System registers	SW00000 to 65534	RW	–	–
Hold registers	MW0000000 to 1048575	RW	MW0000000 to 0065534	RW
Data registers	GW0000000 to 2097151	RW	–	–
Input registers	IW00000 to 17FFF	R	IW00000 to 0FFFF	R
Output registers	OW00000 to 17FFF	RW	–	–

Note: R: Read only, RW: Read/Write

Format

The format of this instruction is shown below.

MSG-SNDE	
[B] Execute MB000000	[B] Busy MB000002
[B] Abort MB000001	[B] Complete MB000003
[W] Dev-Typ MW000001	[B] Error MB000004
[W] Pro-Typ MW000002	
[W] Cir-No MW000003	
[W] Ch-No MW000004	
[A] Param MA00010	

Icon: 


Key entry: MSG-SNDE

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute (Send execution command)	○	×	×	×	×	×	×	×	×
Abort (Send abort command)	○	×	×	×	×	×	×	×	×
Dev-Typ (Communications device type)	×	○	×	×	×	×	×	○	○
Pro-Typ (Communications protocol)	×	○	×	×	×	×	×	○	○
Cir-No (Circuit number)	×	○	×	×	×	×	×	○	○
Ch-No (Communications buffer channel number)	×	○	×	×	×	×	×	○	○
Param (First address of parameter list)	×	×	×	×	×	×	○*1	×	×
Busy (Processing)	○*2	×	×	×	×	×	×	×	×
Complete (Processing completed)	○*2	×	×	×	×	×	×	×	×
Error (Error occurred)	○*2	×	×	×	×	×	×	×	×

*1. M, G, or D register only.

*2. C and # registers cannot be used.

Refer to the following manual for details on I/O items, parameters, and programming examples.

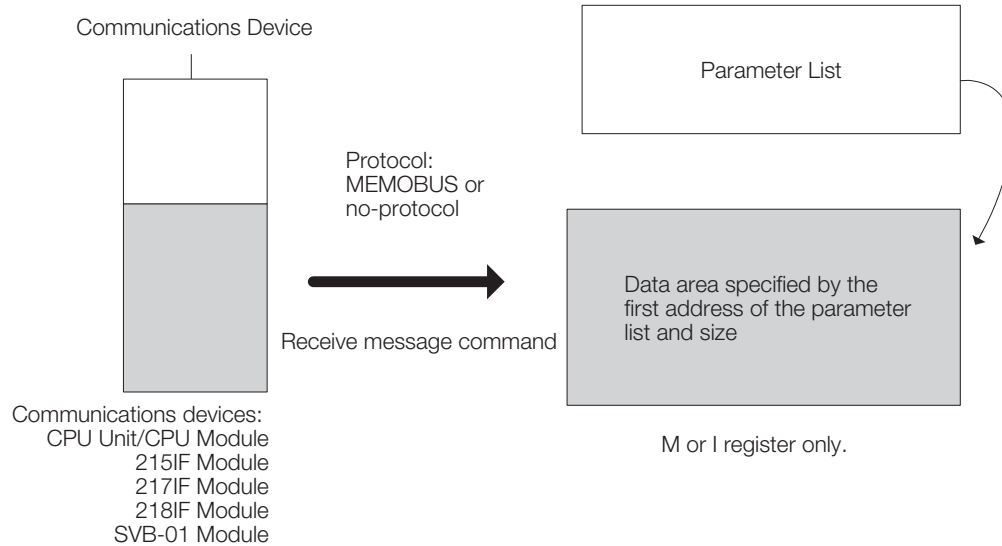
 MP3000 Series Communications User's Manual (Manual No.: SIEP C880725 12)

4.10.7 Receive Message (MSG-RCV)

A message is received from a remote station on the specified circuit of the communications device type. Keep the message receive command ON until the Complete Bit turns ON. This function supports the following communications devices and protocols.

Communications devices: CPU Unit/CPU Module, 215IF Module, 217IF Module, 218IF Module, and SVB-01 Module

Protocol: MEMOBUS communications or no-protocol



Note: The Complete Bit turns ON when the message reception is completed. Until then, keep the receive message command ON.

Format

The format of this instruction is shown below.

MSG-RCV	
[B] Execute	[B] Busy
MB000000	MB000002
[B] Abort	[B] Complete
MB000001	MB000003
[W] Dev-Typ	[B] Error
MW000001	MB000004
[W] Pro-Typ	
MW000002	
[W] Cir-No	
MW000003	
[W] Ch-No	
MW000004	
[A] Param	
MA000010	

Icon: **MSG**
RCV

Key entry: MSG-RCV

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute (Receive execution command)	○	×	×	×	×	×	×	×	×
Abort (Receive abort command)	○	×	×	×	×	×	×	×	×
Dev-Typ (Communications device type)	×	○	×	×	×	×	×	○	○
Pro-Typ (Communications protocol)	×	○	×	×	×	×	×	○	○
Cir-No (Circuit number)	×	○	×	×	×	×	×	○	○
Ch-No (Communications buffer channel number)	×	○	×	×	×	×	×	○	○
Param (First address of parameter list)	×	×	×	×	×	×	○ ^{*1}	×	×
Busy (Processing)	○ ^{*2}	×	×	×	×	×	×	×	×
Complete (Processing completed)	○ ^{*2}	×	×	×	×	×	×	×	×
Error (Error occurred)	○ ^{*2}	×	×	×	×	×	×	×	×

*1. M, G, or D register only.

*2. C and # registers cannot be used.

Refer to the following manual for details on I/O items, parameters, and programming examples.

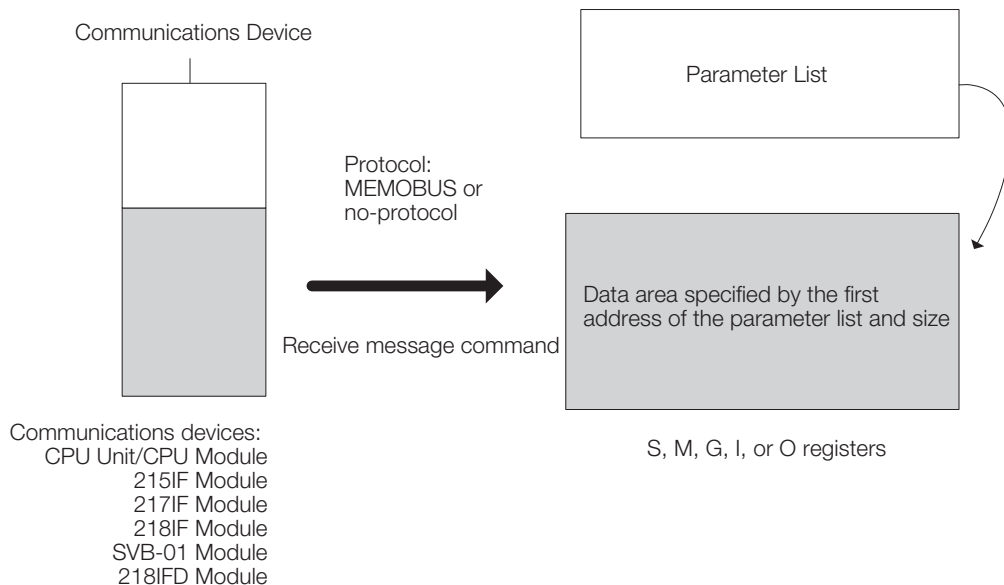
📖 MP2000 Series Ladder Programming User's Manual (Manual No.: SIEZ-C887-1.2)

4.10.8 Receive Message Extended (MSG-RCVE)

A message is received from a remote station on the specified circuit of the communications device type. Keep the message receive command ON until the Complete Bit turns ON. This function supports the following communications devices and protocols.

Communications devices: CPU Unit/CPU Module, 215IF Module, 217IF Module, 218IF Module, SVB-01 Module, and 218IF Module

Protocol: MEMOBUS communications or no-protocol



Note: The Complete Bit turns ON when the message reception is completed. Until then, keep the receive message command ON.

4.10.8 Receive Message Extended (MSG-RCVE)

The basic operation is the same as for the MSG-RCV function. However, normally, you should use the MSG-RCVE function for compatibility with the MP3000-series Machine Controllers.

The MSG-RCV function is compatible with the MP2000-series Machine Controllers. The accessible range of registers is different, as shown below.

Name of the register	Access Range for the MSG-RCVE		Access Range for the MSG-RCV	
	Start	End	Start	End
System registers	SW00000	65534	–	–
Hold registers	MW0000000	1048575	MW0000000	0065534
Data registers	GW0000000	2097151	–	–
Input registers	IW00000	17FFF	IW00000	0FFFF
Output registers	OW00000	17FFF	–	–

Note: R: Read only, RW: Read/Write

Format

The format of this instruction is shown below.

MSG-RCVE	
[B] Execute	[B] Busy
MB000000	MB000002
[B] Abort	[B] Complete
MB000001	MB000003
[W] Dev-Typ	[B] Error
MW000001	MB000004
[W] Pro-Typ	
MW000002	
[W] Cir-No	
MW000003	
[W] Ch-No	
MW000004	
[A] Param	
MA00010	

Icon: MSG-RCVE

Key entry: MSG-RCVE

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute (Receive execution command)	○	×	×	×	×	×	×	×	×
Abort (Receive abort command)	○	×	×	×	×	×	×	×	×
Dev-Typ (Communications device type)	×	○	×	×	×	×	×	○	○
Pro-Typ (Communications protocol)	×	○	×	×	×	×	×	○	○
Cir-No (Circuit number)	×	○	×	×	×	×	×	○	○
Ch-No (Communications buffer channel number)	×	○	×	×	×	×	×	○	○
Param (First address of parameter list)	×	×	×	×	×	×	○*1	×	×
Busy (Processing)	○*2	×	×	×	×	×	×	×	×
Complete (Processing completed)	○*2	×	×	×	×	×	×	×	×
Error (Error occurred)	○*2	×	×	×	×	×	×	×	×

*1. M, G, or D register only.

*2. C and # registers cannot be used.

Refer to the following manual for details on I/O items, parameters, and programming examples.

MP3000 Series Communications User's Manual (Manual No.: SIEP C880725 12)

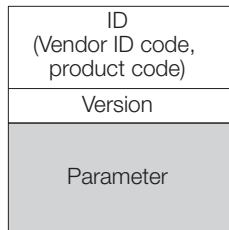
4.10.9 Write SERVOPACK Parameter (MLNK-SVW)

This instruction writes all the parameters that are saved in the Machine Controller as a SERVOPACK parameter backup file to the SERVOPACK that is specified with the circuit number and axis number.

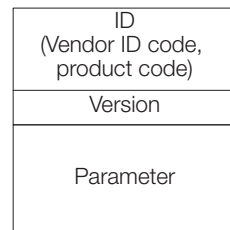
The MLNK-SVW instruction can be used to write SERVOPACK parameters using only a ladder program (i.e., without the use of MPE720).

This instruction is convenient when replacing a SERVOPACK and at other times.

Backup file of SERVOPACK parameters in the Machine Controller



Parameters for the SERVOPACK that is specified with the circuit number and axis number



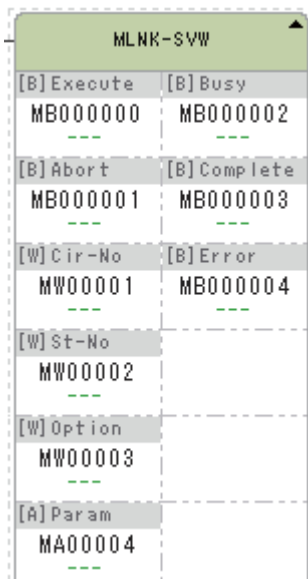
Written in one operation.

Vendor ID code: An ID code managed by the MECHATROLINK Members Association that identifies the vendor.

Product code: A unique code given to each device.

Format

The format of this instruction is shown below.



Icon: 

Key entry: MLNK-SVW

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute (Write command)	○	×	×	×	×	×	×	×	×
Abort (Write processing abort command)	○	×	×	×	×	×	×	×	×
Cir-No (Circuit number)	×	○	×	×	×	×	×	○	○
St-No (Axis number)	×	○	×	×	×	×	×	○	○
Option (Option settings)	×	○	×	×	×	×	×	○	○
Param (First address of parameter table)	×	×	×	×	×	×	○*1	×	×
Busy (Writing)	○*2	×	×	×	×	×	×	×	×
Complete (Write completed)	○*2	×	×	×	×	×	×	×	×
Error	○*2	×	×	×	×	×	×	×	×

*1. M or D register only.

*2. C and # registers cannot be used.

4.10.9 Write SERVOPACK Parameter (MLNK-SVW)

The following table describes each input and output item.

I/O Item	Meaning	I/O
Execute (Write command)	Writing the SERVOPACK parameters begins when this command is turned ON.	IN
Abort (Write processing abort command)	The write process is aborted when this command is turned ON.	IN
Cir-No (Circuit number)	Destination circuit number (1 to 16)	IN
St-No (Axis number)	Destination axis number (1 to 16).	IN
Option (Option settings)	Command Option Bit Settings Bit E: ID Check Enable/Disable; 0: Enable, 1: Disable Bit F: Version Check Enable/Disable; 0: Enable, 1: Disable The other bits are not used. Any settings in the other bits are ignored.	IN
Param (First address of parameter table)	First address of function workspace	IN
Busy (Writing)	Turns ON while the SERVOPACK parameters are being written.	OUT
Complete (Write completed)	Turns ON for one scan only after the SERVOPACK parameters are written.	OUT
Error	Turns ON for one scan only when an error occurs. (The error details are output to PARAM00 and PARAM01.)	OUT

The option settings are described in the following table.

Bit	Meaning
0 to D	Not used. (Settings will be ignored.)
E	ID Check Enable/Disable (0: Enable, 1: Disable) If the source ID information is not the same as the ID information at the write destination, an inconsistent ID information error occurs. If this bit is set to 1 (disable), this error will not be detected and the write process will still be executed. If this bit is set to 1 (disable), the model information is not checked. This can result in parameters for the wrong model type to be written, which can cause problems. An inconsistent ID Information error will also occur if a SERVOPACK parameters file that was edited or saved offline is used. In this case, make sure that there are no problems before you set this bit to 1 (disable).
F	Version Check Enable/Disable (0: Enable, 1: Disable) If the version of the source SERVOPACK (communications interface) is not the same as the version at the write destination, an inconsistent version error occurs. SERVOPACK parameters and setting ranges are sometimes different for different versions. Make sure that there are no problems before you set this bit to 1 (disable). This will allow you to write the parameters. An inconsistent version error will also occur if a SERVOPACK parameters file that was edited or saved offline is used. In this case, make sure that there are no problems before you set this bit to 1 (disable).

◆ Details on Function Workspace

This section provides the details on the function workspace. The parameter number corresponds to the word offset from the first address.

Example For example, if the first address is MA00100, set the value in MW00105 to set PARAM 05.

Parameter No.	IN/OUT	Meaning
PARAM 00	OUT	Processing Result
PARAM 01	OUT	Error code
PARAM 02	OUT	Copy of Cir-No
PARAM 03	OUT	Copy of St-No
PARAM 04	SYS	For system use #1
PARAM 05	SYS	For system use #2
PARAM 06	SYS	For system use #3

■ Processing Result (PARAM00)

This parameter outputs the result of processing for the SERVOPACK.

- 0000 hex: Processing (Busy)
- 1000 hex: Processing completed (Complete)
- 8□□□ hex: Error occurred (Error)
The following errors can occur.

Error Code	Meaning
8100 hex	Reserved.
8200 hex	Address setting error (The set data address is outside of the valid range.)
8300 hex	Reserved.
8400 hex	Circuit number setting error (The set circuit number is outside of the valid range.)
8500 hex	Reserved.
8600 hex	Axis number setting error (The set axis number is outside of the valid range.)
8700 hex	Reserved.
8800 hex	Communications interface task error (An error was returned from the communications interface task.)
8900 hex	Reserved.
8A00 hex	Function execution duplication error (More than one MLNK-SVW function was executed at the same time. Or, the MLNK-SVR function was being executed.)

■ Error Code (PARAM 01)

This parameter outputs the error code from the communications interface task. This parameter is valid only when the processing result (PARAM00) is 8800 hex.

Error Code	Meaning
0000 hex	Reserved.
0001 hex	No SERVOPACK parameter backup file
0002 hex	Backup file error
0003 hex	Inconsistent ID information
0004 hex	Inconsistent version
0005 hex	Module error
0006 hex	SERVOPACK controller command duplication error
0007 hex	Communications error
0008 hex	Undefined command
0009 hex	Invalid parameter
000A hex	Internal system error

4.10.9 Write SERVOPACK Parameter (MLNK-SVW)

■ **Copy of Cir-No (PARAM 02)**

This is a copy of the Cir-No input data.

■ **Copy of St-No (PARAM 03)**

This is a copy of the St-No input data.

■ **For System Use #1 (PARAM04)**

This parameter is used by the system. Set this parameter to 0000 hex from a user program in the first scan after the power supply is turned ON. Do not modify this parameter at any other time.

■ **For System Use #2 (PARAM05)**

This parameter is used by the system. Set this parameter to 0000 hex from a user program in the first scan after the power supply is turned ON. Do not modify this parameter at any other time.

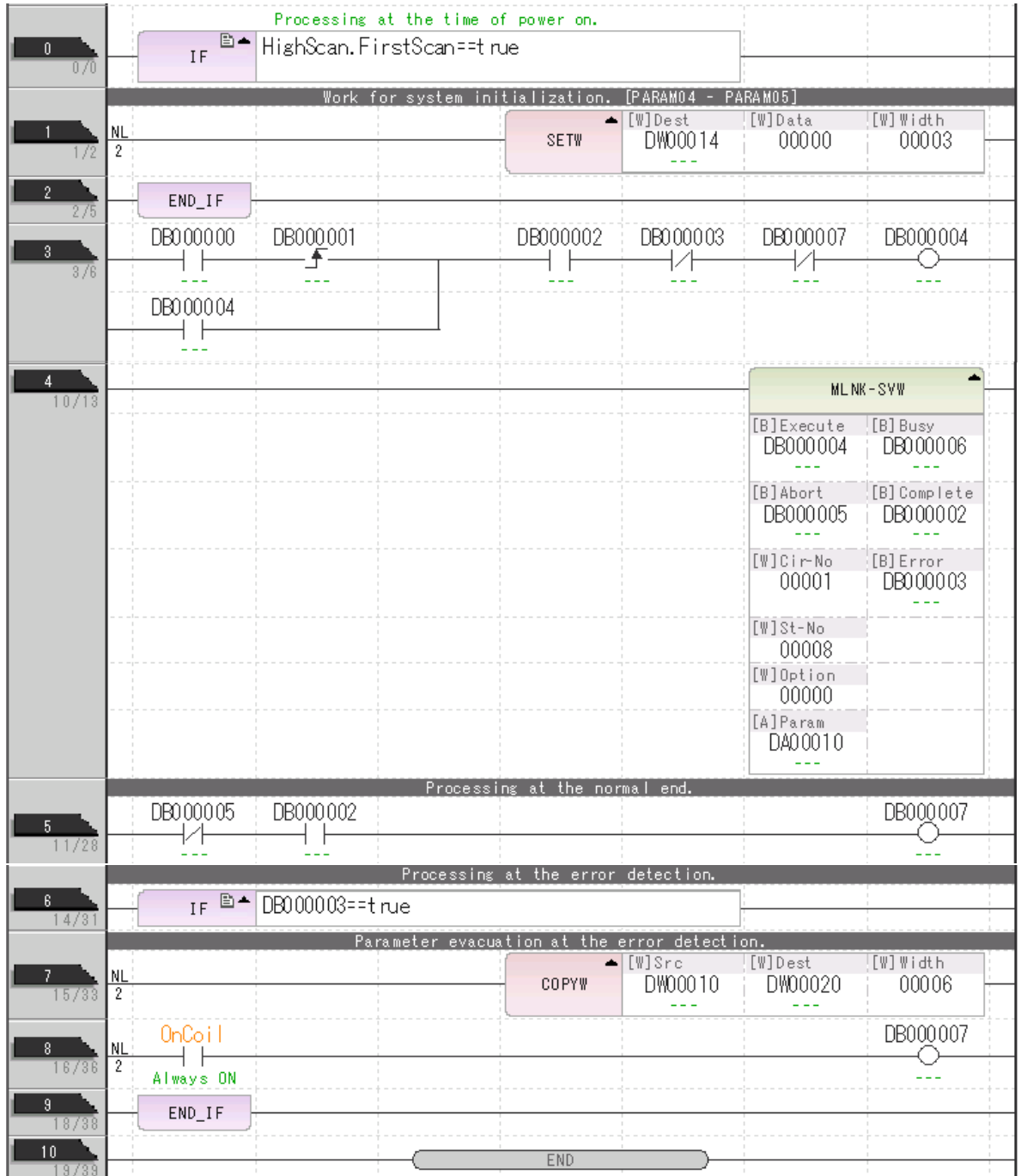
■ **For System Use #3 (PARAM06)**

This parameter is used by the system. Set this parameter to 0000 hex from a user program in the first scan after the power supply is turned ON. Do not modify this parameter at any other time.

Programming Example

The following programming example shows how to write parameters to the SERVOPACK.


If a backup file of the SERVOPACK parameters exists in the Machine Controller, the SERVOPACK parameters are written once to the specified SERVOPACK when DB000000 turns ON. The specified SERVOPACK is the one that is defined in the module configuration definition with a MECHATROLINK circuit number of 1 and defined in the MECHATROLINK detailed definition with ST#8.



4.10.10 Read SERVOPACK Parameter (MLNK-SVR)

All of the parameters are read from the RAM area of the SERVOPACK with the specified circuit number and axis number and then the read parameters are saved by overwriting the SERVOPACK parameter backup file that is saved in the Machine Controller. The MLNK-SVR instruction can be used to read SERVOPACK parameters using only a ladder program (i.e., without the use of MPE720).

This instruction is convenient when replacing a SERVOPACK and at other times.

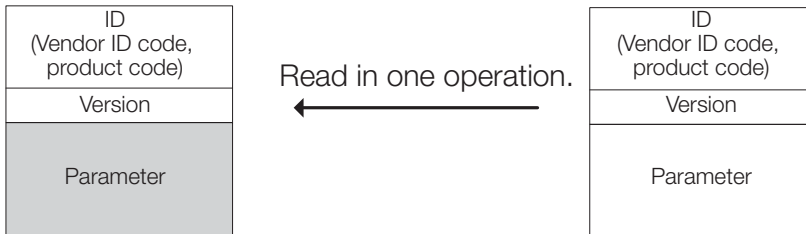


Important

- Machine Controller software version 1.23 or higher and MPE720 software version 7.34 or higher are required to execute the MLNK-SVR instruction.
- The MLNK-SVR instruction reads the parameters from the RAM area in the SERVOPACK. Therefore, if there are any difference between the parameter settings in the RAM area in the SERVOPACK and the parameter settings in non-volatile memory, the parameter settings written to the Controller and the parameter settings in the RAM area in the SERVOPACK will not agree.

Backup file of SERVOPACK parameters in the Machine Controller

Parameters for the SERVOPACK that is specified with the circuit number and axis number



Vendor ID code: An ID code managed by the MECHATROLINK Members Association that identifies the vendor.

Product code: A unique code given to each device.

Format

The format of this instruction is shown below.

MLNK-SVR	
[B] Execute	[B] Busy
MB000000	MB000002
[B] Abort	[B] Complete
MB000001	MB000003
[W] Cir-No	[B] Error
MW000001	MB000004
[W] St-No	
MW000002	
[W] Option	
MW000003	
[A] Param	
MA000004	

Icon: 
Key entry: MLNK-SVR

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute (Read command)	○	×	×	×	×	×	×	×	×
Abort (Read processing abort command)	○	×	×	×	×	×	×	×	×
Cir-No (Circuit number)	×	○	×	×	×	×	×	○	○
St-No (Axis number)	×	○	×	×	×	×	×	○	○
Option (Option settings)	×	○	×	×	×	×	×	○	○

Continued on next page.

Continued from previous page.

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Param (First address of parameter table)	×	×	×	×	×	×	O*1	×	×
Busy (Reading)	O*2	×	×	×	×	×	×	×	×
Complete (Read completed)	O*2	×	×	×	×	×	×	×	×
Error	O*2	×	×	×	×	×	×	×	×

*1. M or D register only.

*2. C and # registers cannot be used.

The following table describes each input and output item.

I/O Item	Description	I/O
Execute (Read command)	Reading the SERVOPACK parameters begins when this command is turned ON. This command must be kept ON while the instruction is in execution.	IN
Abort (Read processing abort command)	The read process is aborted when this command is turned ON.	IN
Cir-No (Circuit number)	Destination circuit number (1 to 16)	IN
St-No (Axis number)	Destination axis number (1 to 32)	IN
Option (Option settings)	Command Option Bit Settings Bit E: ID Check Enable/Disable; 0: Enable, 1: Disable Bit F: Version Check Enable/Disable; 0: Enable, 1: Disable The other bits are not used. Any settings in the other bits are ignored.	IN
Param (First address of parameter table)	First address of function workspace	IN
Busy (Reading)	Turns ON while the SERVOPACK parameters are being read.	OUT
Complete (Read completed)	Turns ON for one scan only after the SERVOPACK parameters are read.	OUT
Error	Turns ON for one scan only when an error occurs. (The error details are output to PARAM00 and PARAM01.)	OUT

The option settings are described in the following table.

Bit	Meaning
0 to D	Not used. (Settings will be ignored.)
E	ID Check Enable/Disable (0: Enable, 1: Disable) If the source ID information is not the same as the ID information at the read destination, an inconsistent ID information error occurs. If this bit is set to 1 (disable), this error will not be detected and the read process will still be executed. If this bit is set to 1 (disable), the model information is not checked. This can result in parameters for the wrong model type to be read, which can cause problems. If you replace a SERVOPACK, set this bit to 1 (disable). An inconsistent ID Information error will also occur if a SERVOPACK parameters file that was edited or saved offline is used. In this case, make sure that there are no problems before you set this bit to 1 (disable).
F	Version Check Enable/Disable (0: Enable, 1: Disable) If the version of the source SERVOPACK (communications interface) is not the same as the version at the read destination, an inconsistent version error occurs. SERVOPACK parameters and setting ranges are sometimes different for different versions. Make sure that there are no problems before you set this bit to 1 (disable). This will allow you to read the parameters. An inconsistent version error will also occur if a SERVOPACK parameters file that was edited or saved offline is used. In this case, make sure that there are no problems before you set this bit to 1 (disable).

◆ Details on Function Workspace

This section provides the details on the function workspace. The parameter number corresponds to the word offset from the first address.

Example For example, if the first address is MA00100, set the value in MW00105 to set PARAM 05.

Parameter No.	IN/OUT	Description
PARAM 00	OUT	Processing Result
PARAM 01	OUT	Error code
PARAM 02	OUT	Copy of Cir-No
PARAM 03	OUT	Copy of St-No
PARAM 04	SYS	For system use #1
PARAM 05	SYS	For system use #2
PARAM 06	SYS	For system use #3

■ Processing Result (PARAM 00)

This parameter outputs the result of processing for the SERVOPACK.

- 0000 hex: Processing (Busy)
- 1000 hex: Processing completed (Complete)
- 8□□□ hex: Error occurred (Error)
The following errors can occur.

Error Code	Meaning
8100 hex	Reserved.
8200 hex	Reserved.
8300 hex	Reserved.
8400 hex	Circuit number setting error (The set circuit number is outside of the valid range.)
8500 hex	Reserved.
8600 hex	Axis number setting error (The set axis number is outside of the valid range.)
8700 hex	Reserved.
8800 hex	Communications interface task error (An error was returned from the communications interface task.)
8900 hex	Reserved.
8A00 hex	Function execution duplication error (More than one MLNK-SVR function was executed at the same time. Or, the MLNK-SVW function was being executed.)

■ Error Code (PARAM 01)

This parameter outputs the error code from the communications interface task. This parameter is valid only when the processing result (PARAM 00) is 8800 hex.

Error Code	Meaning
0000 hex	No error
0001 hex	No SERVOPACK parameter backup file
0002 hex	Backup file error
0003 hex	Inconsistent ID information
0004 hex	Inconsistent version
0005 hex	Module error
0006 hex	SERVOPACK controller command duplication error
0007 hex	Communications error
0008 hex	Reserved.
0009 hex	Reserved.
000A hex	Internal system error

■ Copy of Cir-No (PARAM 02)

This is a copy of the Cir-No input data.

■ Copy of St-No (PARAM 03)

This is a copy of the St-No input data.

■ For System Use #1 (PARAM 04)

This parameter is used by the system. Set this parameter to 0000 hex from a user program in the first scan after the power supply is turned ON. Do not modify this parameter at any other time.

■ For System Use #2 (PARAM 05)

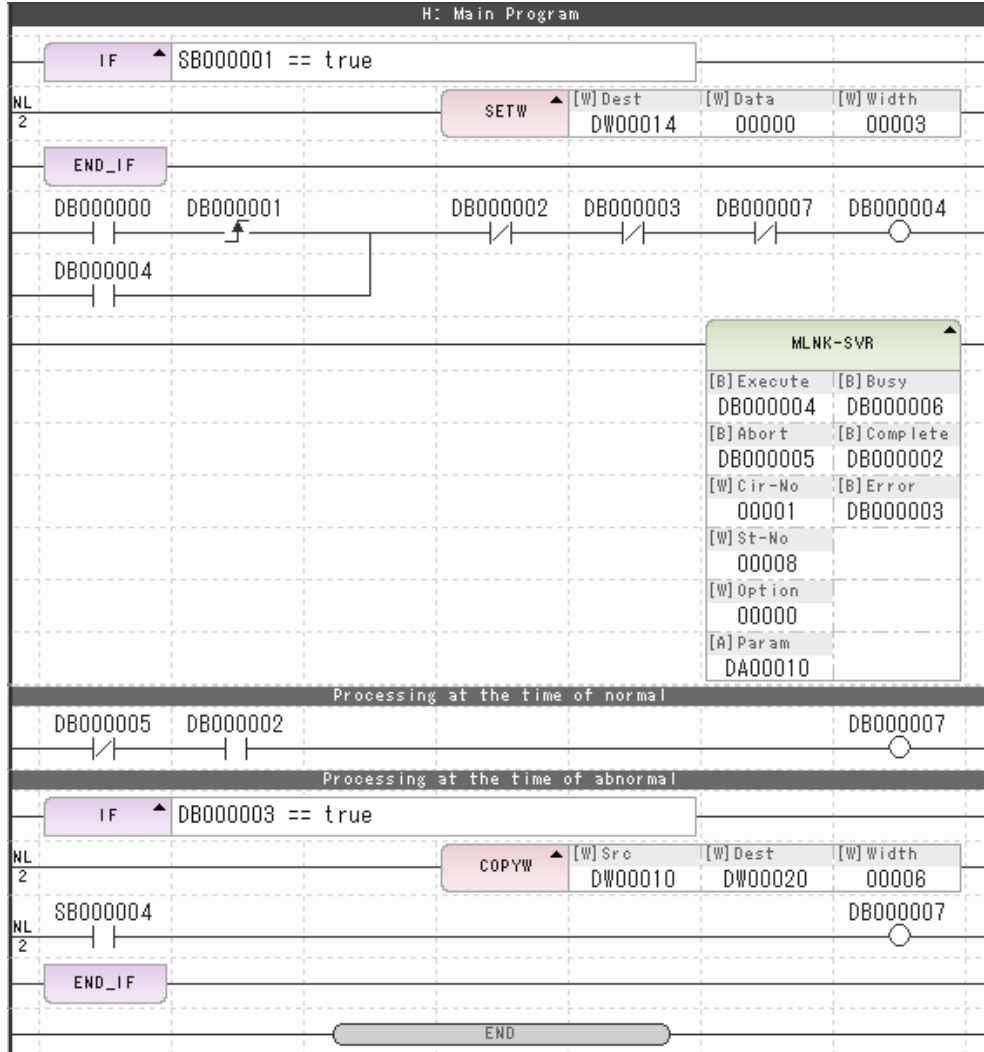
This parameter is used by the system. Set this parameter to 0000 hex from a user program in the first scan after the power supply is turned ON. Do not modify this parameter at any other time.

■ For System Use #3 (PARAM 06)

This parameter is used by the system. Set this parameter to 0000 hex from a user program in the first scan after the power supply is turned ON. Do not modify this parameter at any other time.

Programming Example

The following programming example shows how to read parameters from the SERVOPACK. If a backup file of the SERVOPACK parameters exists in the Machine Controller, the SERVOPACK parameters are read once from the specified SERVOPACK when DB000000 turns ON. The specified SERVOPACK is the one that is defined in the module configuration definition with a MECHATROLINK circuit number of 1 and defined in the MECHATROLINK detailed definition with ST#8.



4.10.11 Flash Operation (FLASH-OP)

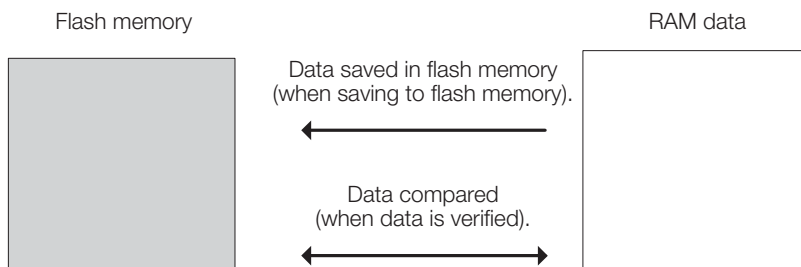
You can compare the data in flash memory and RAM in the Machine Controller or you can save the RAM data to the flash memory. You can use the FLASH-OP instruction to save data in the flash memory using only a ladder program (i.e., without the use of MPE720).

This instruction is convenient to save the data to flash memory after reading the SERVOPACK parameters with the MLNK-SVR instruction.



Important

1. Machine Controller software version 1.23 or higher and MPE720 software version 7.34 or higher are required to execute the FLASH-OP instruction.
2. Do not turn OFF the power supply to the Machine Controller until saving the data to flash memory has been completed.
If you turn OFF the power supply to the Machine Controller while data is being saved to flash memory, the data will be lost.
If you then turn ON the power supply to the Machine Controller, the Machine Controller will start with the factory default conditions.



Format

The format of this instruction is shown below.

FLASH-OP	
[B] Execute	[B] Busy
M000000	M000002
[B] Reserve	[B] Complete
M000001	M000003
[W] Option	[B] Error
M000001	M000004
[A] Param	[W] Error Code
M000010	M000002
	[W] Status
	M000003

Icon:

Key entry: FLASH-OP

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute (Flash operation command)	○	×	×	×	×	×	×	×	×
Reserve (Reserved for system)	○	×	×	×	×	×	×	×	×
Option (Option settings)	×	○	×	×	×	×	×	○	○
Param (First address of parameter table)	×	×	×	×	×	×	○*1	×	×
Busy (Executing)	○*2	×	×	×	×	×	×	×	×
Complete (Execution completed)	○*2	×	×	×	×	×	×	×	×
Error	○*2	×	×	×	×	×	×	×	×
Error Code (Error code)	×	○	×	×	×	×	×	○	○
Status (Comparison result)	×	○	×	×	×	×	×	○	○

*1. M, G, or D register only.

*2. C and # registers cannot be used.

The following table describes each input and output item.

I/O Item	Description	I/O
Execute (Flash operation command)	The flash operation instruction is started when this command is turned ON. This command must be kept ON while the instruction is in execution.	IN
Reserve (Reserved for system)	–	–
Option (Option settings)	Command Option Bit Settings Bit D: CPU Operation; 0: Execute in CPU RUN status, 1: Execute with CPU stopped. Bit E: Verify Disable/Enable; 0: Disable, 1: Enable Bit F: Flash Save Disable/Enable; 0: Disable, 1: Enable The other bits are not used. Any settings in the other bits are ignored.	IN
Param (First address of parameter table)	First address of function workspace	IN
Busy (Executing)	Turns ON during the flash operation.	OUT
Complete (Execution completed)	Turns ON for one scan only when the flash operation is completed.	OUT
Error	Turns ON for one scan only when an error occurs.	OUT
ErrorCode (Error code)	Turns ON for one scan only when an error occurs.	OUT
Status (Comparison result)	Outputs the comparison result for one scan only after verification has been completed. Otherwise outputs 0. Comparison Result; 1: No differences, 2: One or more differences	OUT

The option settings are described in the following table.

Bit	Meaning
0 to C	Not used. (Settings will be ignored.)
D	CPU Operating Status during Flash Operation Execution (0: RUN, 1: STOP) Select the CPU operating status for execution of the flash operation. If you select 1 (STOP), the CPU will stop to execute the flash operation and then the CPU will start again when execution of the flash operation has been completed. The CPU will stop if you select 1 (STOP). Make sure that no problems will occur before you use this selection.
E	Verify Disable/Enable; 0: Disable, 1: Enable Select whether to compare flash memory and RAM data. If you select 1 (enable), the data in flash memory and RAM will be compared. If you enable both the verify and flash save operations, the data in flash memory and RAM will be compared and if any differences are found, the RAM data will be saved in the flash memory.
F	Flash Save Disable/Enable Setting Select whether to save the data to flash memory. If you select 1 (enable), the data in RAM will be saved to flash memory. If you enable both the verify and flash save operations, the data in flash memory and RAM will be compared and if any differences are found, the RAM data will be saved in the flash memory.

◆ Details on Function Workspace

This section provides the details on the function workspace. The parameter number corresponds to the word offset from the first address.

Example For example, if the first address is MA00100, set the value in MW00101 to set PARAM 01.

Parameter No.	IN/OUT	Meaning
PARAM 00	SYS	For system use #1
PARAM 01	SYS	For system use #2

■ For System Use #1 (PARAM 00)

This parameter is used by the system. Set this parameter to 0000 hex from a user program in the first scan after the power supply is turned ON. Do not modify this parameter at any other time.

■ For System Use #2 (PARAM 01)

This parameter is used by the system. Set this parameter to 0000 hex from a user program in the first scan after the power supply is turned ON. Do not modify this parameter at any other time.

■ Error Codes

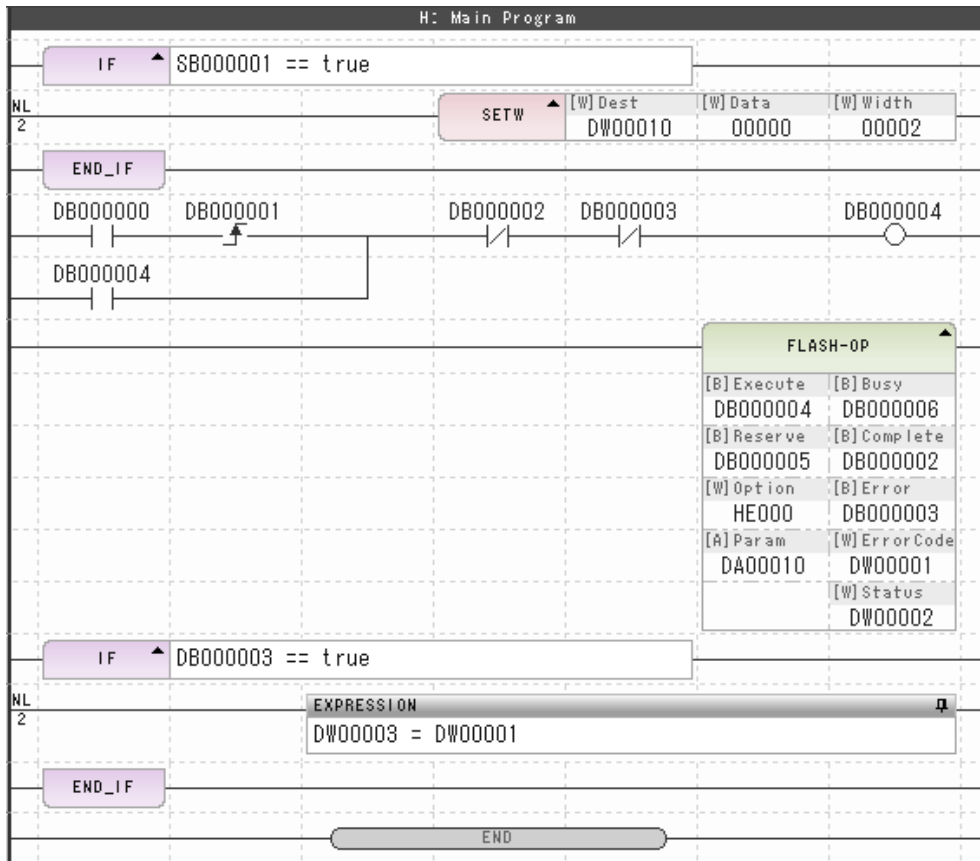
The following errors can occur.

Error Code	Meaning
0000 hex	Normal
0001 hex	Instruction duplication
0002 hex	Internal system error
0003 hex	Neither the flash save or verify operation was specified.

Programming Example

The following programming example shows how to save data to the flash memory.

The data is verified when DB000000 is turned ON. If there are any differences in the data in the flash memory and RAM, the CPU is stopped and the data in RAM is saved to flash memory. When saving the data to flash memory has been completed, the CPU is automatically started.



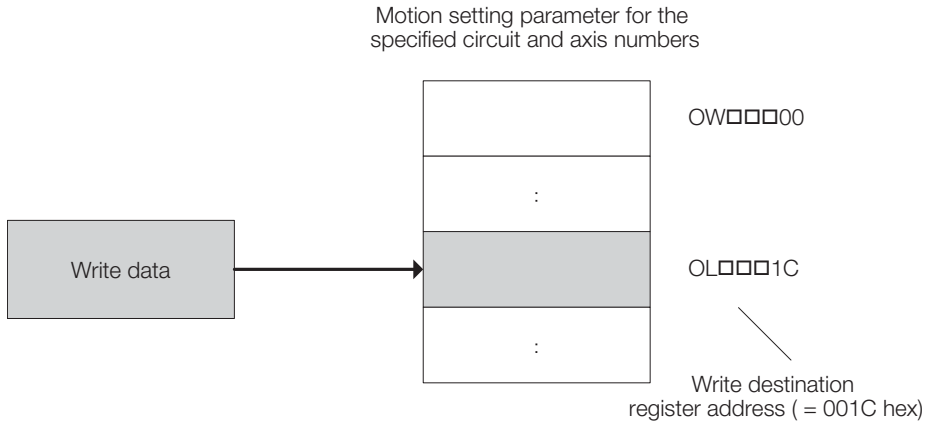
4.10.12 Write Motion Register (MOTREG-W)

This system function is used to access specified motion registers.

Values are written to a motion register by specifying the circuit number, axis number, and register address.

This function is used with motion setting parameters.

Information This function is useful for storing the same motion setting parameter for multiple axes with different circuit and axis numbers. If the STORE instruction or an EXPRESSION instruction is used to write to the motion registers, you need to consider an offset to address the circuit and axis numbers.



Format

The format of this instruction is shown below.

Icon: **MOT REG-W**
Key entry: MOTREG-W

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Axis-Inf (Axis information)	x	○	x	x	x	x	x	x	x
Reg-No (Register address)	x	○	x	x	x	x	x	x	x
Mode	x	○	x	x	x	x	x	x	x
WR-Data (Write data)	x	○	○	x	x	x	x	x	x
Error	○*	x	x	x	x	x	x	x	x
RD-Data (Read data)	x	○*	○*	x	x	x	x	x	x

* C and # registers cannot be used. These parameters may be omitted.

The following table describes each input and output item.

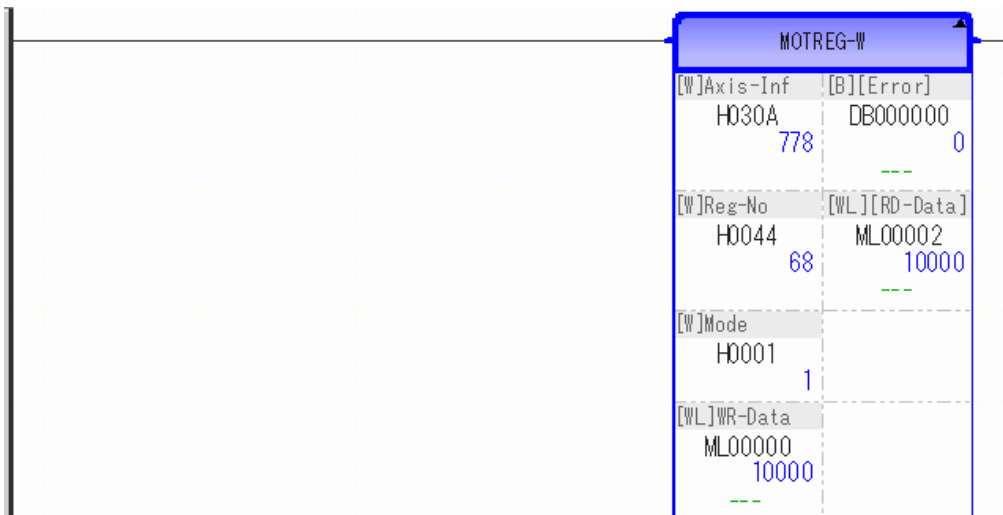
I/O Item	Description	I/O
Axis-Inf (Axis information)	Circuit number and axis number (Cir-No) Upper byte: Circuit number (01 to 10 hex) Lower byte: Axis number (01 to 10 hex)	IN
Reg-No (Register address)	Integer register: 0000 to 007F hex Double-length integer register: 0000 to 007E hex	IN
Mode	Access type and access size • Upper byte: Access type 0: Write WR-Data to specified register. 1: Write inclusive OR of specified register and WR-Data to specified register. 2: Write AND of specified register and WR-Data to specified register. Others: Write WR-Data to specified register. • Lower byte: Access size 0: Integer data 1: Double-length integer data Others: Integer data	IN
WR-Data (Write data)	If the access size for Mode is an integer and the input data type is a double-length integer, only the lower word will be used.	IN
Error	Error cause (Turns ON when an error occurs.) The register could not be written to or read from because the circuit number, axis number, or register address is out of range, or because the Module does not exist. When an error occurs, RD-Data is set to 0.	OUT
RD-Data (Read data)	This is the data that is read after writing is completed. If integer data is specified, the data is output with the sign.	OUT

Programming Example

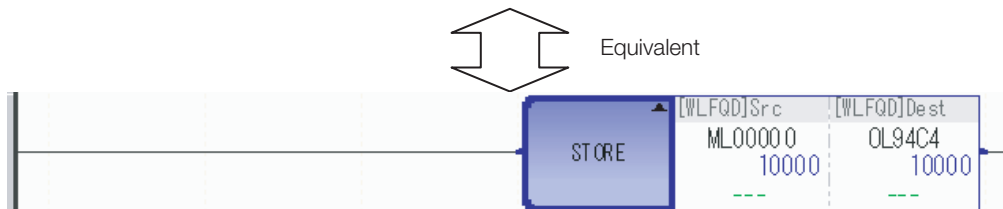
In the following programming example, the value of the write data (ML00000) is written to the STEP Travel Distance parameter in OL□□□44 for axis number 10 on circuit number 3.

Set the following items.

- Axis-Inf = 030A hex (circuit 3, axis 10)
- Register address = 0044 hex
- Mode = 0001 hex (double-length Integer)



The same result can be achieved by directly specifying the register address and storing data with the STORE instruction.



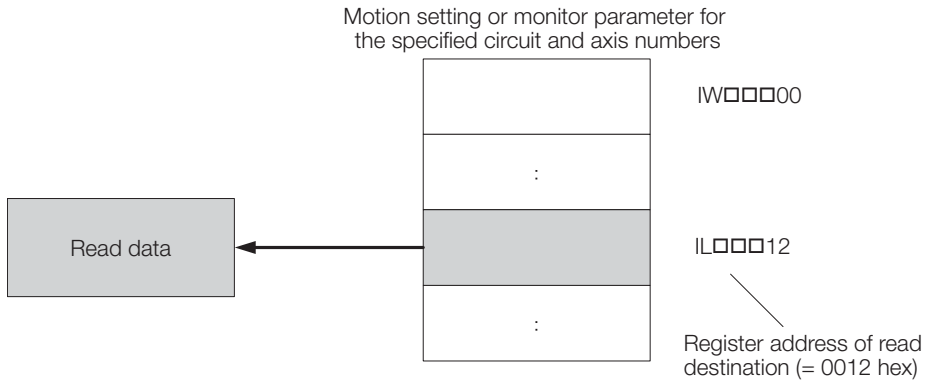
4.10.13 Read Motion Register (MOTREG-R)

This system function is used to access specified motion registers.

The value is read from a motion register by specifying the circuit number, axis number, and register address.

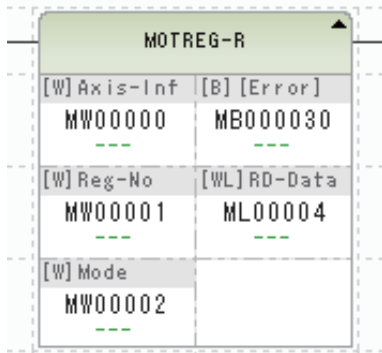
This function is to be used with motion setting parameters and motion monitor parameters.

Information This function is useful for reading the same motion setting parameter from multiple axes with different circuit and axis numbers. If the STORE instruction or an EXPRESSION instruction is used to read from the motion registers, you need to consider an offset to address the circuit and axis numbers.



Format

The format of this instruction is shown below.



Icon: **MOT**
REG-R
Key entry: MOTREG-R

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Axis-Inf (Axis information)	x	○	x	x	x	x	x	x	x
Reg-No (Register address)	x	○	x	x	x	x	x	x	x
Mode	x	○	x	x	x	x	x	x	x
Error	○*	x	x	x	x	x	x	x	x
RD-Data (Read data)	x	○*	○*	x	x	x	x	x	x

* C and # registers cannot be used. These parameters may be omitted.

The following table describes each input and output item.

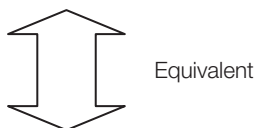
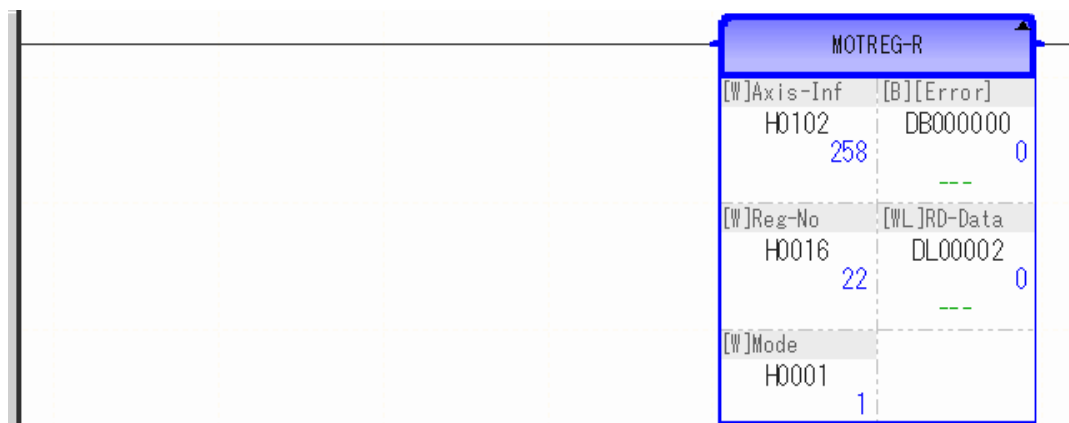
I/O Item	Description	I/O
Axis-Inf (Axis information)	Circuit number and axis number (Cir-No) Upper byte: Circuit number (01 to 10 hex) Lower byte: Axis number (01 to 10 hex)	IN
Reg-No (Register address)	Integer register: 0000 to 007F hex Double-length integer register: 0000 to 007E hex	IN
Mode	Register type and access size • Upper byte: Register type 0: I registers (motion monitor parameters) 1: O registers (motion setting parameters) Others: I registers • Lower byte: Access size 0: Integer data 1: Double-length integer data Others: Integer data	IN
Error	Error cause (Turns ON when an error occurs.) The register could not be written to or read from because the circuit number, axis number, or register address is out of range, or because the Module does not exist. When an error occurs, RD-Data is set to 0.	OUT
RD-Data (Read data)	If integer data is specified, the data is output with the sign.	OUT

Programming Example

In the following programming example, the Machine Coordinate System Feedback Position in IL8096 for axis number 2 on circuit number 1 is read.

Set the following items.

- Axis-Inf = 0102 hex (circuit 1, axis 2)
- Register address = 0016 hex
- Mode = 0001 hex (motion monitor parameter, double-length integer)



The same result can be achieved by directly specifying the register address and storing data with the STORE instruction in DL00002.



4.10.14 Import (IMPORT/IMPORTL/IMPORTLE)

Register data is imported from a USB memory device, the built-in RAM in the CPU Unit/CPU Module, or an FTP server and copied into registers.

The format of the import file is selectable between binary data (bin) and CSV data (csv).

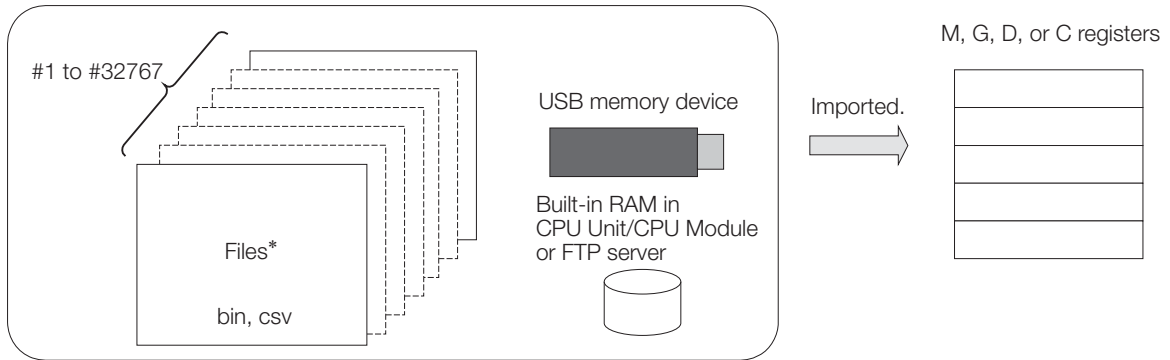
You can specify to import into M registers, G registers, D registers, or C registers.

Two of the following instructions can be executed at the same time: IMPORT, IMPORTL, and IMPORTLE.

■ Differences between IMPORT, IMPORTL, and IMPORTLE

Item		IMPORT	IMPORTL	IMPORTLE
Number of words to move		1 to 32,767	1 to 2,147,483,647	1 to 2,147,483,647
File names		Fixed	Fixed	Can be specified
Supporting versions	USB memory device	Version 1.00 or higher	Version 1.08 or higher	Version 1.30 or higher
	Built-in RAM in CPU Unit/CPU Module	Version 1.30 or higher	Version 1.30 or higher	
	FTP server			

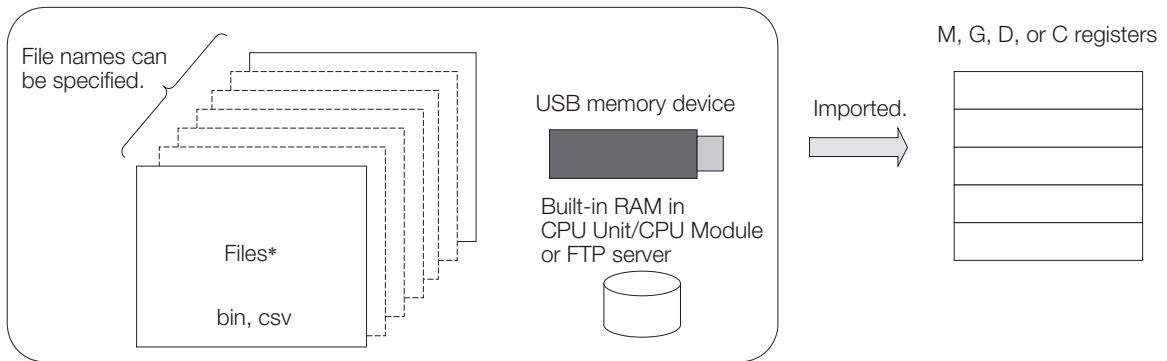
<IMPORT or IMPORTL>



* The data is imported from the following files in a USB memory device, the built-in RAM in the CPU Unit/CPU Module, or an FTP server.

```
\MP_DATA\DAT00001.BIN (CSV)
.
\MP_DATA\DAT32767.BIN (CSV)
```

<IMPORTLE>



* The data is imported from specified files in a USB memory device, the built-in RAM in the CPU Unit/CPU Module, or an FTP server.

Format

The format of this instruction is shown below.

IMPORT	
[B] Execute	[B] Busy
MB000000	MB000002
[B] Abort	[B] Complete
MB000001	MB000003
[W] Drv-No	[B] Error
MW000001	MB000004
[W] Data-No	
MW000002	
[W] Size	
MW000003	
[W] Ch-No	
MW000004	
[A] Dest	
MA00100	
[A] Param	
MA00010	

Icon:

	Import
	Import Extend
	Import with Specified File Name

Key entry: IMPORT/IMPORTL/IMPORTLE

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute (Import command)	○	×	×	×	×	×	×	×	×
Abort (Import abort command)	○	×	×	×	×	×	×	×	×
Drv-No (Drive number)	×	○	×	×	×	×	×	○	○
Data-No (Data number)	×	○	×	×	×	×	×	○	○
Size (Number of words to move)	×	○*1	○*2	×	×	×	×	○	○
Ch-No (Parallel execution channel number)	×	○	×	×	×	×	×	○	○
Dest (First destination register address)	×	×	×	×	×	×	○	×	×
Param (First address of parameter list)	×	×	×	×	×	×	○	×	×
Busy (Importing)	○	×	×	×	×	×	×	×	×
Complete (Execution of import completed)	○	×	×	×	×	×	×	×	×
Error	○	×	×	×	×	×	×	×	×

*1. For the IMPORT instruction.

*2. For the IMPORTL instruction.

The following table describes each input and output item.

<IMPORT or IMPORTL>

I/O Item	Description	I/O
Execute (Import command)	Import execution begins when this command is turned ON. This command must be kept ON while the instruction is in execution.	IN
Abort (Import abort command)	The import process is aborted when this command is turned ON.	IN
Drv-No (Drive number)	Drive number (1: USB memory device, 2: Built-in RAM in CPU Unit/CPU Module, 101 to 120: FTP server)	IN
Data-No (Data number)	Data number (1 to 32,767)	IN
Size (Number of words to move)	Number of words to move (IMPORT: 1 to 32,767, IMPORTL: 1 to 2,147,483,647)	IN
Ch-No (Parallel execution channel number)	Parallel execution channel number (1 or 2)	IN

Continued on next page.

Continued from previous page.

I/O Item	Description	I/O
Dest (First destination register address)	First destination register address (MA, GA, DA, or CA)	IN
Param (First address of parameter list)	First address of parameter list (MA, GA, or DA)	IN
Busy (Importing)	Turns ON while importing is in progress.	OUT
Complete (Execution of import completed)	Turns ON when execution of the import is completed.	OUT
Error	Turns ON when an error occurs.	OUT

<IMPORTLE>

I/O Item	Description	I/O
Execute (Import command)	Import execution begins when this command is turned ON. This command must be kept ON while the instruction is in execution.	IN
Abort (Import abort command)	The import process is aborted when this command is turned ON.	IN
Drv-No (Drive number)	Drive number (1: USB memory device, 2: Built-in RAM in CPU Unit/CPU Module, 101 to 120: FTP server)	IN
Size (Number of words to move)	Number of words to move (1 to 2,147,483,647)	IN
Ch-No (Parallel execution channel number)	Parallel execution channel number (1 or 2)	IN
Dest (First destination register address)	First destination register address (MA, GA, DA, or CA)	IN
Param (First address of parameter list)	First address of parameter list (MA, GA, or DA)	IN
FILENAME (File name)	File name (ASCII) storage register address* (MA, GA, DA, or CA)	IN
Busy (Importing)	Turns ON while importing is in progress.	OUT
Complete (Execution of import completed)	Turns ON when execution of the import is completed.	OUT
Error	Turns ON when an error occurs.	OUT

* You can specify directory levels if you select a USB memory device or the built-in RAM in the CPU Unit/CPU Module with the drive number.

Use a forward slash (/) to separate directory levels.

You cannot specify directory levels if you select an FTP server with the drive number. Specify only the file name.

The following restrictions apply to file names, including directory specifications.

- USB memory device or built-in RAM in CPU Unit/CPU Module: 250 characters max.
- FTP server: 32 characters max.

* Always delineate the end of the file name with a 0 (NULL character).

◆ Parameter Details

This section describes the parameters in detail.

<IMPORT>

Address	Data Type	Parameter No.	IN/OUT	Description
0	W	PARAM00	OUT	Processing Result
1	W	PARAM01	IN	Format
2	W	PARAM02	IN	Number of offset lines in the CSV file
3	W	PARAM03	IN	Word offset for data in the file
4	W	PARAM04	OUT	Reserved for system.
5	W	PARAM05	OUT	Reserved for system.

<IMPORTL or IMPORTLE>

Address	Data Type	Parameter No.	IN/OUT	Description
0	W	PARAM00	OUT	Processing Result
1	W	PARAM01	IN	Format
2	L	PARAM02	IN	Number of offset lines in the CSV file
4	L	PARAM03	IN	Word offset for data in the file
6	W	PARAM04	OUT	Reserved for system.
7	W	PARAM05	OUT	Reserved for system.

■ Processing Result (PARAM00)

This parameter reports the processing result of the IMPORT, IMPORTL, or IMPORTLE instruction.

- 00□□ hex: Busy (Busy)
- 10□□ hex: Completed (Complete)
- 8□□□ hex: Error occurred (Error)
The following errors can occur.

Error Code	Description
8101 hex	Drive number out of range error
8102 hex	Data number out of range error
8103 hex	Number of words to move out of range error
8104 hex	Parallel execution channel number out of range error
8105 hex	Destination or source register address out of range error
8106 hex	Format type out of range error
8107 hex	Open type out of range error
8108 hex	Word offset for data in the file out of range error
8109 hex	First address of parameter list out of range error
810A hex	Number of offset lines in the file out of range error
810C hex	File name error
810E hex	FTP reception error
8201 hex	No USB memory device
8202 hex	File open error
8203 hex	File seek error
8204 hex	File write error
8205 hex	File read error
8206 hex	File close error
8301 hex	Cannot be processed because there are too many files
8302 hex	File I/O timeout

■ Format Type (PARAM01)

This parameter sets the format of the import file.

To import register list data from the MPE720, set the format to 2.

1: Imports data from a binary file (DAT□□□□□.BIN).

The □□□□□ is set with the numeric value specified for the Data-No.

2: Imports data from a CSV file (DAT□□□□□.CSV).

The □□□□□ is set with the numeric value specified for the Data-No.

■ Number of offset lines in the CSV file (PARAM02)

For CSV files, specify the number of offset lines.

To import register list data from the MPE720, set the format to 2.

This parameter is ignored for binary files.

4.10.14 Import (IMPORT/IMPORTL/IMPORTLE)

■ **Word offset for data in the file (PARAM03)**

This parameter sets the number of words to offset.

<IMPORT>

The setting range is 0 to 32,766.

<IMPORTL or IMPORTLE>

The setting range depends on the software version of the MP3000-series Machine Controller. Set the value according to the following table.

Version	Setting Range
Version 1.21 or lower	0 to 32,766
Version 1.22 or higher	0 to 2,147,483,646

■ **Reserved for System (PARAM04)**

This parameter specifies the work area used by the system.

■ **Reserved for System (PARAM05)**

This parameter specifies the work area used by the system.

Programming Example

In the following programming example, the register list data in the MPE720 is imported into the MW01234 to MW01243 registers.

Refer to the following section for operating procedures for the MPE720.

Additional Information on page 4-246

EXPRESSION

```
DW00011=2; // csv
2=2
DW00012=2; // line offset
2=2
```

IMPORT

[B]Execute DB000000 0	[B]Busy DB000002 0
[B]Abort DB000001 0	[B]Complete DB000003 0
[W]Drv-No 00001 1	[B]Error DB000004 0
[W]Data-No 00001 1	
[W]Size 00010 10	
[W]Ch-No 00001 1	
[A]Dest MA01234 ---	
[A]Param DA00010 ---	

Before Instruction Execution

Register	Value
MW01233	0
MW01234	0
MW01235	0
MW01236	0
MW01237	0
MW01238	0
MW01239	0
MW01240	0
MW01241	0
MW01242	0
MW01243	0
MW01244	0



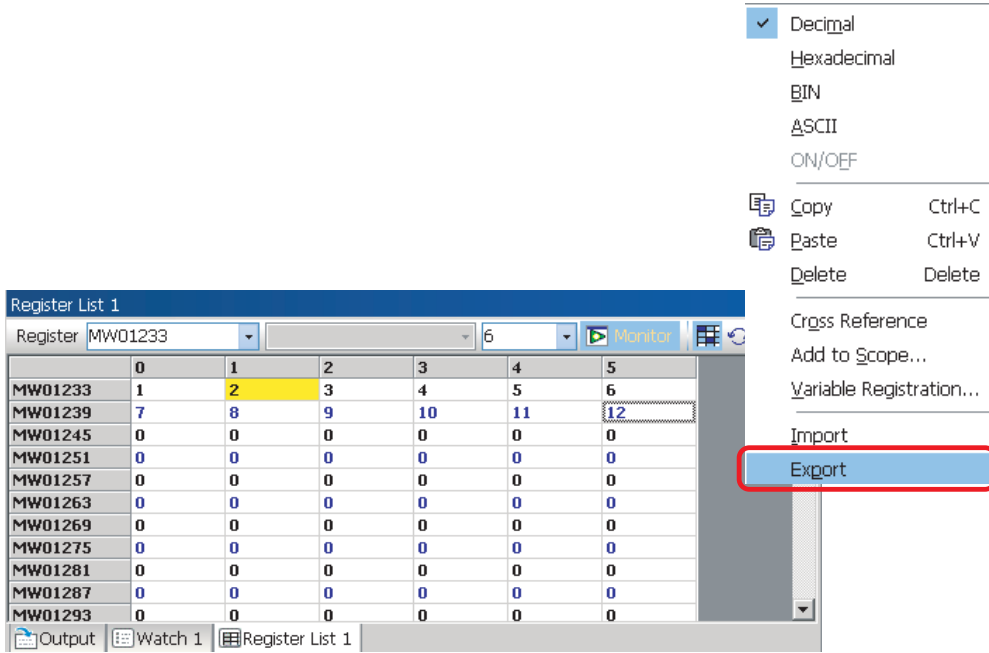
After Instruction Execution

Register	Value
MW01233	0
MW01234	2
MW01235	3
MW01236	4
MW01237	5
MW01238	6
MW01239	7
MW01240	8
MW01241	9
MW01242	10
MW01243	11
MW01244	0

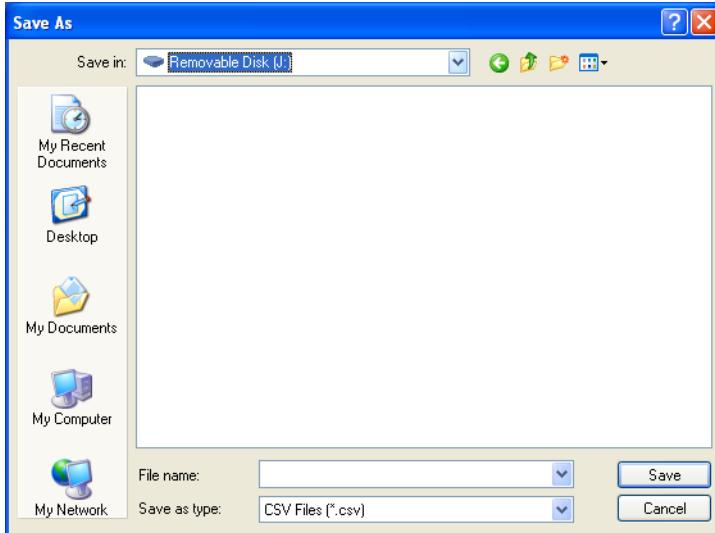
Additional Information

Use the following procedure to export the register list data on the MPE720. The following procedure is based on the programming example given earlier in this section.

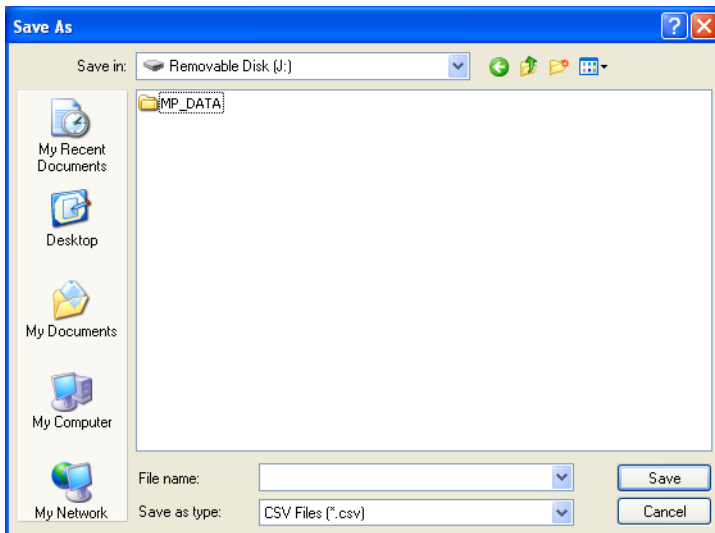
1. Insert a USB memory device into the PC.
2. Display the register list on the MPE720.
3. Right-click on the register list and select *Export* from the menu.



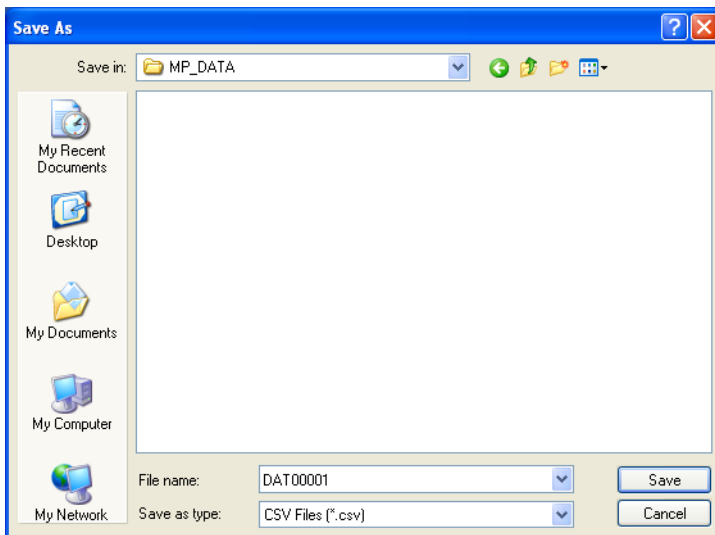
4. Select the drive for the USB memory device.



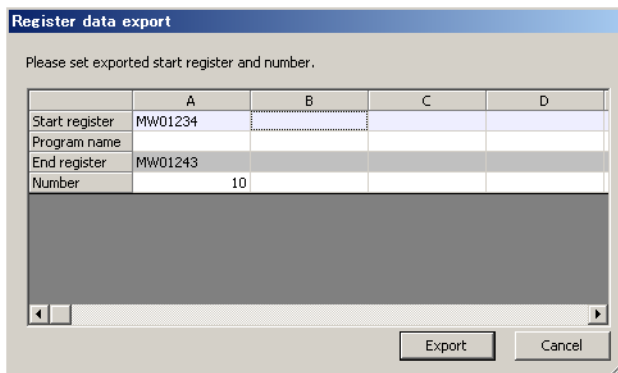
- Click the MP_DATA folder.
If the MP_DATA folder does not exist, create one.



- Enter "DAT00001" in the File name Box and click the Save Button.



- Enter "MW01234" in the Start Register Box and the number "10" in the Number Box, and then click the Export Button.



- Remove the USB memory device from the PC.
- Insert the USB memory device into the Machine Controller.

10. Wait for the USB ACCESS indicator to light.



11. Create the programming example that is given earlier in this section.

12. Execute the IMPORT, IMPORTL, or IMPORTL instruction.

4.10.15 Export (EXPORT/EXPORTL/EXPORTLE)

Register data is exported to a USB memory device, the built-in RAM in the CPU Unit/CPU Module, or an FTP server.

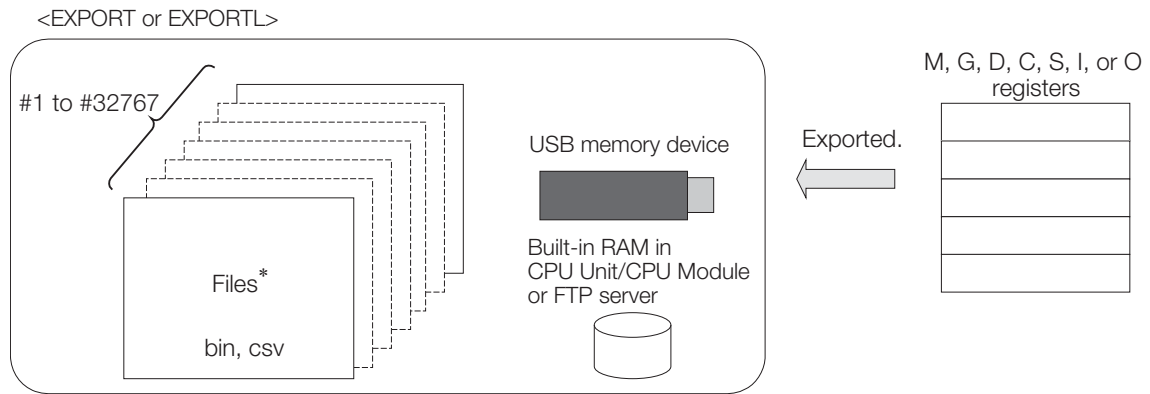
The format of the export file is selectable between binary data (bin) and CSV data (csv).

You can specify to export from M registers, G registers, D registers, C registers, S registers, I registers, or O registers.

Two of the following instructions can be executed at the same time: EXPORT, EXPORTL, or EXPORTLE.

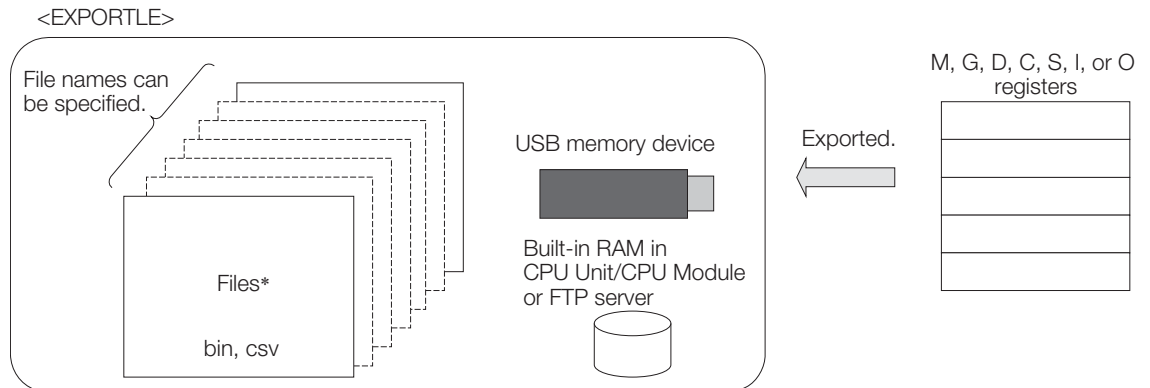
■ Differences between EXPORT, EXPORTL, and EXPOTRLE

Item		EXPORT	EXPORTL	EXPORTLE
Applicable data types		W	L	L
File names		Fixed	Fixed	Can be specified
Supporting versions	USB memory device	Version 1.00 or higher	Version 1.08 or higher	Version 1.30 or higher
	Built-in RAM in CPU Unit/CPU Module	Version 1.30 or higher	Version 1.30 or higher	
	FTP server			



* The data is exported to the following files in a USB memory device, the built-in RAM in the CPU Unit/CPU Module, or an FTP server.

\MP_DATA\DAT00001.BIN (CSV)
⋮
\MP_DATA\DAT32767.BIN (CSV)



* The data is exported to specified files in a USB memory device, the built-in RAM in the CPU Unit/CPU Module, or an FTP server.

Format

The format of this instruction is shown below.

EXPORT	
[B] Execute	[B] Busy
MB000000	MB000002
[B] Abort	[B] Complete
MB000001	MB000003
[W] Drv-No	[B] Error
MW000001	MB000004
[W] Data-No	
MW000002	
[W] Size	
MW000003	
[W] Ch-No	
MW000004	
[A] Src	
MA00100	
[A] Str	
MA00200	
[A] Param	
MA00010	

Icon:

EX PORT	Export
EX PORT L	Export Extend
EX PORT LE	Export with Specified File Name

Key entry: EXPORT/EXPORTL/EXPORTLE

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute (Export command)	○	×	×	×	×	×	×	×	×
Abort (Export abort command)	○	×	×	×	×	×	×	×	×
Drv-No (Drive number)	×	○	×	×	×	×	×	○	○
Data-No (Data number)	×	○	×	×	×	×	×	○	○
Size (Number of words to move)	×	○*1	○*2	×	×	×	×	○	○
Ch-No (Parallel execution channel number)	×	○	×	×	×	×	×	○	○
Src (First source register address)	×	×	×	×	×	×	○	×	×
Str (Register address for text string output)	×	×	×	×	×	×	○	×	×
Param (First address of parameter list)	×	×	×	×	×	×	○	×	×
Busy (Exporting)	○	×	×	×	×	×	×	×	×
Complete (Execution of export completed)	○	×	×	×	×	×	×	×	×
Error	○	×	×	×	×	×	×	×	×

*1. For the EXPORT instruction.

*2. For the EXPORTL instruction.

4.10 System Function Instructions

4.10.15 Export (EXPORT/EXPORTL/EXPORTLE)

The following table describes each input and output item.

<EXPORT or EXPORTL>

I/O Item	Description	I/O
Execute (Export command)	Export execution begins when this command is turned ON. This command must be kept ON while the instruction is in execution.	IN
Abort (Export abort command)	The export process is aborted when this command is turned ON.	IN
Drv-No (Drive number)	Drive number (1: USB memory device, 2: Built-in RAM in CPU Unit/CPU Module, 101 to 120: FTP server)	IN
Data-No (Data number)	Data number (1 to 32,767)	IN
Size (Number of words to move)	Number of words to move (EXPORT: 1 to 32,767, EXPORTL: 1 to 2,147,483,647)	IN
Ch-No (Parallel execution channel number)	Parallel execution channel number (1 or 2)	IN
Src (First source register address)	First source register address (MA, GA, DA, CA, SA, IA, or OA)	IN
Str (Register address for text string output)	Register address for text string output ^{*1*2} (MA, GA, DA, or CA)	IN
Param (First address of parameter list)	First address of parameter list (MA, GA, or DA)	IN
Busy (Exporting)	Turns ON while exporting is in progress.	OUT
Complete (Execution of export completed)	Turns ON when execution of the export is completed.	OUT
Error	Turns ON when an error occurs.	OUT

*1. Valid for CSV files. This item is ignored for binary files.

*2. Always delineate the end of a string with a 0 (NULL character).

<EXPORTLE>

I/O Item	Description	I/O
Execute (Export command)	Export execution begins when this command is turned ON. This command must be kept ON while the instruction is in execution.	IN
Abort (Export abort command)	The export process is aborted when this command is turned ON.	IN
Drv-No (Drive number)	Drive number (1: USB memory device, 2: Built-in RAM in CPU Unit/CPU Module, 101 to 120: FTP server)	IN
Size (Number of words to move)	Number of words to move (1 to 2,147,483,647)	IN
Ch-No (Parallel execution channel number)	Parallel execution channel number (1 or 2)	IN
Src (First source register address)	First source register address (MA, GA, DA, CA, SA, IA, or OA)	IN
Str (Register address for text string output)	Register address for text string output ^{*1*2} (MA, GA, DA, or CA)	IN
Param (First address of parameter list)	First address of parameter list (MA, GA, or DA)	IN
FILENAME (File name)	File name (ASCII) storage register address ^{*3} (MA, GA, DA, or CA)	IN
Busy (Exporting)	Turns ON while exporting is in progress.	OUT
Complete (Execution of export completed)	Turns ON when execution of the export is completed.	OUT
Error	Turns ON when an error occurs.	OUT

*1. Valid for CSV files. This item is ignored for binary files.

*2. Always delineate the end of a string with a 0 (NULL character).

*3. You can specify directory levels if you select a USB memory device or the built-in RAM in the CPU Unit/CPU Module with the drive number. Use a forward slash (/) to separate directory levels. You cannot specify directory levels if you select an FTP server with the drive number. Specify only the file name. The following restrictions apply to file names, including directory specifications.

- USB memory device or built-in RAM in CPU Unit/CPU Module: 250 characters max.
- FTP server: 32 characters max.

Always delineate the end of the file name with a 0 (NULL character).

◆ Parameter Details

This section describes the parameters in detail.

<EXPORT>

Address	Data Type	Parameter No.	IN/OUT	Description
0	W	PARAM00	OUT	Processing Result
1	W	PARAM01	IN	Format
2	W	PARAM02	IN	File open type
3	W	PARAM03	IN	Word offset for data in the file
4	W	PARAM04	OUT	Reserved for system.
5	W	PARAM05	OUT	Reserved for system.

<EXPORTL or EXPORTLE>

Address	Data Type	Parameter No.	IN/OUT	Description
0	W	PARAM00	OUT	Processing Result
1	W	PARAM01	IN	Format
2	L	PARAM02	IN	File open type
4	L	PARAM03	IN	Word offset for data in the file
6	W	PARAM04	OUT	Reserved for system.
7	W	PARAM05	OUT	Reserved for system.

■ Processing Result (PARAM00)

This parameter reports the processing result of the EXPORT, EXPORTL, or EXPORTLE instruction.

- 00□□ hex: Busy (Busy)
- 10□□ hex: Completed (Complete)
- 8□□□ hex: Error (Error)

The following errors can occur.

Error Code	Meaning
8101 hex	Drive number out of range error
8102 hex	Data number out of range error
8103 hex	Number of words to move out of range error
8104 hex	Parallel execution channel number out of range error
8105 hex	Destination or source register address out of range error
8106 hex	Format type out of range error
8107 hex	Open type out of range error
8108 hex	Word offset for data in the file out of range error
8109 hex	First address of parameter list out of range error
810B hex	Text string error (NULL character not detected)
810C hex	File name error
810D hex	FTP transmission error
8201 hex	No USB memory device
8202 hex	File open error
8203 hex	File seek error
8204 hex	File write error
8205 hex	File read error
8206 hex	File close error
8301 hex	Cannot be processed because there are too many files
8302 hex	File I/O timeout

4.10.15 Export (EXPORT/EXPORTL/EXPORTLE)

■ **Format Type (PARAM01)**

This parameter sets the format of the export file.

To export register list data from the MPE720, set this parameter to 2.

1: Exports data to a binary file (DAT□□□□□.BIN).

The □□□□□ is set with the numeric value specified for the Data-No.

2: Exports data to a CSV file (DAT□□□□□.CSV).

The □□□□□ is set with the numeric value specified for the Data-No.

■ **File Open Type (PARAM02)**

This parameter sets the file open type for binary files.

1: Create and export to a new file.

2: Export to an existing file.

Select this type to change to only certain portions of existing data.

For CSV files, set this parameter to 1.

■ **Word offset for data in the file (PARAM03)**

For binary files, specify the number of offset words.

This parameter is ignored for CSV files.

<EXPORT>

The setting range is 0 to 32,766.

<EXPORTL or EXPORTLE>

The setting range depends on the software version of the MP3000-series Machine Controller. Set the value according to the following table.

Version	Setting Range
Version 1.21 or lower	0 to 32,766
Version 1.22 or higher	0 to 2,147,483,646

■ **Reserved for System (PARAM04)**

This parameter specifies the work area used by the system.

■ **Reserved for System (PARAM05)**

This parameter specifies the work area used by the system.

Programming Example

In the following programming example, the data from the MW01234 to MW01243 registers is exported to a CSV file.

Register Data

Register	Value
MW01233	1
MW01234	2
MW01235	3
MW01236	4
MW01237	5
MW01238	6
MW01239	7
MW01240	8
MW01241	9
MW01242	10
MW01243	11
MW01244	12



File in USB Memory Device (DAT00001.CSV)

Contents of Text File
MW1234 ↵
↵
00002 ↵
00003 ↵
00004 ↵
00005 ↵
00006 ↵
00007 ↵
00008 ↵
00009 ↵
00010 ↵
00011 ↵

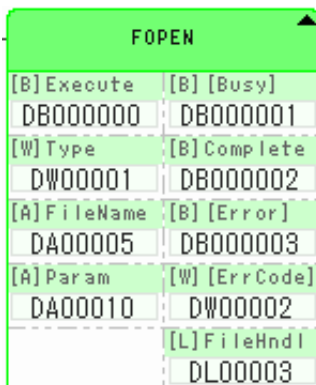
4.11 Storage Operation Instructions

4.11.1 Open File (FOPEN)

The file with the specified name is opened. When this instruction is executed, a file handle for specifying the file in other instructions is output.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute	○ ^{*1}	×	×	×	×	×	×	×	×
Type	×	○	×	×	×	×	×	×	○
FileName	×	×	×	×	×	×	○ ^{*2}	×	×
Param	×	×	×	×	×	×	○ ^{*3}	×	×
Busy ^{*4}	○ ^{*1}	×	×	×	×	×	×	×	×
Complete	○ ^{*1}	×	×	×	×	×	×	×	×
Error ^{*4}	○ ^{*1}	×	×	×	×	×	×	×	×
ErrCode ^{*4}	×	○ ^{*1}	×	×	×	×	×	×	×
FileHndl	×	×	○ ^{*1}	×	×	×	×	×	×



*1. C and # registers cannot be used.

*2. M, G, D, or C register only.

*3. M, G, or D register only.

*4. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Execute	Execute Instruction	IN	Processing is executed on the rising edge when this bit is turned ON. The processing itself is executed even if this bit is turned OFF afterward.
Type	Open Type	IN	Refer to the following section for details on open types.  <i>Type (Open Type)</i> on page 4-256
FileName	File Name	IN	Specify the first register in which the applicable file name (drive name + folder names + file name) has been stored. Specify the folder names and file name up to 250 alphanumeric characters plus the NULL character. <ul style="list-style-type: none"> Drive name: "1:/": □□□USB memory device, "2:/": □□□Built-in RAM If the drive name is omitted, the USB memory device is selected. Folder names: The separator between folders is "/".
Param	Parameters	IN/OUT	First address of function workspace
Busy	Processing	OUT	This bit is turned ON while the function being executed. This bit is turned OFF when processing is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Complete	Processing Completed	OUT	This bit is turned ON when function execution is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Error	Error Occurred	OUT	This bit is turned ON if an error occurs during function execution. This bit is OFF when Execute (Execute Instruction) is OFF. However, this bit is turned ON when Param is outside the range of registers.
ErrCode	Error Code	OUT	This item outputs the error code. Refer to the following section for details on error codes.  <i>Appendix F Error Codes</i> The value is 0 when Execute (Execute Instruction) is OFF. However, 8000 hex (Param is outside range of registers) is output when Param is outside the range of registers.
FileHndl	File Handle	OUT	This item stores the identification data for the file that was opened. The value is 0 when Execute (Execute Instruction) is OFF.

Note: 1. If Execute is turned OFF while this instruction is being executed, the processing result cannot be obtained because the output data of the instruction is cleared.

- Note the following precautions when specifying the registers used in Param. The function cannot be correctly processed if registers overlap.
 - Make sure the registers do not overlap those of another instruction.
 - Make sure the registers do not overlap when the same instruction is used in different locations.
- Make sure the value of FileHndl (File Handle) will not be overwritten by another program. If the value of FileHndl is different, the FCLOSE instruction will not be able to close the file and it will remain open. This may result in file corruption when the power supply is turned OFF.

Parameters

Offset in Words	Data Type	Purpose	Description
0	W	IN/OUT	System use (status management)
1	W	IN/OUT	System use (status management)
2	W	IN/OUT	System use (status management)
3	W	IN/OUT	System use (status management)

Type (Open Type)

Value	Description	No Existing File When Executed	Existing File When Executed
0x0000	The file is opened as read-only. The file position starts from the beginning of the file.	Error	Normal operation
0x0001	The file is opened as write-only. The file position starts from the beginning of the file. An error will occur if the file exists and is write-protected.	Create new file	Discard existing file and create new file
0x0002	The file is opened for additional writing. The file position for writing starts from the end of the file. An error will occur if the file exists and is write-protected.	Create new file	Overwrite file
Other than above	Error	–	–

Note: Files are opened as text files.

Operation Overview

After this instruction is executed, FileHndle (File Handle) for specifying the file in other instructions is output.

The data stored in FileHndl is required when specifying the file in other instructions.

- Information**
1. If the CPU is stopped, all opened files are closed. If another instruction is being executed, files are closed after processing of the instruction is completed.
 2. Only ASCII characters can be used for file and directory names.

Create the program so that files opened with FOPEN are always closed with the FCLOSE instruction.

Files cannot be opened in the following cases:

- The directory and file do not exist.
- A file is write-protected.
- The number of files that can be simultaneously opened was exceeded.
- The target file is already opened.
- The character size exceeds the maximum value.
- The USB memory device is not installed.
- File name error (no NULL before the maximum number of characters (no NULL in range of registers)).
- Range of open type error.
- Four storage operation instructions are already being executed.
- The registers assigned to Param exceed the applicable range.

4.11.2 Close File (FCLOSE)

The specified file is closed.

Format

The format of this instruction is shown below.

FCLOSE	
[B] Execute	[B] Busy
DB000000	DB000001
[L] FileHndl	[B] Complete
DL000002	DB000002
[A] Param	[B] Error
DA000010	DB000003
	[W] ErrCode
	DW000001


I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute	○ ^{*1}	×	×	×	×	×	×	×	×
FileHndl	×	×	○ ^{*1}	×	×	×	×	×	×
Param	×	×	×	×	×	×	○ ^{*2}	×	×
Busy ^{*3}	○ ^{*1}	×	×	×	×	×	×	×	×
Complete	○ ^{*1}	×	×	×	×	×	×	×	×
Error ^{*3}	○ ^{*1}	×	×	×	×	×	×	×	×
ErrCode ^{*3}	×	○ ^{*1}	×	×	×	×	×	×	×

*1. C and # registers cannot be used.

*2. M, G, or D register only.

*3. Optional.

Details on I/O Items

I/O Item	Name	IN/OUT	Description
Execute	Execute Instruction	IN	Processing is executed on the rising edge when this bit is turned ON. The processing itself is executed even if this bit is turned OFF afterward.
FileHndl	File Handle	IN	Specify the file handle.
Param	Parameters	IN/OUT	First address of function workspace
Busy	Processing	OUT	This bit is turned ON while the function being executed. This bit is turned OFF when processing is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Complete	Processing Completed	OUT	This bit is turned ON when function execution is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Error	Error Occurred	OUT	This bit is turned ON if an error occurs during function execution. This bit is OFF when Execute (Execute Instruction) is OFF. However, this bit is turned ON when Param is outside the range of registers.
ErrCode	Error Code	OUT	This item outputs the error code. Refer to the following section for details on error codes.  <i>Appendix F Error Codes</i> The value is 0 when Execute (Execute Instruction) is OFF. However, 8000 hex (Param is outside range of registers) is output when Param is outside the range of registers.

Note: 1. If Execute is turned OFF while this instruction is being executed, the processing result cannot be obtained because the output data of the instruction is cleared.

2. Note the following precautions when specifying the registers used in Param. The function cannot be correctly processed if registers overlap.
- Make sure the registers do not overlap those of another instruction.
 - Make sure the registers do not overlap when the same instruction is used in different locations.

Operation Overview

The file specified by FileHndl (File Handle) is closed.

- Information**
1. If the USB memory device is ejected after a file is opened on the device, the file remains opened.
 2. Always use this instruction to close files opened with FOPEN.

Files cannot be closed in the following cases:

- The processing cannot be executed because the target file is being used in another instruction.
- The file cannot be saved (e.g., insufficient space on the destination or the directory was deleted).
- The target file is already closed.
- Four storage operation instructions are already being executed.
- Param is outside the range of registers.

4.11.3 Read Data from File (FREAD)

The target file and data size are specified and the data is read from the target file. The data can be read up to 2,000 bytes.

Format

The format of this instruction is shown below.

FREAD	
[B] Execute	[B] [Busy]
DB000000	DB000001
[L] FileHndl	[B] Complete
DL000001	DB000002
[W] Size	[B] [Error]
DW000003	DB000003
[W] Count	[W] [ErrCode]
DW000004	DW000005
[A] Dest	[B] [FileEnd]
DA000006	DB000004
[A] Param	[W] [RdCount]
DA000010	DW000020

Details on I/O Items


I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute	○ ^{*1}	×	×	×	×	×	×	×	×
FileHndl	×	×	○ ^{*1}	×	×	×	×	×	×
Size	×	○	×	×	×	×	×	×	○
Count	×	○	×	×	×	×	×	×	○
Dest	×	×	×	×	×	×	○ ^{*2}	×	×
Param	×	×	×	×	×	×	○ ^{*2}	×	×
Busy ^{*3}	○ ^{*1}	×	×	×	×	×	×	×	×
Complete	○ ^{*1}	×	×	×	×	×	×	×	×
Error ^{*3}	○ ^{*1}	×	×	×	×	×	×	×	×
ErrCode ^{*3}	×	○ ^{*1}	×	×	×	×	×	×	×
FileEnd ^{*3}	○ ^{*1}	×	×	×	×	×	×	×	×
RdCount ^{*3}	×	○ ^{*1}	×	×	×	×	×	×	×

*1. C and # registers cannot be used.

*2. M, G, or D register only.

*3. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Execute	Execute Instruction	IN	Processing is executed on the rising edge when this bit is turned ON. The processing itself is executed even if this bit is turned OFF afterward.
FileHndl	File Handle	IN	Specify the handle of the file to read.
Size	Block Size	IN	The size in bytes of one block of data to read (1 to 2,000).
Count	Block Count	IN	Number of blocks to read (Block Count: 1 to 2,000).
Dest	Read Data Destination	IN/OUT	Specify the register address to store the read data.
Param	Parameters	IN/OUT	First address of function workspace
Busy	Processing	OUT	This bit is turned ON while the function being executed. This bit is turned OFF when processing is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Complete	Processing Completed	OUT	This bit is turned ON when function execution is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Error	Error Occurred	OUT	This bit is turned ON if an error occurs during function execution. This bit is OFF when Execute (Execute Instruction) is OFF. However, this bit is turned ON when Param is outside the range of registers.
ErrCode	Error Code	OUT	This item outputs the error codes. Refer to the following section for details on error codes.  <i>Appendix F Error Codes</i> The value is 0 when Execute (Execute Instruction) is OFF. However, 8000 hex (Param is outside range of registers) is output when Param is outside the range of registers.
FileEnd	End of File	OUT	This bit is turned ON when the EOF (a one-byte code added after the end of a text file) is reached. This bit is OFF when Execute (Execute Instruction) is OFF.
RdCount	Read Block Count	OUT	This item stores the number of blocks that were actually read.

Note: 1. If Execute is turned OFF while this instruction is being executed, the processing result cannot be obtained because the output data of the instruction is cleared.

2. Note the following precautions when specifying the registers used in Param. The function cannot be correctly processed if registers overlap.
- Make sure the registers do not overlap those of another instruction.
 - Make sure the registers do not overlap when the same instruction is used in different locations.

Operation Overview

The data is read from the file specified by FileHndl (File Handle) at the position indicated by the file position indicator and stored in Dest (Read Data Destination). The file position indicator is moved by only the size of the data that was read. The size of data to read is calculated as Size (Block Size) × Count (Block Count). Set the size of data to read to a maximum of 2,000 bytes.

The number of blocks that were actually read is stored in RdCount. Normally RdCount = Count. If the size of the file is not a multiple of Size, the final block will not be read because it is smaller than Size, RdCount will be less than Count, and FileEnd will be turned ON. However, if the size to read (Size × Count) is smaller than the file, RdCount = Count and FileEnd is not turned ON. When this instruction is executed on an area of the file that exceeds the file size, FileEnd is turned ON.

File data cannot be read in the following cases:

- The target file cannot be accessed.
- The processing cannot be executed because the target file is being used in another instruction.
- Four storage operation instructions are already being executed.
- Param is outside the range of registers.
- Size is specified outside the range of registers.
- Count is specified outside the range of registers.
- Size × Count is outside the applicable range.
- The read destination registers are outside the applicable range.

- Information**
1. The data is handled as little-endian.
 2. If the data size from the start position to read up to the end of the file cannot be divided by the block size, the final block is not written to Dest (Read Data Destination) and FileEnd is turned ON.

4.11.4 Write Data to File (FWRITE)

The target file and data size are specified and the data is written to the target file. The data can be written up to 2,000 bytes.

Format

The format of this instruction is shown below.

FWRITE	
[B] Execute	[B] [Busy]
DB000000	DB000001
[L] FileHndl	[B] Complete
DL000001	DB000002
[W] Size	[B] [Error]
DW000003	DB000003
[W] Count	[W] [ErrCode]
DW000004	DW000005
[A] Src	[W] [WrCount]
DA000006	DW000020
[A] Param	
DA000010	

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute	○ ^{*1}	×	×	×	×	×	×	×	×
FileHndl	×	×	○ ^{*1}	×	×	×	×	×	×
Size	×	○	×	×	×	×	×	×	○
Count	×	○	×	×	×	×	×	×	○
Src	×	×	×	×	×	×	○ ^{*2}	×	×
Param	×	×	×	×	×	×	○ ^{*3}	×	×
Busy ^{*4}	○ ^{*1}	×	×	×	×	×	×	×	×
Complete	○ ^{*1}	×	×	×	×	×	×	×	×
Error ^{*4}	○ ^{*1}	×	×	×	×	×	×	×	×
ErrCode ^{*4}	×	○ ^{*1}	×	×	×	×	×	×	×
WrCount ^{*4}	×	○ ^{*1}	×	×	×	×	×	×	×


*1. C and # registers cannot be used.

*2. M, G, D, or C register only.

*3. M, G, or D register only.

*4. Optional.

Details on I/O Items

Item	Name	I/O	Description
Execute	Execute Instruction	IN	Processing is executed on the rising edge when this bit is turned ON. The processing itself is executed even if this bit is turned OFF afterward.
FileHndl	File Handle	IN	Specify the handle of the file to write.
Size	Block Size	IN	The size in bytes of one block of data to write (1 to 2,000).
Count	Block Count	IN	Number of blocks to write (Block Count: 1 to 2,000).
Src	Write Data Source	IN	Specify the register address that stores the data to write.
Param	Parameters	IN/OUT	First address of function workspace
Busy	Processing	OUT	This bit is turned ON while the function being executed. This bit is turned OFF when processing is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Complete	Processing Completed	OUT	This bit is turned ON when function execution is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Error	Error Occurred	OUT	This bit is turned ON if an error occurs during function execution. This bit is OFF when Execute (Execute Instruction) is OFF. However, this bit is turned ON when Param is outside the range of registers.
ErrCode	Error Code	OUT	This item outputs the error code. Refer to the following section for details on error codes.  <i>Appendix F Error Codes</i> The value is 0 when Execute (Execute Instruction) is OFF. However, 8000 hex (Param is outside range of registers) is output when Param is outside the range of registers.
WrCount	Write Block Count	OUT	This item outputs the number of blocks that were written.

Note: 1. If Execute is turned OFF while this instruction is being executed, the processing result cannot be obtained because the output data of the instruction is cleared.

2. Note the following precautions when specifying the registers used in Param. The function cannot be correctly processed if registers overlap.

- Make sure the registers do not overlap those of another instruction.
- Make sure the registers do not overlap when the same instruction is used in different locations.

Operation Overview

The data stored in Src (Write Data Source) is written to the file specified by FileHndl (File Handle) at the position indicated by the file position indicator. After the data is written, execute the FCLOSE (Close File) instruction to save the file.

The size of data to write is calculated as Size (Block Size) × Count (Block Count). Set the size of data to write to a maximum of 2,000 bytes.

The file position indicator is moved by only the size of the data that was written.

File data cannot be written in the following cases:

- The processing cannot be executed because the target file is being used in another instruction.
- Four storage operation instructions are already being executed.
- Param is outside the range of registers.
- Size is specified outside the range of registers.
- Count is specified outside the range of registers.
- Size × Count is outside the applicable range.
- The registers to write are outside the applicable range.

Information The data is handled as little-endian.

4.11.5 Set File Position Indicator (FSEEK)

The file position indicator is set for the specified file and data can be written to the desired position in the file.

Format

The format of this instruction is shown below.

FSEEK	
[B] Execute	[B] [Busy]
DB000000	DB000001
[L] FileHndl	[B] Complete
DL00001	DB000002
[L] Offset	[B] [Error]
DL00003	DB000003
[W] Origin	[W] [ErrCode]
DW00005	DW00006
[A] Param	
DA00010	


I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute	○*1	×	×	×	×	×	×	×	×
FileHndl	×	×	○*1	×	×	×	×	×	×
Offset	×	×	○	×	×	×	×	×	○
Origin	×	○	×	×	×	×	×	×	○
Param	×	×	×	×	×	×	○*2	×	×
Busy*3	○*1	×	×	×	×	×	×	×	×
Complete	○*1	×	×	×	×	×	×	×	×
Error*3	○*1	×	×	×	×	×	×	×	×
ErrCode*3	×	○*1	×	×	×	×	×	×	×

*1. C and # registers cannot be used.

*2. M, G, or D register only.

*3. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Execute	Execute Instruction	IN	Processing is executed on the rising edge when this bit is turned ON. The processing itself is executed even if this bit is turned OFF afterward.
FileHndl	File Handle	IN	Specify the handle of the target file.
Offset	Offset	IN	Specify the number of bytes to move from the specified Origin (Reference Position).
Origin	Reference Position	IN	Specify the reference for the offset. 0 (SEEK_SET): Start of the file 1 (SEEK_CUR): Current position in the file 2 (SEEK_END): End of the file Other than above: Error
Param	Parameters	IN/OUT	First address of function workspace
Busy	Processing	OUT	This bit is turned ON while the function being executed. This bit is turned OFF when processing is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Complete	Processing Completed	OUT	This bit is turned ON when function execution is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Error	Error Occurred	OUT	This bit is turned ON if an error occurs during function execution. This bit is OFF when Execute (Execute Instruction) is OFF. However, this bit is turned ON when Param is outside the range of registers.
ErrCode	Error Code	OUT	This item outputs the error code. Refer to the following section for details on error codes.  <i>Appendix F Error Codes</i> The value is 0 when Execute (Execute Instruction) is OFF. However, 8000 hex (Param is outside range of registers) is output when Param is outside the range of registers.

Note: 1. If Execute is turned OFF while this instruction is being executed, the processing result cannot be obtained because the output data of the instruction is cleared.

2. Note the following precautions when specifying the registers used in Param. The function cannot be correctly processed if registers overlap.
- Make sure the registers do not overlap those of another instruction.
 - Make sure the registers do not overlap when the same instruction is used in different locations.

Operation Overview

The file position indicator is set to the position at which Offset is added to the position specified by Origin (Reference Position). When Origin is SEEK_END (End of of the file), a position specified from the end of the file can be set by setting Offset to a negative value.

In the following cases, an error occurs and Error is turned ON.

- The target file cannot be accessed.
- The processing cannot be executed because the target file is being used in another instruction.
- Four storage operation instructions are already being executed.
- Param is outside the range of registers.
- A file seek error occurred.
- The offset is outside the applicable range (the file area has been exceeded).
- Origin is outside the applicable range.

4.11.6 Read Line from File to String (FGETS)

One line (1,999 characters maximum) is read from the specified file to a text string.

Format


The format of this instruction is shown below.

FGETS	
[B] Execute	[B] [Busy]
DB000000	DB000003
[B] TrimNL	[B] Complete
DB000002	DB000004
[L] FileHndl	[B] [Error]
DL000001	DB000005
[A] Dest	[W] [ErrCode]
DA000006	DW000003
[A] Param	[B] FileEnd
DA000010	DB000006
	[W] RdCount
	DW000004

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute	○*1	×	×	×	×	×	×	×	×
TrimNL	○	×	×	×	×	×	×	×	×
FileHndl	×	×	○*1	×	×	×	×	×	×
Dest	×	×	×	×	×	×	○*2	×	×
Param	×	×	×	×	×	×	○*2	×	×
Busy *3	○*1	×	×	×	×	×	×	×	×
Complete	○*1	×	×	×	×	×	×	×	×
Error *3	○*1	×	×	×	×	×	×	×	×
ErrCode *3	×	○*1	×	×	×	×	×	×	×
FileEnd	○*1	×	×	×	×	×	×	×	×
RdCount	×	○*1	×	×	×	×	×	×	×

*1. C and # registers cannot be used.
 *2. M, G, or D register only.
 *3. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Execute	Execute Instruction	IN	Processing is executed on the rising edge when this bit is turned ON. The processing itself is executed even if this bit is turned OFF afterward.
TrimNL	Limit Newline Codes	IN	TRUE: Delete Newline Codes FALSE: Do Not Delete Newline Codes
FileHndl	File Handle	IN	Specify the handle of the target file.
Dest	Read Data Destination	IN/OUT	Specify the register address to store the read data.
Param	Parameters	IN/OUT	First address of function workspace
Busy	Processing	OUT	This bit is turned ON while the function being executed. This bit is turned OFF when processing is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Complete	Processing Completed	OUT	This bit is turned ON when function execution is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Error	Error Occurred	OUT	This bit is turned ON if an error occurs during function execution. This bit is OFF when Execute (Execute Instruction) is OFF. However, this bit is turned ON when Param is outside the range of registers.
ErrCode	Error Code	OUT	This item outputs the error code. Refer to the following section for details on error codes.  <i>Appendix F Error Codes</i> The value is 0 when Execute (Execute Instruction) is OFF. However, 8000 hex (Param is outside range of registers) is output when Param is outside the range of registers.
FileEnd	End of File	OUT	This bit is turned ON when the EOF (a one-byte code added after the end of a text file) is reached. This bit is OFF when Execute (Execute Instruction) is OFF.
RdCount	Read Data Size	OUT	This item stores the data size that was read.

Note: 1. If Execute is turned OFF while this instruction is being executed, the processing result cannot be obtained because the output data of the instruction is cleared.

2. Note the following precautions when specifying the registers used in Param. The function cannot be correctly processed if registers overlap.

- Make sure the registers do not overlap those of another instruction.
- Make sure the registers do not overlap when the same instruction is used in different locations.

Operation Overview

For the data to read, only one line of data is read from the file specified by FileHndl (File Handle) at the position indicated by the file position indicator and stored in Dest (Read Data Destination).

The size of data that can be read with this instruction is up to 1,999 bytes plus the NULL character. To read a line longer than 1,999 bytes, you must split up the line by executing this instruction multiple times.

The file position indicator is moved to the next line. If the file position indicator reaches the end of the file, FileEnd (End of File) is turned ON.

When this instruction is executed on an area of the file that exceeds the file size, FileEnd is turned ON.

If TRUE (Delete Newline Codes) is selected for TrimNL (Limit Newline Codes), the newline codes (CR, LF, CRLF) are deleted from the line and then the line is stored in Dest. If FALSE (Do Not Delete Newline Codes) is selected, the size of the read data also includes the newline codes.

In the following cases, one line of the file cannot be read and Error is turned ON.

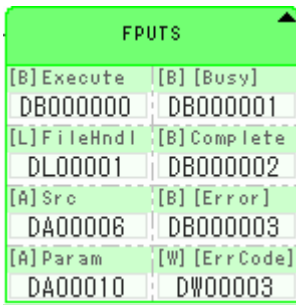
- The target file cannot be accessed.
- The processing cannot be executed because the target file is being used in another instruction.
- Four storage operation instructions are already being executed.
- Param is outside the range of registers.
- The read destination registers are outside the applicable range.
- If an out of range error (8113 hex) occurs while this instruction is being executed, the file position indicator is moved to the location that was processed before the error occurred. To execute this instruction after an out of range error has occurred, redo the processing from the location at which the file is once again opened.

4.11.7 Write String to File (FPUTS)

A text string (1,999 characters maximum) is written to the specified file.

Format


The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute	○*1	×	×	×	×	×	×	×	×
FileHndl	×	×	○*1	×	×	×	×	×	×
Src	×	×	×	×	×	×	○*2	×	×
Param	×	×	×	×	×	×	○*3	×	×
Busy *4	○*1	×	×	×	×	×	×	×	×
Complete	○*1	×	×	×	×	×	×	×	×
Error *4	○*1	×	×	×	×	×	×	×	×
ErrCode *4	×	○*1	×	×	×	×	×	×	×

*1. C and # registers cannot be used.
 *2. M, G, D, or C register only.
 *3. M, G, or D register only.
 *4. Optional.

Details on I/O Items

Item	Name	I/O	Description
Execute	Execute Instruction	IN	Processing is executed on the rising edge when this bit is turned ON. The processing itself is executed even if this bit is turned OFF afterward.
FileHndl	File Handle	IN	Specify the handle of the target file.
Src	Write Data Source	IN	Specify the register address that stores the data to write.
Param	Parameters	IN/OUT	First address of function workspace
Busy	Processing	OUT	This bit is turned ON while the function being executed. This bit is turned OFF when processing is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Complete	Processing Completed	OUT	This bit is turned ON when function execution is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Error	Error Occurred	OUT	This bit is turned ON if an error occurs during function execution. This bit is OFF when Execute (Execute Instruction) is OFF. However, this bit is turned ON when Param is outside the range of registers.
ErrCode	Error Code	OUT	This item outputs the error code. Refer to the following section for details on error codes.  <i>Appendix F Error Codes</i> The value is 0 when Execute (Execute Instruction) is OFF. However, 8000 hex (Param is outside range of registers) is output when Param is outside the range of registers.

Note: 1. If Execute is turned OFF while this instruction is being executed, the processing result cannot be obtained because the output data of the instruction is cleared.

2. Note the following precautions when specifying the registers used in Param. The function cannot be correctly processed if registers overlap.

- Make sure the registers do not overlap those of another instruction.
- Make sure the registers do not overlap when the same instruction is used in different locations.

Operation Overview

The data stored in Src (Write Data Source) is written to the file specified by FileHndl (File Handle) at the position indicated by the file position indicator. After the data is written, execute the FCLOSE (Close File) instruction to save the file.

To insert a newline, add newline codes (CR, LF) in the input text string.

The size of data that can be written at one time is up to 1,999 bytes plus the NULL character. If newline codes are added to the line, the size of the newline codes is also included.

In the following cases, the text string cannot be written to the file and Error is turned ON.

- The target file cannot be accessed.
- The processing cannot be executed because the target file is being used in another instruction.
- Four storage operation instructions are already being executed.
- Param is outside the range of registers.
- The write registers are outside the applicable range.

4.11.8 Copy File (FCOPY)

The specified file is copied.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute	○*1	×	×	×	×	×	×	×	×
Option	×	○	×	×	×	×	×	×	○
SrcFile	×	×	×	×	×	×	○*2	×	×
DstFile	×	×	×	×	×	×	○*2	×	×
Param	×	×	×	×	×	×	○*3	×	×
Busy *4	○*1	×	×	×	×	×	×	×	×
Complete	○*1	×	×	×	×	×	×	×	×
Error *4	○*1	×	×	×	×	×	×	×	×
ErrCode *4	×	○*1	×	×	×	×	×	×	×


*1. C and # registers cannot be used.

*2. M, G, D, or C register only.

*3. M, G, or D register only.


*4. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Execute	Execute Instruction	IN	Processing is executed on the rising edge when this bit is turned ON. The processing itself is executed even if this bit is turned OFF afterward.
Option	Option Settings	IN	Refer to the following section for details on option settings.  <i>Option Settings</i> on page 4-273
SrcFile	Source File Name	IN	Specify the first register in which the source file name (drive name + folder names + file name) of the write data has been stored. Specify the folder names and file name up to 250 alphanumeric characters plus the NULL character. <ul style="list-style-type: none"> • Drive name: "1:/" : □□□USB memory device, "2:/" : □□□Built-in RAM If the drive name is omitted, the USB memory device is selected. • Folder names: The separator between folders is "/".

Continued on next page.

Continued from previous page.

I/O Item	Name	I/O	Description
DstFile	Destination File Name	IN	Specify the first register in which the destination file name (drive name + folder names + file name) of the read data has been stored. Specify the folder names and file name up to 250 alphanumeric characters plus the NULL character. <ul style="list-style-type: none"> Drive name: "1:/" : □□□USB memory device, "2:/" : □□□Built-in RAM If the drive name is omitted, the USB memory device is selected. Folder names: The separator between folders is "/".
Param	Parameters	IN/OUT	First address of function workspace
Busy	Processing	OUT	This bit is turned ON while the function being executed. This bit is turned OFF when processing is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Complete	Processing Completed	OUT	This bit is turned ON when function execution is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Error	Error Occurred	OUT	This bit is turned ON if an error occurs during function execution. This bit is OFF when Execute (Execute Instruction) is OFF. However, this bit is turned ON when Param is outside the range of registers.
ErrCode	Error Code	OUT	This item outputs the error code. Refer to the following section for details on error codes.  <i>Appendix F Error Codes</i> The value is 0 when Execute (Execute Instruction) is OFF. However, 8000 hex (Param is outside range of registers) is output when Param is outside the range of registers.

Note: 1. If Execute is turned OFF while this instruction is being executed, the processing result cannot be obtained because the output data of the instruction is cleared.

2. Note the following precautions when specifying the registers used in Param. The function cannot be correctly processed if registers overlap.
- Make sure the registers do not overlap those of another instruction.
 - Make sure the registers do not overlap when the same instruction is used in different locations.

Option Settings

Bit	Meaning
0	Overwrite Permission Setting OFF: Overwriting is prohibited ON: Overwriting is permitted
1 to F	Reserved for system (set to 0).

Operation Overview

The file specified by SrcFile (Source File Name) is copied to the file specified by DstFile (Destination File Name).

If a file with the same name already exists, the file is copied according to the overwrite permission setting in Option (Option Settings). If the overwrite permission setting is set to prohibit overwriting, an error occurs and the file is not copied.

Information Only ASCII characters can be used for file and directory names.

In the following cases, the file cannot be copied and Error is turned ON.

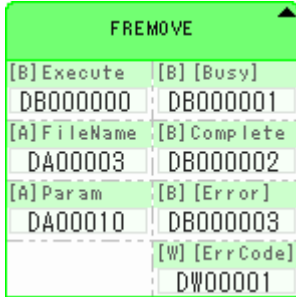
- The specified path does not exist.
- The target file cannot be accessed.
- The processing cannot be executed because the target file is being used in another instruction.
- Four storage operation instructions are already being executed.
- File name error (no NULL before the maximum number of characters or no NULL in range of registers).
- Param is outside the range of registers.
- A file with the same name already exists when overwriting is prohibited.

4.11.9 Delete File (FREMOVE)

The specified file is deleted.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute	○*1	×	×	×	×	×	×	×	×
FileName	×	×	×	×	×	×	○*2	×	×
Param	×	×	×	×	×	×	○*3	×	×
Busy *4	○*1	×	×	×	×	×	×	×	×
Complete	○*1	×	×	×	×	×	×	×	×
Error *4	○*1	×	×	×	×	×	×	×	×
ErrCode *4	×	×	○*1	×	×	×	×	×	×


- *1. C and # registers cannot be used.
- *2. M, G, D, or C register only.
- *3. M, G, or D register only.
- *4. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Execute	Execute Instruction	IN	Processing is executed on the rising edge when this bit is turned ON. The processing itself is executed even if this bit is turned OFF afterward.
FileName	File Name	IN	Specify the first register in which the applicable file name (drive name + folder names + file name) has been stored. Specify the folder names and file name up to 250 alphanumeric characters plus the NULL character. • Drive name: "1:/": □□□USB memory device, "2:/": □□□Built-in RAM If the drive name is omitted, the USB memory device is selected. • Folder names: The separator between folders is "/".
Param	Parameters	IN/OUT	First address of function workspace
Busy	Processing	OUT	This bit is turned ON while the function being executed. This bit is turned OFF when processing is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Complete	Processing Completed	OUT	This bit is turned ON when function execution is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Error	Error Occurred	OUT	This bit is turned ON if an error occurs during function execution. This bit is OFF when Execute (Execute Instruction) is OFF. However, this bit is turned ON when Param is outside the range of registers.

Continued on next page.

Continued from previous page.

I/O Item	Name	I/O	Description
ErrCode	Error Code	OUT	This item outputs the error code. Refer to the following section for details on error codes.  <i>Appendix F Error Codes</i> The value is 0 when Execute (Execute Instruction) is OFF. However, 8000 hex (Param is outside range of registers) is output when Param is outside the range of registers.

Note: 1. If Execute is turned OFF while this instruction is being executed, the processing result cannot be obtained because the output data of the instruction is cleared.

2. Note the following precautions when specifying the registers used in Param. The function cannot be correctly processed if registers overlap.

- Make sure the registers do not overlap those of another instruction.
- Make sure the registers do not overlap when the same instruction is used in different locations.

Operation Overview

The file specified by FileName (File Name) is deleted.

Information Only ASCII characters can be used for file and directory names.

In the following cases, the file cannot be deleted and Error is turned ON.

- The target file cannot be accessed.
- The processing cannot be executed because the target file is being used in another instruction.
- Four storage operation instructions are already being executed.
- Param is outside the range of registers.
- File name error (no NULL before the maximum number of characters or no NULL in range of registers).



Do not delete files that have been opened.

4.11.10 Rename File (FRENAME)

The specified file is renamed.

Format

The format of this instruction is shown below.

FRENAME	
[B] Execute	[B] [Busy]
DB000000	DB000001
[W] Option	[B] Complete
DW000001	DB000002
[A] SrcFile	[B] [Error]
DA000003	DB000003
[A] DstFile	[W] [ErrCode]
DA000006	DW000002
[A] Param	
DA000010	

4.11 Storage Operation Instructions

4.11.10 Rename File (FRENAME)

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute	○ ^{*1}	×	×	×	×	×	×	×	×
Option	×	○	×	×	×	×	×	×	○
SrcFile	×	×	×	×	×	×	○ ^{*2}	×	×
DstFile	×	×	×	×	×	×	○ ^{*2}	×	×
Param	×	×	×	×	×	×	○ ^{*3}	×	×
Busy ^{*4}	○ ^{*1}	×	×	×	×	×	×	×	×
Complete	○ ^{*1}	×	×	×	×	×	×	×	×
Error ^{*4}	○ ^{*1}	×	×	×	×	×	×	×	×
ErrCode ^{*4}	×	○ ^{*1}	×	×	×	×	×	×	×



*1. C and # registers cannot be used.

*2. M, G, D, or C register only.

*3. M, G, or D register only.

*4. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Execute	Execute Instruction	IN	Processing is executed on the rising edge when this bit is turned ON. The processing itself is executed even if this bit is turned OFF afterward.
Option	Option Settings	IN	Refer to the following section for details on option settings.  <i>Option Settings</i> on page 4-273
SrcFile	Source File Name	IN	Specify the first register in which the source file name (drive name + folder names + file name) of the write data has been stored. Specify the folder names and file name up to 250 alphanumeric characters plus the NULL character. <ul style="list-style-type: none"> Drive name: "1:/" : <input type="checkbox"/><input type="checkbox"/><input type="checkbox"/> USB memory device, "2:/" : <input type="checkbox"/><input type="checkbox"/><input type="checkbox"/> Built-in RAM If the drive name is omitted, the USB memory device is selected. Folder names: The separator between folders is "/".
DstFile	Destination File Name	IN	Specify the first register in which the destination file name (drive name + folder names + file name) of the read data has been stored. Specify the folder names and file name up to 250 alphanumeric characters plus the NULL character. <ul style="list-style-type: none"> Drive name: "1:/" : <input type="checkbox"/><input type="checkbox"/><input type="checkbox"/> USB memory device, "2:/" : <input type="checkbox"/><input type="checkbox"/><input type="checkbox"/> Built-in RAM If the drive name is omitted, the USB memory device is selected. Folder names: The separator between folders is "/".
Param	Parameters	IN/OUT	First address of function workspace
Busy	Processing	OUT	This bit is turned ON while the function being executed. This bit is turned OFF when processing is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Complete	Processing Completed	OUT	This bit is turned ON when function execution is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Error	Error Occurred	OUT	This bit is turned ON if an error occurs during function execution. This bit is OFF when Execute (Execute Instruction) is OFF. However, this bit is turned ON when Param is outside the range of registers.
ErrCode	Error Code	OUT	This item outputs the error code. Refer to the following section for details on error codes.  <i>Appendix F Error Codes</i> The value is 0 when Execute (Execute Instruction) is OFF. However, 8000 hex (Param is outside range of registers) is output when Param is outside the range of registers.

Note: 1. If Execute is turned OFF while this instruction is being executed, the processing result cannot be obtained because the output data of the instruction is cleared.

2. Note the following precautions when specifying the registers used in Param. The function cannot be correctly processed if registers overlap.
- Make sure the registers do not overlap those of another instruction.
 - Make sure the registers do not overlap when the same instruction is used in different locations.

Option Settings

Bit	Meaning
0	Overwrite Permission Setting OFF: Overwriting is prohibited ON: Overwriting is permitted
1 to F	Reserved for system (set to 0).

Operation Overview

The file specified by SrcFile (Source File Name) is renamed to the name specified by DstFile (Destination File Name). Directories can also be renamed in the same manner.

If different directories are specified by SrcFile and DstFile, the files are moved to the directory specified by DstFile. However, different drive names cannot be specified by SrcFile and DstFile.

If a file with the same name already exists, the file is copied according to the overwrite permission setting in Option (Option Settings). If the overwrite permission setting is set to prohibit overwriting, an error occurs and renaming a file, moving files, and overwriting directories cannot be executed.

Information Only ASCII characters can be used for file and directory names.

In the following cases, the file cannot be renamed and Error is turned ON.

- The specified path does not exist.
- The target file cannot be accessed.
- The processing cannot be executed because the target file is being used in another instruction.
- Four storage operation instructions are already being executed.
- Param is outside the range of registers.
- File name error (no NULL before the maximum number of characters or no NULL in range of registers).
- A file with the same name already exists when overwriting is prohibited.



Important

1. Do not change text strings used as input values while the instruction is being executed.
2. Do not access the same file with multiple storage instructions at the same time.
3. Do not rename a directory that contains files that have been opened.

4.11.11 Create Directory (DCREATE)

A directory is created with the specified name.

Format

The format of this instruction is shown below.

DCREATE	
[B] Execute	[B] [Busy]
DB000000	DB000001
[A] DirName	[B] Complete
DA000005	DB000002
[A] Param	[B] [Error]
DA000010	DB000003
	[W] [ErrCode]
	DW000001

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute	○*1	×	×	×	×	×	×	×	×
DirName	×	×	×	×	×	×	○*2	×	×
Param	×	×	×	×	×	×	○*3	×	×
Busy *4	○*1	×	×	×	×	×	×	×	×
Complete	○*1	×	×	×	×	×	×	×	×
Error *4	○*1	×	×	×	×	×	×	×	×
ErrCode *4	×	○*1	×	×	×	×	×	×	×

*1. C and # registers cannot be used.

*2. M, G, D, or C register only.

*3. M, G, or D register only.


*4. Optional.

Details on I/O Items

Item	Name	I/O	Description
Execute	Execute Instruction	IN	Processing is executed on the rising edge when this bit is turned ON. The processing itself is executed even if this bit is turned OFF afterward.
DirName	Directory Name	IN	Specify the first register in which the target directory name (drive name + folder names) has been stored. Specify the folder name up to 200 alphanumeric characters plus the NULL character. <ul style="list-style-type: none"> Drive name: "1:/" : ...USB memory device "2:/" : ...Built-in RAM (If the drive name is omitted, the USB memory device is selected.) Directory name: The separator between directories is "/".
Param	Parameters	IN/OUT	First address of function workspace
Busy	Processing	OUT	This bit is turned ON while the function being executed. This bit is turned OFF when processing is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Complete	Processing Completed	OUT	This bit is turned ON when function execution is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Error	Error Occurred	OUT	This bit is turned ON if an error occurs during function execution. This bit is OFF when Execute (Execute Instruction) is OFF. However, this bit is turned ON when Param is outside the range of registers.

Continued on next page.

Continued from previous page.

Item	Name	I/O	Description
ErrCode	Error Code	OUT	<p>This item outputs the error code. Refer to the following section for details on error codes.</p> <p> <i>Appendix F Error Codes</i></p> <p>The value is 0 when Execute (Execute Instruction) is OFF. However, 8000 hex (Param is outside range of registers) is output when Param is outside the range of registers.</p>

Note: 1. If Execute is turned OFF while this instruction is being executed, the processing result cannot be obtained because the output data of the instruction is cleared.

2. Note the following precautions when specifying the registers used in Param. The function cannot be correctly processed if registers overlap.

- Make sure the registers do not overlap those of another instruction.
- Make sure the registers do not overlap when the same instruction is used in different locations.

Operation Overview

The directory specified by DirName (Directory Name) is created.

Information Only ASCII characters can be used for file and directory names.

In the following cases, the directory cannot be created and Error is turned ON.

- The directory name already exists.
- A path was specified that does not exist.
- The target file cannot be accessed.
- The processing cannot be executed because the target file is being used in another instruction.
- Other storage operation instructions are already being executed.
- Param is outside the range of registers.
- Directory name error (no NULL before the maximum number of characters or no NULL in range of registers).



Important

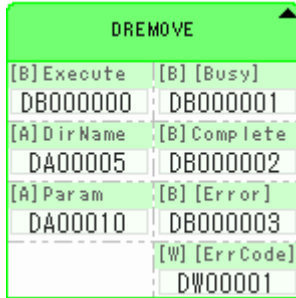
- Do not change text strings used as input values while the instruction is being executed.
- Do not access the same file with multiple storage instructions at the same time.
- Do not rename a directory that contains files that have been opened.
- The DCREATE instruction and DREMOVE instruction cannot be executed at the same time.

4.11.12 Delete Directory (DREMOVE)

The specified directory is deleted. All files and subdirectories inside the directory are deleted.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute	O*1	x	x	x	x	x	x	x	x
DirName	x	x	x	x	x	x	O*2	x	x
Param	x	x	x	x	x	x	O*3	x	x
Busy*4	O*1	x	x	x	x	x	x	x	x
Complete	O*1	x	x	x	x	x	x	x	x
Error*4	O*1	x	x	x	x	x	x	x	x
ErrCode*4	x	O*1	x	x	x	x	x	x	x


- *1. C and # registers cannot be used.
- *2. M, G, D, or C register only.
- *3. M, G, or D register only.
- *4. Optional.

Details on I/O Items

Item	Name	I/O	Description
Execute	Execute Instruction	IN	Processing is executed on the rising edge when this bit is turned ON. The processing itself is executed even if this bit is turned OFF afterward.
DirName	Directory Name	IN	Specify the first register in which the target directory name (drive name + folder names) has been stored. Specify the folder name up to 200 alphanumeric characters plus the NULL character. <ul style="list-style-type: none"> • Drive name: "1:/" : ...USB memory device "2:/" : ...Built-in RAM (If the drive name is omitted, the USB memory device is selected.) • Directory name: The separator between directories is "/".
Param	Parameters	IN/OUT	First address of function workspace
Busy	Processing	OUT	This bit is turned ON while the function being executed. This bit is turned OFF when processing is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Complete	Processing Completed	OUT	This bit is turned ON when function execution is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Error	Error Occurred	OUT	This bit is turned ON if an error occurs during function execution. This bit is OFF when Execute (Execute Instruction) is OFF. However, this bit is turned ON when Param is outside the range of registers.

Continued on next page.

Continued from previous page.

Item	Name	I/O	Description
ErrCode	Error Code	OUT	This item outputs the error code. Refer to the following section for details on error codes.  <i>Appendix F Error Codes</i> The value is 0 when Execute (Execute Instruction) is OFF. However, 8000 hex (Param is outside range of registers) is output when Param is outside the range of registers.

Note: 1. If Execute is turned OFF while this instruction is being executed, the processing result cannot be obtained because the output data of the instruction is cleared.

2. Note the following precautions when specifying the registers used in Param. The function cannot be correctly processed if registers overlap.

- Make sure the registers do not overlap those of another instruction.
- Make sure the registers do not overlap when the same instruction is used in different locations.

Operation Overview

The directory specified by DirName (Directory Name) is deleted. All files and subdirectories inside the directory are deleted.

Information Only ASCII characters can be used for file and directory names.

In the following cases, the directory cannot be deleted and Error is turned ON.

- The target file cannot be accessed.
- The processing cannot be executed because the target file is being used in another instruction.
- Other storage operation instructions are already being executed.
- Param is outside the range of registers.
- Directory name error (no NULL before the maximum number of characters or no NULL in range of registers).



Important

- Do not access the same file with multiple storage instructions at the same time.
- Do not rename a directory that contains files that have been opened.
- The DCREATE instruction and DREMOVE instruction cannot be executed at the same time.
- Do not change text strings used as input values while the instruction is being executed.

4.11.13 Send File to FTP Server (FTPPUT)

The specified file is transferred to the FTP server.

Format

The format of this instruction is shown below.

FTPPUT	
[B] Execute	[B] [Busy]
DB000000	DB000001
[W] Drv-No	[B] Complete
DW000001	DB000002
[W] Option	[B] [Error]
DW000002	DB000003
[A] SrcFile	[W] [ErrCode]
DA000005	DW000003
[A] Param	
DA000010	

4.11 Storage Operation Instructions

4.11.13 Send File to FTP Server (FTPPUT)

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Execute	○*1	×	×	×	×	×	×	×	×
Drv-No	×	○	×	×	×	×	×	×	○
Option	×	○	×	×	×	×	×	×	○
SrcFile	×	×	×	×	×	×	○*2	×	×
Param	×	×	×	×	×	×	○*3	×	×
Busy *4	○*1	×	×	×	×	×	×	×	×
Complete	○*1	×	×	×	×	×	×	×	×
Error *4	○*1	×	×	×	×	×	×	×	×
ErrCode *4	×	○*1	×	×	×	×	×	×	×



*1. C and # registers cannot be used.

*2. M, G, D, or C register only.

*3. M, G, or D register only.

*4. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Execute	Execute Instruction	IN	Processing is executed on the rising edge when this bit is turned ON. The processing itself is executed even if this bit is turned OFF afterward.
Drv-No	Drive Number	IN	Destination drive number (101 to 120: FTP server) Configure these settings in "FTP Client Settings".
Option	Option Settings	IN	Refer to the following section for details on option settings.  <i>Option Settings</i> on page 4-279
SrcFile	Source File Name	IN	Specify the first register in which the source file name (drive name + folder names + file name) has been stored. <ul style="list-style-type: none"> Drive name: "1:/" : <input type="checkbox"/><input type="checkbox"/><input type="checkbox"/> USB memory device, "2:/" : <input type="checkbox"/><input type="checkbox"/><input type="checkbox"/> Built-in RAM If the drive name is omitted, the USB memory device is selected. Folder names: The separator between folders is "/". The maximum number of characters for the path (including the drive name, folder names, and separators): When "1:/" or "2:/" is added to the drive name, the maximum number of characters is 61 characters plus the NULL character. When the drive name is omitted, the maximum number of characters is 58 characters plus the NULL character. Specify the file name up to 31 characters plus the NULL character.
Param	Parameters	IN/OUT	First address of function workspace
Busy	Processing	OUT	This bit is turned ON while the function being executed. This bit is turned OFF when processing is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Complete	Processing Completed	OUT	This bit is turned ON when function execution is completed. This bit is OFF when Execute (Execute Instruction) is OFF.
Error	Error Occurred	OUT	This bit is turned ON if an error occurs during function execution. This bit is OFF when Execute (Execute Instruction) is OFF. However, this bit is turned ON when Param is outside the range of registers.
ErrCode	Error Code	OUT	This item outputs the error code. Refer to the following section for details on error codes.  <i>Appendix F Error Codes</i> The value is 0 when Execute (Execute Instruction) is OFF. However, 8000 hex (Param is outside range of registers) is output when Param is outside the range of registers.

- Note: 1. If Execute is turned OFF while this instruction is being executed, the processing result cannot be obtained because the output data of the instruction is cleared.
2. Note the following precautions when specifying the registers used in Param. The function cannot be correctly processed if registers overlap.
- Make sure the registers do not overlap those of another instruction.
 - Make sure the registers do not overlap when the same instruction is used in different locations.

Option Settings

Bit	Meaning
0	Reserved for system (set to 0).
1	Setting to Delete the File after the FTP Transfer Is Completed OFF: Do not delete ON: Delete
2 to F	Reserved for system (set to 0).

Operation Overview

The file specified by SrcFile is transferred to the FTP server specified by Drv-No.

Information Only ASCII characters can be used for file and directory names.

In the following cases, the file cannot be transferred to the FTP server and Error is turned ON.

- The target file cannot be accessed.
- The processing cannot be executed because the target file is being used in another instruction.
- Other storage operation instructions are already being executed.
- Param is outside the range of registers.
- File name error (no NULL before the maximum number of characters or no NULL in range of registers).



Multiple FTPPUT instructions cannot be executed at the same time.

4.12 String Operation Instructions

4.12.1 Convert Integer to String (INT2STR)

An integer is converted to a text string.

Format

The format of this instruction is shown below.


INT2STR	
[WLQ] In	[A] Dest
DQ00020	DA00010
[W] Option	[B] Sts
DW00001	DB000000
[W] MinLen	
DW00002	

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In	×	○	○	○	×	×	×	×	○
Option	×	○	×	×	×	×	×	×	○
MinLen	×	○	×	×	×	×	×	×	×
Dest	×	×	×	×	×	×	○*1	×	×
Sts *2	○*1	×	×	×	×	×	×	×	×

*1. M, G, or D register only.

*2. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
In	Numeric Value	IN	Specify the register or numeric value to convert.
Option	Option Settings	IN	Refer to the following section for details on option settings.  <i>Option Settings</i> on page 4-280
MinLen	Minimum Number of Digits	IN	Specify the minimum number of digits (0 to 127). Leading spaces are added if the numeric value is less than the minimum number of digits.
Dest	Output Text String	OUT	Specify the register to store the output text string. Output processing is not performed when Sts (Status) is ON.
Sts	Status	OUT	Status is turned OFF when processing was performed normally. Status is turned ON when the following errors occur. <ul style="list-style-type: none"> The number of characters exceeds the maximum value. The text string to output exceeds the maximum range of registers. MinLen is outside the applicable range. The input value cannot be converted correctly (Option or MinLen are outside the applicable range).

Option Settings

Bit	Description
0	ON: Hexadecimal notation OFF: Decimal notation
1	ON: Pad upper digits with zeros if less than the maximum number of digits OFF: Do not pad upper digits with zeros if less than the maximum number of digits
2 to F	Reserved for system (set to 0).

Operation Overview

The value of In (Numeric Value) is converted to a text string and stored in Dest (Output Text String). Spaces are added to the text string if the number of digits is less than the minimum number of digits specified by MinLength (Minimum Number of Digits). Switch between decimal and hexadecimal notation with the hexadecimal notation setting in Option (Option Settings).

- In = 123, MinLen = 2, Option = 0x0000 (decimal notation and no zero padding) → Dest = "123"
- In = 123, MinLen = 7, Option = 0x0000 (decimal notation and no zero padding) → Dest = " 123"
- In = 123, MinLen = 7, Option = 0x0002 (decimal notation and zero padding) → Dest = "0000123"
- In = -123, MinLen = 7, Option = 0x0002 (decimal notation and zero padding) → Dest = "-000123"
- In = 123, MinLen = 2, Option = 0x0001 (hexadecimal notation and no zero padding) → Dest = "7B"
- In = 123, MinLen = 7, Option = 0x0001 (hexadecimal notation and no zero padding) → Dest = " 7B"
- In = 123, MinLen = 7, Option = 0x0003 (hexadecimal notation and zero padding) → Dest = "000007B"
- In = -123, MinLen = 7, Option = 0x0003 (hexadecimal notation and zero padding) → Dest = "000FF85"



Important

If the same text string data is accessed by different tasks at the same time, the data may be corrupted. Create the program so that the data is not accessed by different tasks at the same time.

4.12.2 Convert Real Number to String (REAL2STR)

A real number is converted to a text string.

Format

The format of this instruction is shown below.


REAL2STR	
[FD] In	[A] Dest
DF00020	DA00010
[W] Option	[B] Sts
DW00001	DB000000
[W] MinLen	
DW00002	
[W] DecLen	
DW00003	

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
In	x	x	x	x	○	○	x	x	○
Option	x	○	x	x	x	x	x	x	○
MinLen	x	○	x	x	x	x	x	x	○
DecLen	x	○	x	x	x	x	x	x	○
Dest	x	x	x	x	x	x	○*1	x	x
Sts *2	○*1	x	x	x	x	x	x	x	x

*1. M, G, or D register only.

*2. Optional.

Details on I/O Items

Item	Name	I/O	Description
In	Numeric Value	IN	Specify the converted register to convert.
Option	Option Settings	IN	Refer to the following section for details on option settings.  Option Settings on page 4-283
MinLen	Minimum Number of Digits	IN	Specify the minimum number of digits (0 to 327). Leading spaces are added if the numeric value is less than the minimum number of digits.
DecLen	Number of Digits in Decimal Part	IN	Specify the number of digits in the decimal part (0 to 15).
Dest	Output Text String	OUT	Specify the register to store the output text string (327 bytes maximum). Output processing is not performed when Sts (Status) is ON.
Sts	Status	OUT	Status is turned OFF when processing was performed normally. Status is turned ON when the following errors occur. <ul style="list-style-type: none"> The number of characters exceeds the maximum value. The text string to output exceeds the maximum range of registers. MinLen or DecLen is outside the applicable range. The input value cannot be converted correctly.

Option Settings

Bit	Description
0	ON: Exponent notation OFF: Decimal point notation
1	ON: Omit + sign ("- is not omitted) OFF: Do not omit + sign
2 to F	Reserved for system (set to 0).

Operation Overview

The value of In (Numeric Value) is converted to a text string and stored in Dest (Output Text String).

For MinLen (Minimum Number of Digits), set the minimum number of digits. Spaces are added to the beginning of the text string if the number of digits is less than the minimum number of digits specified by MinLength.

For DecLen (Number of Digits in Decimal Part), set the number of digits in the decimal part. The part that cannot be displayed is rounded.

Set exponent notation with Option (Option Settings).



If the same text string data is accessed by different tasks at the same time, the data may be corrupted. Create the program so that the data is not accessed by different tasks at the same time.

Important

4.12.3 Convert String to Integer (STR2INT)

A text string is converted to an integer.

Format

The format of this instruction is shown below.

STR2INT	
[A] Src	[WLQ] Out
DA00010	VAR MSG R CV_PARAM 001.STS D#000001
[B] Sts	DB000000

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src	×	×	×	×	×	×	○*1	×	×
Out	×	○*2	○*2	○*2	×	×	×	×	×
Sts *3	○*2	×	×	×	×	×	×	×	×

*1. M, G, D, or C register only.

*2. M, G, or D register only.

*3. Optional.


Details on I/O Items

I/O Item	Name	I/O	Description
Src	Input Text String	IN	Specify the first register in which the text string to input is stored.
Out	Output Value	OUT	This item outputs the integer. 0 is output when Sts (Status) is ON.
Sts	Status	OUT	Status is turned OFF when processing was performed normally. Status is turned ON when the following errors occur. <ul style="list-style-type: none"> The number of characters exceeds the maximum value. The input value cannot be converted correctly.

Operation Overview

The text string in Src (Input Text String) is converted to an integer and stored in Out (Output Value). The text string in Src can be composed of only the characters 0 to 9. Note that the text string can also be correctly converted if “+” or “-” indicating the sign is at the beginning of the text string.

- Src = “12345” → Out = 12345
- Src = “+12345” → Out= 12345
- Src = “-12345” → Out= -12345



If the same text string data is accessed by different tasks at the same time, the data may be corrupted. Create the program so that the data is not accessed by different tasks at the same time.

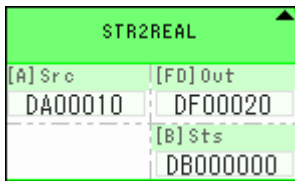
Important

4.12.4 Convert String to Real Number (STR2REAL)

A text string is converted to a real number (single-precision floating-point value or double-precision floating-point value).

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src	×	×	×	×	×	×	○*1	×	×
Out	×	×	×	×	○*2	○*2	×	×	×
Sts *3	○*2	×	×	×	×	×	×	×	×

*1. M, G, D, or C register only.

*2. M, G, or D register only.

*3. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Src	Input Text String	IN	Specify the first register in which the text string to input is stored.
Out	Output Value	OUT	This item outputs the real number. 0 is output when Sts (Status) is ON.
Sts	Status	OUT	Status is turned OFF when processing was performed normally. Status is turned ON when the following errors occur. <ul style="list-style-type: none"> • The number of characters exceeds the maximum value. • The input value cannot be converted correctly. • When FLOAT type is specified for Out (Output Value) and the absolute value of Out is larger than the range of FLOAT. Note: When the absolute value of Out is smaller than the range of FLOAT, no error occurs and "0.0" is output.

Operation Overview

The text string in Src (Input Text String) is converted to a real number and stored in Out (Output Value). Input Src with the following format.

- Sign: "+", "-", or no sign.
- Integer part: Composed of the numbers 0 to 9.
- Decimal part: From '.' (decimal point) immediately after the integer part to the exponent part. Composed of the numbers 0 to 9 up to 15 digits and can also be omitted.
- Exponent part: "e+nnn" or "e-nnn" or e can be uppercase characters. nnn is 1 to 308.

Information

Input Examples

- Src = "12.345" → Out = 12.345
- Src = "+12.345" → Out = 12.345
- Src = "-12.345" → Out = -12.345
- Src = "12" → Out = 12.0



Important

If the same text string data is accessed by different tasks at the same time, the data may be corrupted. Create the program so that the data is not accessed by different tasks at the same time.

4.12.5 Store String (STRSET)

The desired text string (including multi-byte characters) is stored in registers.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
StrIn	×	×	×	×	×	×	×	×	○
Dest	×	×	×	×	×	×	○*1	×	×
Sts*2	○*1	×	×	×	×	×	×	×	×

*1. M, G, D, or S register only.

*2. Optional.


Details on I/O Items

I/O Item	Name	I/O	Description
StrIn	Input Text String	IN	127 characters maximum (127 bytes not including the NULL character).
Dest	Output Text String	OUT	Specify the register to store the output text string. Output processing is not performed when Sts (Status) is ON.
Sts	Status	OUT	Status is turned OFF when processing was performed normally. Status is turned ON when the following errors occur. <ul style="list-style-type: none"> The number of characters exceeds the maximum value. The text string to output exceeds the maximum range of registers. The input value cannot be converted correctly.

Operation Overview

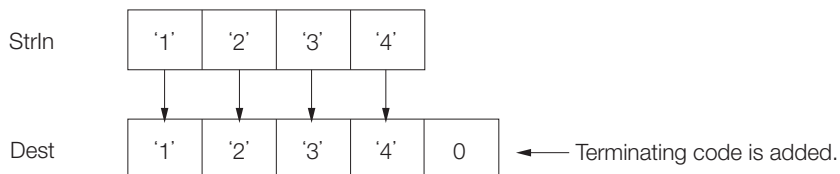
The StrIn (Input Text String) data is stored in Dest (Output Text String) as a text string. A NULL character will be automatically added to the end of the text string.

When entering newline codes, do so using escape characters such as "\n" and "\r\n".



Important If the same text string data is accessed by different tasks at the same time, the data may be corrupted. Create the program so that the data is not accessed by different tasks at the same time.

Example When StrIn = "1234":



4.12.6 Partially Delete String (STRDEL)

A part of the specified text string is deleted. The start position and size to delete can be specified.

Format

The format of this instruction is shown below.

STRDEL	
[A] Src	[A] Dest
DA00010	DA00011
[W] Pos	[B] Sts
DW00001	DB000000
[W] Size	
DW00002	

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src	x	x	x	x	x	x	○*1	x	x
Pos	x	○	x	x	x	x	x	x	○
Size	x	○	x	x	x	x	x	x	○
Dest	x	x	x	x	x	x	○*2	x	x
Sts *3	○*2	x	x	x	x	x	x	x	x

*1. M, G, D, or C register only.

*2. M, G, or D register only.

*3. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Src	Deletion Target	IN	Specify the first register in which the text string for deletion is stored.
Pos	Deletion Start Position	IN	Specify the byte position to start deleting from (0 to 1,999). When 0, the instruction deletes the data from the first byte.
Size	Deletion Size	IN	Specify the number of bytes to delete (0 to 1,999).
Dest	Deletion Result	OUT	Specify the register to store the output text string. Output processing is not performed when Sts (Status) is ON.
Sts	Status	OUT	Status is turned OFF when processing was performed normally. Status is turned ON when the following errors occur. <ul style="list-style-type: none"> The number of characters exceeds the maximum value. The text string to output exceeds the maximum range of registers. Pos or Size (Deletion Size) is outside the applicable range or larger than the number of bytes in Src. Pos + Size is larger than the number of bytes in Src.

Operation Overview

Data in the amount of the specified size is deleted from the character at the specified position in the text string specified by Src. The text string data after deletion is stored in Dest (Deletion Result).

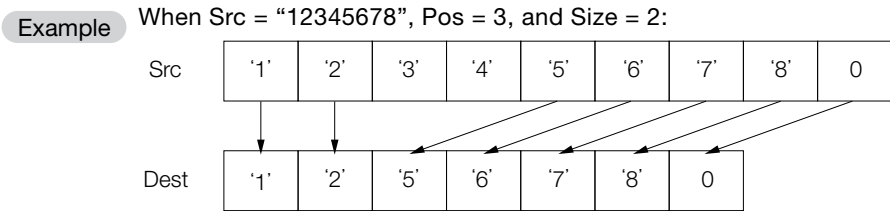
The text string after the deleted part is copied to Pos (Deletion Start Position). If there is no text string after the deletion part, a NULL character is added to the position at Pos.

- Src = "1234567", Pos = 2, Size = 2 → Dest = "14567"
- Src = "1234567", Pos = 0 (=1), Size = 2 → Dest = "34567"
- Src = "1234567", Pos = 4, Size = 10 → Dest = "123"

If the same text string data is accessed by different tasks at the same time, the data may be corrupted. Create the program so that the data is not accessed by different tasks at the same time.

Important

- Information**
1. The same registers can also be set for Src (Deletion Target) and Dest (Deletion Result).
 2. Text strings handled by this instruction are 1,999 characters maximum (1,999 bytes plus the NULL character).



4.12.7 Copy String (STRCPY)

The specified text string is copied. The size of the strings to copy can be specified.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src	×	×	×	×	×	×	○*1	×	×
Size	×	○	×	×	×	×	×	×	○
Dest	×	×	×	×	×	×	○*2	×	×
Sts *3	○*2	×	×	×	×	×	×	×	×

*1. M, G, D, or C register only.

*2. M, G, or D register only.

*3. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Src	Source	IN	Specify the first register in which the input text string is stored.
Size	Copy Size	IN	Specify the number of bytes to copy (0 to 1,999). If 0 is specified, the entire source text string is copied.
Dest	Destination	OUT	Specify the register to store the output text string. Output processing is not performed when Sts (Status) is ON.
Sts	Status	OUT	Status is turned OFF when processing was performed normally. Status is turned ON when the following errors occur. <ul style="list-style-type: none"> The number of characters exceeds the maximum value. The text string to output exceeds the maximum range of registers. Size (Copy Size) is outside the applicable range.

Operation Overview

The number of bytes specified by Size (Copy Size) is copied from the text string specified by Src (Source) and stored in Dest (Destination). If $\text{Size} \leq \text{number of bytes in Src}$, a NULL character is not added to the end of the text string. If $\text{Size} > \text{number of bytes in Src}$, the remaining characters are padded with NULL characters.

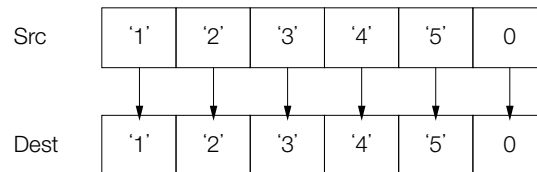


Important

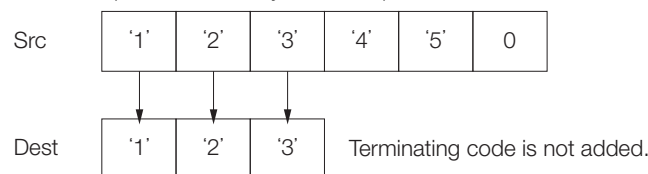
- If the same text string data is accessed by different tasks at the same time, the data may be corrupted. Create the program so that the data is not accessed by different tasks at the same time.
- Ensure that the areas for Src and Dest do not overlap. The text string cannot be copied correctly if the areas overlap.

Example When Src = "12345" (equivalent to Size = 6):

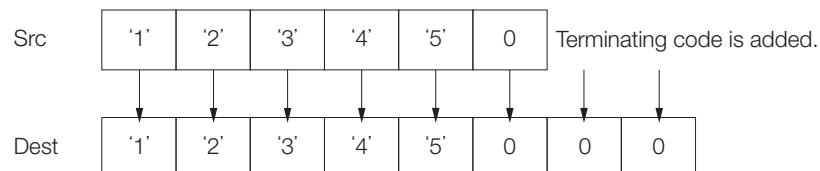
- Size = 0



- Size = 3 (< number of bytes in Src)



- Size = 8 (> number of bytes in Src)

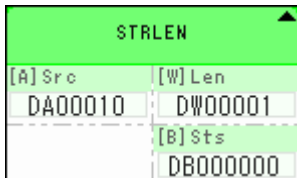


4.12.8 Get String Length (STRLEN)

The length of the text string (number of bytes) is obtained.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src	×	×	×	×	×	×	○*1	×	×
Len	×	○*2	×	×	×	×	×	×	×
Sts *3	○*2	×	×	×	×	×	×	×	×

*1. M, G, D, or C register only.

*2. M, G, or D register only.

*3. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Src	Input Text String	IN	Specify the first register in which the text string is stored.
Len	Text String Length	OUT	This item stores the number of bytes in the text string that was input (0 to 1,999). 0 is output when Sts (Status) is ON.
Sts	Status	OUT	Status is turned OFF when processing was performed normally. Status is turned ON when the number of characters exceeds the maximum value.

Note: Text String Length outputs a value between 0 and 32,767.

Operation Overview

The number of bytes (not including the NULL character) in the text string specified by Src (Input Text String) is stored in Len (Text String Length). Double-byte characters, such as JIS encoded characters, are counted as two bytes.



Important

If the same text string data is accessed by different tasks at the same time, the data may be corrupted. Create the program so that the data is not accessed by different tasks at the same time.

4.12.9 Concatenate Strings (STRCAT)

Two text strings are concatenated. The size of the text strings to concatenate can be specified.

Format

The format of this instruction is shown below.

STRCAT	
[A] Src1	[A] Dest
DA00010	DA00012
[A] Src2	[B] Sts
DA00011	DB000000
[W] Size	
DW00001	

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src1	×	×	×	×	×	×	○*1	×	×
Src2	×	×	×	×	×	×	○*1	×	×
Size	×	○	×	×	×	×	×	×	○
Dest	×	×	×	×	×	×	○*2	×	×
Sts *3	○*2	×	×	×	×	×	×	×	×

*1. M, G, D, or C register only.

*2. M, G, or D register only.

*3. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Src1	Input Text String 1	IN	First register in which Input Text String 1 is stored.
Src2	Input Text String 2	IN	First register in which Input Text String 2 is stored.
Size	Concatenation Size	IN	Specify the size in bytes of Input Text String 2 to concatenate. If 0 is specified, all of Input Text String 2 is concatenated.
Dest	Output Text String	OUT	Specify the register to store the output text string. Output processing is not performed when Sts (Status) is ON.
Sts	Status	OUT	Status is turned OFF when processing was performed normally. Status is turned ON when the following errors occur. <ul style="list-style-type: none"> The number of characters exceeds the maximum value. The text string to output exceeds the maximum range of registers. Size (Concatenation Size) is outside the applicable range. Size > number of bytes in Src2.

Operation Overview


The text string in Src2 (Input Text String 2) is concatenated to the end of Src1 (Input Text String 1). When concatenating text strings, only the size of the string specified by Size (Concatenation Size) is concatenated.

If Size is specified as 0, all of the text string in Src2 is concatenated to the end of Src1. The behavior is also the same when Size is larger than the text string in Src2.

- Src1 = "12345", Src2 = "abcde", Size = 0 → Dest = "12345abcde"
- Src1 = "12345", Src2 = "abcde", Size = 10 → Dest = "12345abcde"
- Src1 = "12345", Src2 = "abcde", Size = 2 → Dest = "12345ab"

A NULL character is added to the end of the text string.

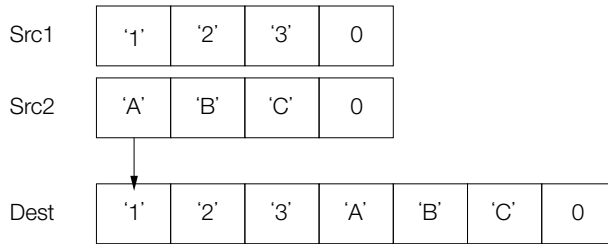
Text strings handled by this instruction are 1,999 characters maximum (1,999 bytes plus the NULL character).



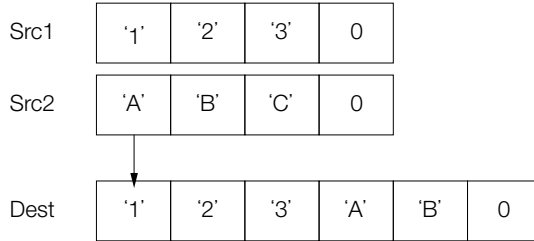
Important If the same text string data is accessed by different tasks at the same time, the data may be corrupted. Create the program so that the data is not accessed by different tasks at the same time.

Example When Src1 = "123", Src2 = "ABC":

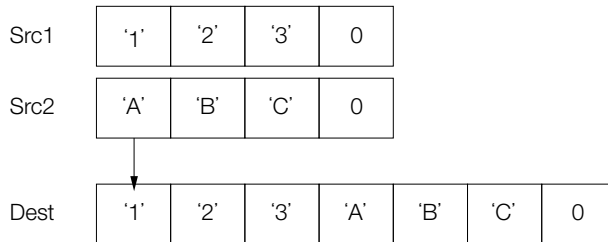
- Size = 0



- Size = 2 (< number of bytes in Src2)



- Size = 8 (> number of bytes in Src2)



4.12.10 Compare Strings (STRCMP)

Two text strings are compared. The size of the strings to compare can be specified.

Format

The format of this instruction is shown below.

STRCMP	
[A] Src1	[W] Result
DA00010	DW00002
[A] Src2	[B] Sts
DA00011	DB000000
[W] Size	
DW00001	

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src1	×	×	×	×	×	×	○ ^{*1}	×	×
Src2	×	×	×	×	×	×	○ ^{*1}	×	×
Size	×	○	×	×	×	×	×	×	○
Result	×	○ ^{*2}	×	×	×	×	×	×	×
Sts ^{*3}	○ ^{*2}	×	×	×	×	×	×	×	×

*1. M, G, D, or C register only.

*2. M, G, or D register only.

*3. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Src1	Input Text String 1	IN	First register in which Input Text String 1 is stored.
Src2	Input Text String 2	IN	First register in which Input Text String 2 is stored.
Size	Comparison Size	IN	Specify the size in bytes of the text string to compare from the beginning of the text string. If 0 is specified, the size of Input Text String 1 is compared.
Result	Comparison Result	OUT	0 is output if the text strings do not match and 1 is output if the text strings match. 0 is output when Sts (Status) is ON.
Sts	Status	OUT	Status is turned OFF when processing was performed normally. Status is turned ON when the following errors occur. <ul style="list-style-type: none"> The number of characters exceeds the maximum value. Size > number of bytes in Src1 or Size > number of bytes in Src2. Size is outside the applicable range.

Operation Overview

Two text strings (Src1 (Input Text String 1) and Src2 (Input Text String 2)) are compared.

Result (Comparison Result) = 1 if the two text strings match. Result (Comparison Result) = 0 if the two text strings do not match. How many bytes to compare from the beginning of the text strings can be determined by Size (Comparison Size). If Size is specified as 0, the length of Src1 is compared.

Examples of the instruction are shown below.

- Src1 = "12345", Src2 = "12367", Size = 0 → Result = 0
- Src1 = "abc123", Src2 = "abc234", Size = 3 → Result = 1
- Src1 = "abc123", Src2 = "abc4567", Size = 10 → Result = 0

If the same text string data is accessed by different tasks at the same time, the data may be corrupted. Create the program so that the data is not accessed by different tasks at the same time.

Important

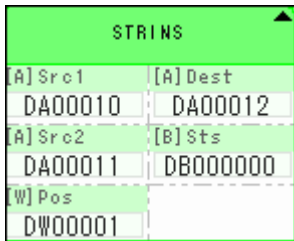
Text strings handled by this instruction are 1,999 bytes maximum (plus the NULL character).

4.12.11 Insert String (STRINS)

A text string is inserted at the specified position inside another string.

Format

The format of this instruction is shown below.



I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src1	×	×	×	×	×	×	○*1	×	×
Src2	×	×	×	×	×	×	○*1	×	×
Pos	×	○	×	×	×	×	×	×	○
Dest	×	×	×	×	×	×	○*2	×	×
Sts*3	○*2	×	×	×	×	×	×	×	×

*1. M, G, D, or C register only.

*2. M, G, or D register only.

*3. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Src1	Base Text String	IN	The register that stores the base text string into which the other text string will be inserted.
Src2	Text String to Insert	IN	The first register of the text string to insert.
Pos	Insertion Position	IN	Specify the byte position in the base text string to insert the text string at (0 to 1,999).
Dest	Text String after Insertion	OUT	Specify the register to store the output text string. Output processing is not performed when Sts (Status) is ON.
Sts	Status	OUT	Status is turned OFF when processing was performed normally. Status is turned ON when the following errors occur. <ul style="list-style-type: none"> • The number of characters exceeds the maximum value. • The text string to output exceeds the maximum range of registers. • Size is outside the applicable range. • Pos > number of bytes in Src1.

Operation Overview

Src2 (Text String to Insert) is inserted into Src1 (Base Text String) at the desired position specified by Pos (Insertion Position). The text string after insertion is stored in Dest (Text String after Insertion).

Examples of the instruction are shown below.

- Src1 = "12345", Src2 = "abc", Pos = 0 → Dest = "abc12345"
- Src1 = "12345", Src2 = "abc", Pos = 3 → Dest = "123abc45"
- Src1 = "12345", Src2 = "abc", Pos = 5 → Dest = "12345abc"

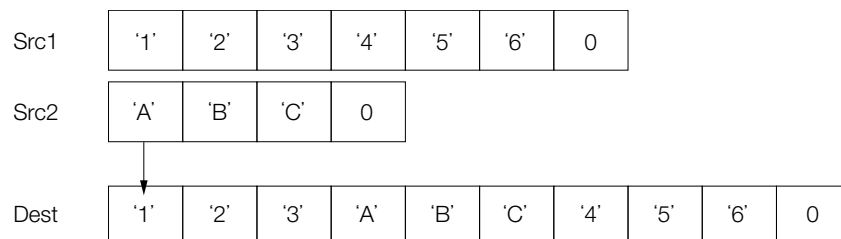


Important

If the same text string data is accessed by different tasks at the same time, the data may be corrupted. Create the program so that the data is not accessed by different tasks at the same time.

Text strings handled by this instruction are 1,999 bytes maximum (plus the NULL character).

Example When Src1 = "123456", Src2 = "ABC", Pos = 3:



4.12.12 Find String (STRFIND)

The specified text string is found inside another string.

Format

The format of this instruction is shown below.

STRFIND	
[A] Src1	[W] Result
DA00010	DW00002
[A] Src2	[B] Sts
DA00011	DB000000
[W] Pos	
DW00001	

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src1	×	×	×	×	×	×	○ ^{*1}	×	×
Src2	×	×	×	×	×	×	○ ^{*1}	×	×
Pos	×	○	×	×	×	×	×	×	○
Result	×	○ ^{*2}	×	×	×	×	×	×	×
Sts ^{*3}	○ ^{*2}	×	×	×	×	×	×	×	×

*1. M, G, D, or C register only.

*2. M, G, or D register only.

*3. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Src1	Target Text String	IN	First register in which the text string to be searched is stored.
Src2	Text String to Find	IN	First register in which the text string to find is stored.
Pos	Search Start Position	IN	Specify the byte position at which the search starts in the text string to be searched (0 to length of the input text string). When 0 is specified, the text string is searched from the first byte.
Result	Search Result	OUT	This item outputs the byte position from the search start position at which the text string was found. 0 is output when the text string is not found or Sts (Status) is ON.
Sts	Status	OUT	Status is turned OFF when processing was performed normally. Status is turned ON when the following errors occur. <ul style="list-style-type: none"> The number of characters exceeds the maximum value. Pos is outside the applicable range. Pos > number of bytes in Src1.

Operation Overview

The search range is from the text string in Src1 (Target Text String) and within the text string in Src2 (Text String to Find). Specify the position with Pos (Search Start Position). If the text string in Src2 was found, the number of bytes from the search start position is stored in Result (Search Result).

Examples of the instruction are shown below.

- Src1 = "12345", Src2 = "34", Pos = 0 (= 1) → Result = 3
- Src1 = "12345", Src2 = "34", Pos = 2 → Result = 1



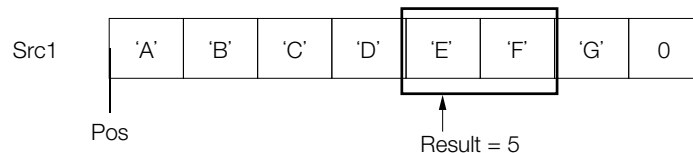
Important

1. If the same text string data is accessed by different tasks at the same time, the data may be corrupted. Create the program so that the data is not accessed by different tasks at the same time.
2. If the target text string is long and if the target text string is at the end of the text string to be searched, the processing time for this instruction may increase and exceed the scan set value.

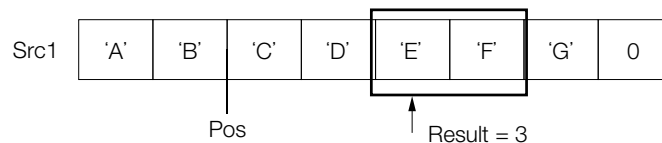
Text strings handled by this instruction are 1,999 bytes maximum (plus the NULL character).

Example When Src1 = "ABCDEFG", Src2 = "EF":

- Pos = 0



- Pos = 2



4.12.13 Extract String (STREXTR)

A text string with the specified start position and size is extracted from another string.

Format

The format of this instruction is shown below.

STREXTR	
[A] Src	[A] Dest
DA00010	DA00011
[W] Pos	[B] Sts
DW00001	DB000000
[W] Size	
DW00002	

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src	x	x	x	x	x	x	O*1	x	x
Pos	x	O	x	x	x	x	x	x	O
Size	x	O	x	x	x	x	x	x	O
Dest	x	x	x	x	x	x	O*2	x	x
Sts*3	O*2	x	x	x	x	x	x	x	x

*1. M, G, D, or C register only.

*2. M, G, or D register only.

*3. Optional.

Details on I/O Items


I/O Item	Name	I/O	Description
Src	Input Text String	IN	Specify the first register in which the input text string is stored.
Pos	Start Position	IN	Specify the byte position from which to start extracting the text string. When 0, the text string is extracted from the first byte.
Size	Size	IN	Specify the number of bytes to extract (0 to 1,999).
Dest	Output Text String	OUT	Specify the register to store the output text string. Output processing is not performed when Sts (Status) is ON.
Sts	Status	OUT	Status is turned OFF when processing was performed normally. Status is turned ON when the following errors occur. <ul style="list-style-type: none"> The number of characters exceeds the maximum value. The text string to output exceeds the maximum range of registers. Pos or Size is outside the applicable range. Pos + Size > number of bytes in Src.

Operation Overview

A text string of the specified Size is extracted from the number of bytes in Pos (Start Position) in the text string in Src (Input Text String). The text string that was extracted is stored in Dest (Output Text String).

Examples of the instruction are shown below.

- Src = "12345", Pos = 0 (=1), Size = 2 → Dest = "12"
- Src = "12345678", Pos = 3, Size = 3 → Dest = "345"

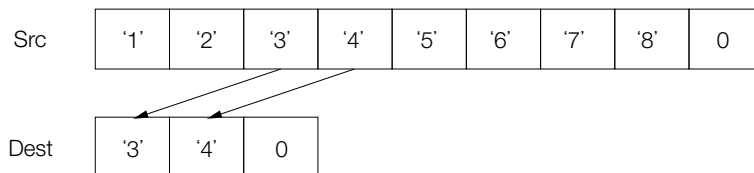


If the same text string data is accessed by different tasks at the same time, the data may be corrupted. Create the program so that the data is not accessed by different tasks at the same time.

Important

Text strings handled by this instruction are 1,999 bytes maximum (plus the NULL character).

Example When Src = "12345678", Pos = 3, and Size = 2:



4.12.14 Extract String from End (STREXTRE)

A text string of the specified size is extracted from the end of another string.

Format

The format of this instruction is shown below.

STREXTRE	
[A] Src	[A] Dest
DA00010	DA00011
[W] Size	[B] Sts
DW00001	DB000000

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src	x	x	x	x	x	x	○*1	x	x
Size	x	○	x	x	x	x	x	x	○
Dest	x	x	x	x	x	x	○*2	x	x
Sts *3	○*2	x	x	x	x	x	x	x	x

*1. M, G, D, or C register only.

*2. M, G, or D register only.

*3. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Src	Input Text String	IN	Specify the first register in which the input text string is stored.
Size	Size	IN	Specify the number of bytes to extract (0 to 1,999).
Dest	Output Text String	OUT	Specify the register to store the output text string. Output processing is not performed when Sts (Status) is ON.
Sts	Status	OUT	Status is turned OFF when processing was performed normally. Status is turned ON when the following errors occur. <ul style="list-style-type: none"> The number of characters exceeds the maximum value. The text string to output exceeds the maximum range of registers. Pos > number of bytes in Src. Pos is outside the applicable range.

Operation Overview

A text string of the specified Size (Size) is extracted from the end of the text string in Src (Input Text String). The text string that was extracted is stored in Dest (Output Text String).

Src = "12345", Size = 2 → Dest = "45"

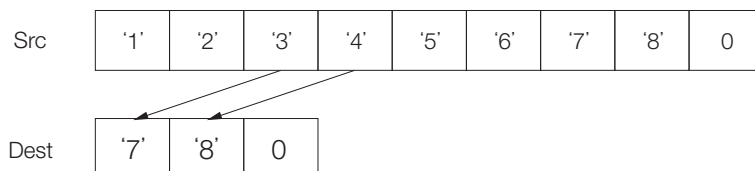


Important

If the same text string data is accessed by different tasks at the same time, the data may be corrupted. Create the program so that the data is not accessed by different tasks at the same time.

Text strings handled by this instruction are 1,999 bytes maximum (plus the NULL character).

Example When Src = "12345678", Size = 2:



4.12.15 Delete Spaces at String Ends (STRTRIM)

Leading and trailing spaces and tabs are deleted from the text string.

Format

The format of this instruction is shown below.

STRTRIM			
[A] Src	[A] Dest		
DA00010	DA00011		
[W] Option	[B] Sts		
DW00001	DB000000		

I/O Item	Applicable Data Types								
	B	W	L	Q	F	D	A	Index	Constant
Src	×	×	×	×	×	×	○ ^{*1}	×	×
Option	×	○	×	×	×	×	×	×	○
Dest	×	×	×	×	×	×	○ ^{*2}	×	×
Sts ^{*3}	○ ^{*2}	×	×	×	×	×	×	×	×

*1. M, G, D, or C register only.

*2. M, G, or D register only.

*3. Optional.

Details on I/O Items

I/O Item	Name	I/O	Description
Src	Input Text String	IN	Specify the first register in which the input text string is stored.
Option	Option Settings	IN	Bit 0: Delete leading tabs and spaces. Bit 1: Delete trailing tabs and spaces. Bit 2 to F: Reserved for system.
Dest	Output Text String	OUT	Specify the register to store the output text string. Output processing is not performed when Sts (Status) is ON.
Sts	Status	OUT	Status is turned OFF when processing was performed normally. Status is turned ON when the following errors occur. <ul style="list-style-type: none"> The number of characters exceeds the maximum value. The text string to output exceeds the maximum range of registers.

Operation Overview

Tabs and spaces at the location specified by Option (Option Settings) are deleted from the text string in Src (Input Text String). The text string after tabs and spaces are deleted is stored in Dest (Output Text String).

Examples of the instruction are shown below.

- Src = " 12345 ", Option = 0x0001 (deleting leading whitespace) → Dest = "12345 "
- Src = " 12345 ", Option = 0x0002 (deleting trailing whitespace) → Dest = " 12345"
- Src = " 12345 ", Option = 0x0003 (deleting leading and trailing whitespace) → Dest = "12345"
- Src = " 12345 ", Option = 0x0000 → Dest = " 12345 "



If the same text string data is accessed by different tasks at the same time, the data may be corrupted. Create the program so that the data is not accessed by different tasks at the same time.

Text strings handled by this instruction are 1,999 bytes maximum (plus the NULL character).

Features of the MPE720 Engineering Tool

5

This chapter describes the key features of the MPE720 Engineering Tool for ladder programming.

- 5.1 Ladder Program Runtime Monitoring 5-4**
- 5.2 Search/Replace 5-5**
 - 5.2.1 Searching and Replacing in Programs 5-5
 - 5.2.2 Searching and Replacing in Project Files 5-7
- 5.3 Cross References 5-10**
- 5.4 Checking for Multiple Coils 5-13**
- 5.5 Forcing Coils ON and OFF 5-14**
 - 5.5.1 Forcing Coils ON or OFF from a Ladder Program 5-14
 - 5.5.2 Changing the Forced ON/OFF Status from the Force Coil List Pane 5-14
- 5.6 Viewing Called Programs 5-17**
- 5.7 Register Lists 5-18**
 - 5.7.1 Displaying the Register Map 5-18
 - 5.7.2 Switching the Register Map Display 5-19
 - 5.7.3 Editing Data 5-20
- 5.8 Tuning Panel 5-21**
- 5.9 Enabling and Disabling Ladder Programs . . 5-22**

5.10	Watching	5-23
	5.10.1 Displaying Watch Data	5-23
	5.10.2 Editing the Value Column	5-23
5.11	Security	5-24
5.12	Tracing	5-25
5.13	Advanced Programming	5-26
	5.13.1 Motion Programs	5-26

This chapter describes the following ladder programming and debugging functions of MPE720 Engineering Tool version 7.

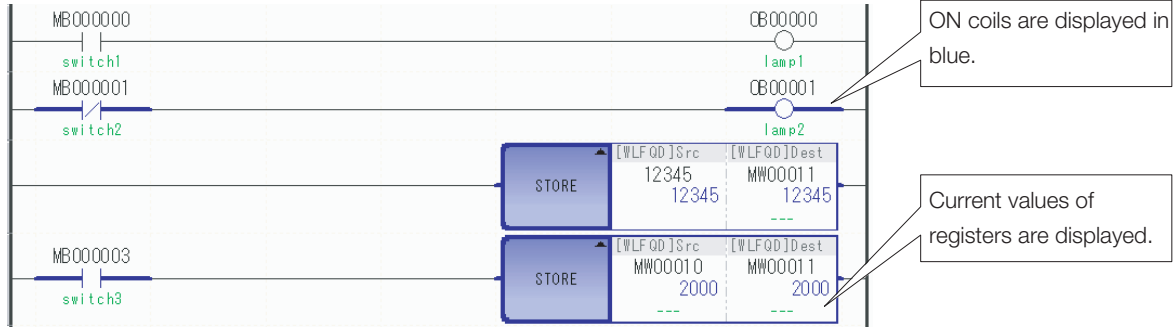
- Ladder program runtime monitoring
- Search/replace
- Cross references
- Multiple coils
- Forcing coils ON and OFF
- Viewing called programs
- Register lists
- Tuning panel
- Enabling and disabling ladder programs
- Watching
- Security
- Tracing
- Using motion programs

5.1 Ladder Program Runtime Monitoring

You can monitor the execution status of each instruction. Using runtime monitoring requires a connection to the Machine Controller.

Instructions where the relay output is ON are displayed in blue.

The current values of the parameter registers of the instructions that are being executed are also displayed.



5.2 Search/Replace

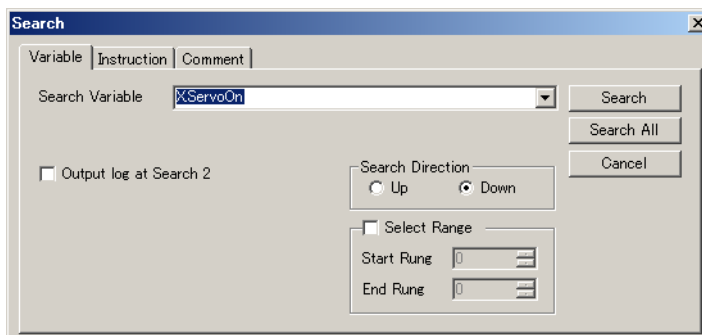
5.2.1 Searching and Replacing in Programs

You can search for variables, instructions, and comments in a specified program. You can also search for and replace registers and register comments.

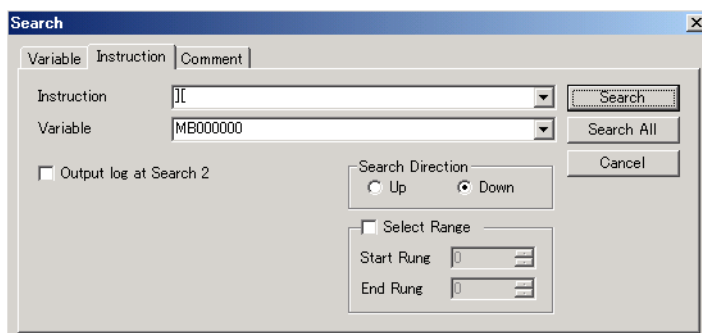
The following section describes how to search for and replace text in programs.

Searching in Programs

1. Bring the program to search to the front in the Ladder Editor, and then select **Edit – Find** from the menu bar.
The Search Dialog Box will be displayed.
2. Click the **Variable, Instruction or Comment** Tab to set the search criteria.

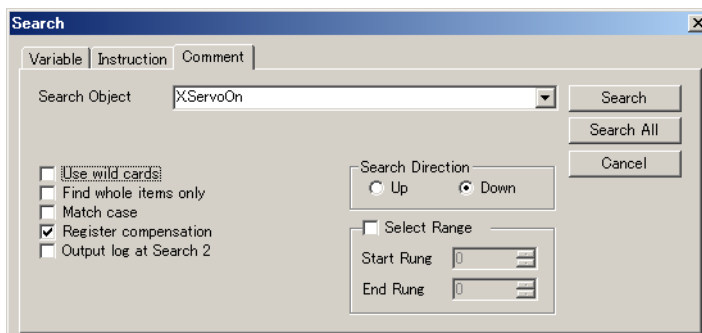


Variable Tab Page: Allows you to search for variables and registers. You can also enter the variable by copying it from the Variable Pane.



Instruction Tab Page: Enter the name of the instruction or the assigned instruction key in the **Instruction** Box.

The **Variable** Box is displayed when an instruction is entered in the **Instruction** Box. If the SEE instruction is entered in the **Instruction** Box, **Variable** changes to **Program Name**. You can also enter the variable by copying it from the Variable Pane.



Comment Tab Page: Allows you to search for object comments, rung comments, program comments, and expression comments.

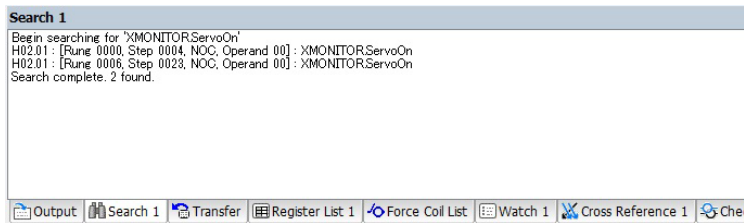
5.2.1 Searching and Replacing in Programs

- **Use wild cards** Check Box: Select this check box to use wildcard characters (* and ?) in the search string.
- **Find whole items only** Check Box: Select this check box to search for comments where the string in the comment box is exactly the same as the search string. Case sensitivity is controlled by the **Match case** Check Box.
- **Match case** Check Box: Select this check box to differentiate between uppercase and lowercase characters.
- **Register compensation** Check Box: Select this check box to convert search strings that are recognized as registers into register notation.
- **Output log at Search 2** Check Box: Select this check box to display the search results in the Search 2 Pane without changing the contents of the Search 1 Pane. If you clear the selection of the check box, the search results will be displayed in the Search 1 Pane.
- **Select Range** Check Box: If you select this check box, you can specify the search range by setting the start and end rungs.

3. Click the **Search Button** or the **Search All Button** to start searching.

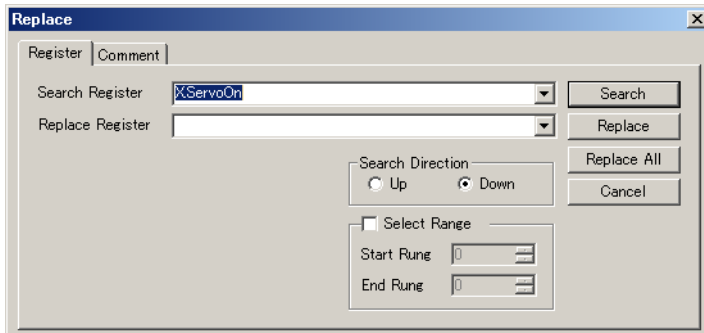
If you click the **Search** Button, the instruction object that was found will be selected.

If you click the **Search All** Button, the search results will be displayed in the Search 1 or Search 2 Panes.

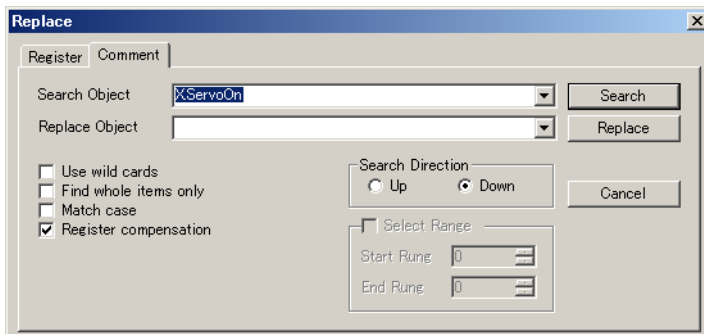


Replacing Text in Programs

1. Bring the program in which to search and replace to the front of the Ladder Editor, and then select **Edit – Replace** from the menu bar. The Replace Dialog Box will be displayed.
2. Click the **Register** or **Comment** Tab to set the search criteria and the replacement string.



Register Tab Page: Allows you to search for and replace registers.



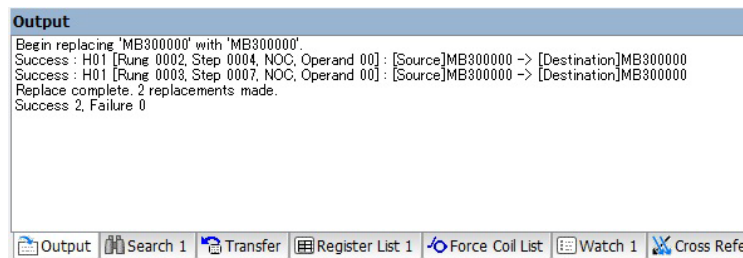
Comment Tab Page: Allows you to search for object comments, rung comments, program comments, and expression comments.

- **Use wild cards** Check Box: Select this check box to use wildcard characters (* and ?) in the search string.
Note: If you enter an * or a ? character in the **Replace Register** or **Replace Object** Box, they will not be handled as wildcards, but as regular characters.
- **Select Range** Check Box: If you select this check box, you can specify the search range by setting the start and end rungs.
However, range selection is disabled on the Comment Tab Page.

3. Start the search/replace operation.

Click the **Search** Button. The instruction object that was found will be selected. If you click the **Replace** Button, the object will be replaced by the contents of the **Replace Register** or **Replace Object** Box.

If you click the **Replace All** Button on the Register Tab Page, the registers that are found will be replaced, and the replacement results will be displayed in the Output Pane.



5.2.2 Searching and Replacing in Project Files

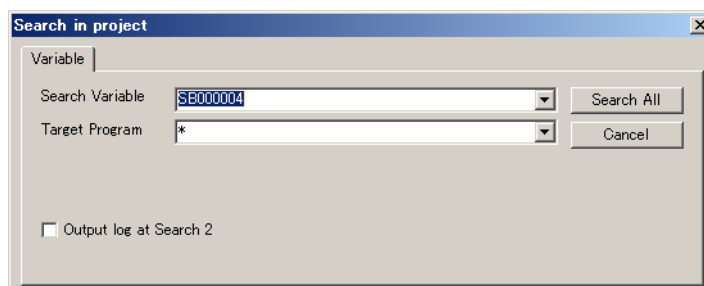
You can search for variables in all ladder programs and motion programs, or in only the specified programs in a project file. You can also search for and replace registers and addresses.

Information You can search the project file only when the Machine Controller is offline.

The following section describes how to search for and replace text in a project file.

Searching in Project Files

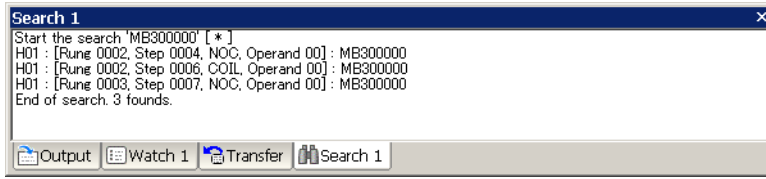
1. Bring the program to search to the front of the Ladder Editor, and then select **Edit – Search in Project** from the menu bar.
The Search in Project Dialog Box will be displayed.
2. Specify the address of the variable to search for and the name of the program to search.



- Note:
1. You can also enter the variable by copying it from the Variable Pane.
 2. Use commas and spaces to specify more than one program in the **Target Program** Box. The following wildcard (*) combinations can also be used in the **Target Program** Box:
, H, L*, I*, A*, F* (all functions), MPM*, and MPS*
Wildcards may be used only in the formats given above. Other uses, such as "H01.*", are not allowed.
 3. **Output log at Search 2** Check Box: Select this check box to display the search results in the Search 2 Pane without changing the contents of the Search 1 Pane. If you clear the selection of the check box, the search results will be displayed in the Search 1 Pane.

3. Start the search operation.

Click the **Search All** Button. A progress bar will be displayed, and the search results will appear in the Search Pane.



Replacing in Project Files

- Information** • After you perform a replace operation on a project file, the project file will be compiled and saved, and there will be no way to return to the previous version. Always create a backup before performing replacements on important files.
- If a motion program is already open in the MPE720 Engineering Builder before the replacement is executed, the program will not be automatically updated. Close the motion program before executing the replacement operation.

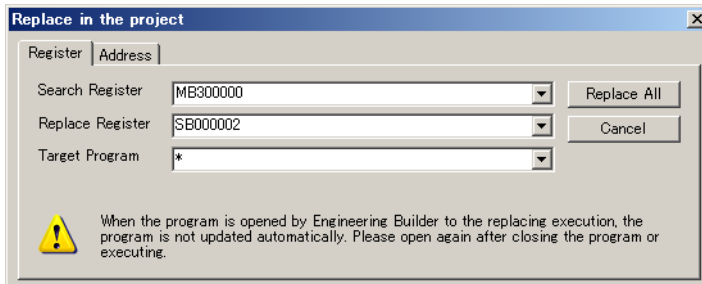
1. Bring the program to search to the front of the Ladder Editor, and then select **Edit – Replace in Project** from the menu bar.

The Replace in the Project Dialog Box will be displayed.

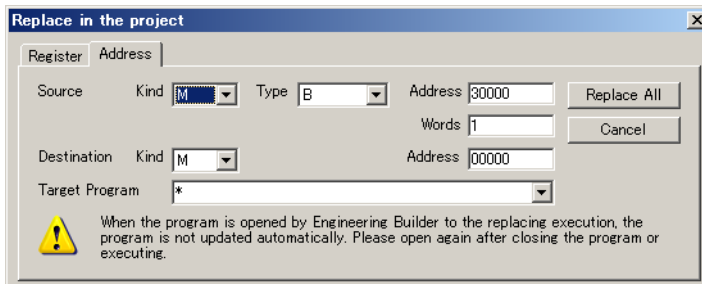
2. Specify the address of the variable to search for and the name of the program to search.

Note: 1. You can also enter the variable by copying it from the Variable Pane.
 2. Use commas and spaces to specify more than one program in the **Target Program** Box. The following wildcard (*) combinations can also be used in the **Target Program** Box:
 , H, L*, I*, A*, F* (all functions), MPM*, and MPS*
 Wildcards may be used only in the formats given above. Other uses, such as "H01.*", are not allowed.

3. Click the **Register** or **Address** Tab to set the search criteria and the replacement value.



Register Tab Page: Allows you to replace registers.



Address Tab Page: Allows you to replace registers that meet the specified criteria.

Note: The following wildcard (*) combinations can also be used in the **Target Program** Box:
 , H, L*, I*, A*, F*, MPM*, MPS*

4. Start the search/replace operation.

Click the **Replace All** Button. The replacement results will be displayed in the Output Pane.

```

Output
Start the replace 'Kind: M Type: B Address: 30000 No. of words: 1' is replace with 'Kind: M Address: 00000'. [*]
Success: H06.02 [Rung 0007, Step 0024, COIL, Operand 00] : [Source]MB300008 -> [Destination]MB000008
Success: H06.02 [Rung 0014, Step 0036, NOC, Operand 00] : [Source]MB300008 -> [Destination]MB000008
Success: H06.02 [Rung 0024, Step 0050, NOC, Operand 00] : [Source]MB300008 -> [Destination]MB000008
Success: H06.02 [Rung 0025, Step 0052, NOC, Operand 00] : [Source]MB300008 -> [Destination]MB000008
Success: H06.02 [Rung 0028, Step 0056, NOC, Operand 00] : [Source]MB300008 -> [Destination]MB000008
----- Start compiling : H06.02 : phase control 2 (electronic cam) -----
Error 0 : Warning 0
Success: H01 [Rung 0002, Step 0004, NOC, Operand 00] : [Source]MB300000 -> [Destination]MB000000
Success: H01 [Rung 0002, Step 0006, COIL, Operand 00] : [Source]MB300000 -> [Destination]MB000000
Success: H01 [Rung 0003, Step 0007, NOC, Operand 00] : [Source]MB300000 -> [Destination]MB000000
Success: H01 [Rung 0004, Step 0010, NOC, Operand 00] : [Source]MB300001 -> [Destination]MB000001
Success: H01 [Rung 0005, Step 0012, NOC, Operand 00] : [Source]MB300001 -> [Destination]MB000001
----- Start compiling : H01 : common settings for axes -----
Error 0 : Warning 0
Success: H04 [Rung 0003, Step 0009, NOC, Operand 00] : [Source]MB300001 -> [Destination]MB000001
----- Start compiling : H04 : main program for positioning -----
Error 0 : Warning 0
Success: L06 [Rung 0000, Step 0000, NOC, Operand 00] : [Source]MB300008 -> [Destination]MB000008
----- Start compiling : L06 : electric cam table data generation -----
Error 0 : Warning 0
End of replace. 12 founds.
Success 12, Failure 0

```

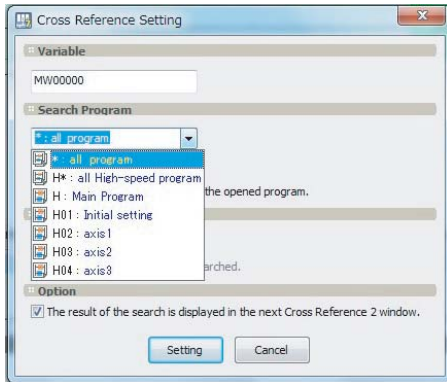
Note: If an error occurs during compilation of a program, the replacements will not be completed.

After the replacement operation, the variables and addresses of the registers that were replaced will be displayed.

5.3 Cross References

Cross referencing allows you to check whether a register is used in a program, and where it is used.

The search results indicate output registers in red, input registers in blue.



Cross referencing executed.

Search Results Display
Red: Output registers
Blue: Input registers

Cross Reference 1 [MW00000-* : All program / Search Result 6]						
Variable	MW00000		Search	Setting...		
Register	Program	Execution Instr...	Execution Step	Write/Read	Comm	
Same Register						
MW00000	H02 : axis1	LOAD : Integer ...	0	Read		
MW00000	H02 : axis1	STORE : Store	1	Write		
Same Memory Address						
MB000004	H02 : axis1	NOC : NO Contact	2	Read		
MB000005	H02 : axis1	COIL : Coil	3	Write		
ML00000	H03 : axis2	LOAD : Integer ...	0	Read		
ML00000	H03 : axis2	STORE : Store	1	Write		

If the value of a register is different from its set value, it means that the value of the register may have been overwritten somewhere in the program. In this case, you can search for the registers using cross references. Check the registers displayed in red, and locate the program that is overwriting them.

Example

The following section describes the search operation on arrays.

1. Register[Register] Arrays



MW00000 and MW00001 are subject to searching.

2. Register[Constant] Arrays



MW00000 and MW00005 are subject to searching.

3. Register[Constant], LONG Arrays





ML00000 and ML00010 are subject to searching.

The following cross-reference criteria can be set. The following tables describe the check boxes.



The local register is searched in the opened program.

Check Box	Search Method
Selected.	A search is made for local registers (D registers) in the active drawing in the MPE720 Window.
Not selected.	A search is made for local registers (D registers) in the specified drawing.

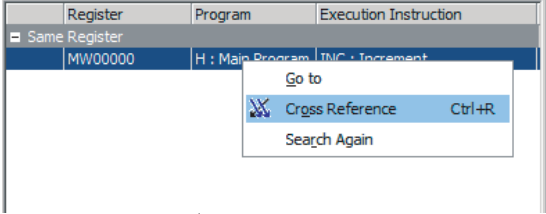
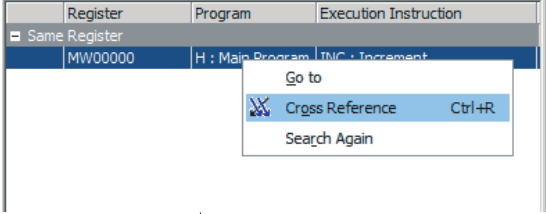
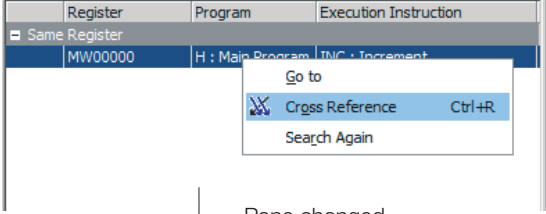
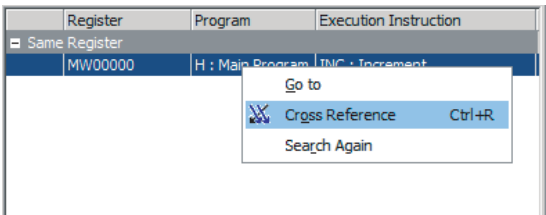
The same register is searched.

Check Box	Search Method
Selected.	<p>A search is made for registers that are the same as the register that was found. Select this check box to display the results in a list when you search the following instruction for a variable of MW00000.</p> 
Not selected.	<p>A search is not made for registers with the same data type as that of the register that was found. Clear the selection of this check box to not display the results in a list when you search the following instruction for a variable of MW00000.</p> 

The same memory address is searched.

Check Box	Search Method
Selected.	<p>Searches for redundant addresses. Select this check box to display the results in a list when you search the following instruction for a variable with a different data type, such as ML00000.</p> 
Not selected.	<p>A search is not performed for redundant addresses. Clear the selection of this check box to not display the results in a list when you search the following instruction for a variable with a different data type, such as ML00000.</p> 

The result of the search is displayed in the next Cross Reference 2 window.

Check Box	Search Method
<p>Selected.</p>	<p>When you perform cross referencing from the Cross Reference Pane, the results will be displayed in a separate pane. Cross reference results can be displayed in up to 3 panes.</p> <p>Cross Reference 1 Pane</p>  <p style="text-align: center;">↓ Pane changed.</p> <p>Cross Reference 2 Pane</p>  <p style="text-align: center;">↓ Pane changed.</p> <p>Cross Reference 3 Pane</p>  <p style="text-align: center;">↓ Pane changed.</p>
<p>Not selected.</p>	<p>When you perform cross referencing from a Cross Reference Pane, the results will be displayed by updating the data in the same pane.</p> <p>Cross Reference 1 Pane</p>  <p style="text-align: center;">Page updated.</p>

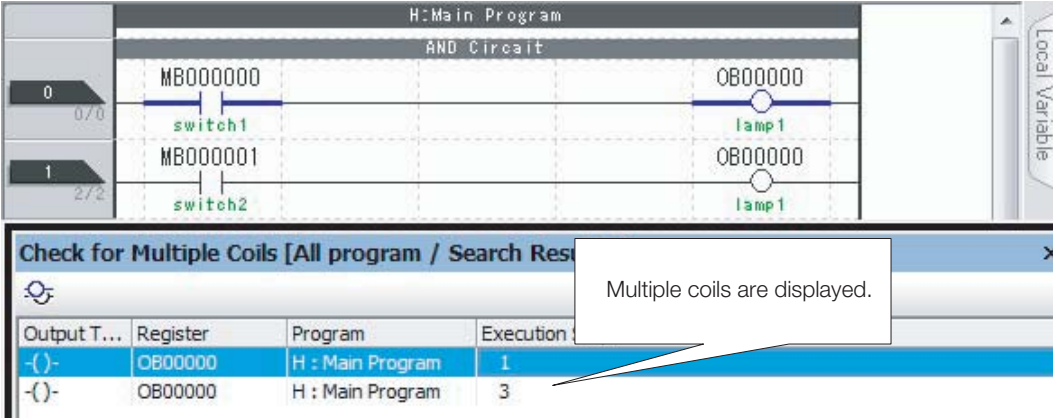
5.4 Checking for Multiple Coils

You can check for multiple coils (different coils that use the same register) in an entire ladder program, and display the search results.

Information When you use a project link connection, the data in the project file is used. Sometimes the displayed results do not match the data in the linked Machine Controller. When you check for multiple coils and use a project link connection, first always read the data to the project file from the Machine Controller.

Select **Debug – Check for Multiple Coils** from the menu bar.

Searching for multiple coils will start, and the results will be displayed in the Check for Multiple Coils Pane.



The screenshot displays a ladder logic program titled 'H: Main Program' with an 'AND Circuit'. It contains two rungs: Rung 0 (0/0) with 'switch1' and Rung 1 (2/2) with 'switch2'. Both rungs are connected to 'Lamp1' coils. Below the ladder logic is the 'Check for Multiple Coils [All program / Search Results]' pane. It contains a table with the following data:

Output T...	Register	Program	Execution
-(-)	OB00000	H : Main Program	1
-(-)	OB00000	H : Main Program	3

A callout box points to the results with the text: "Multiple coils are displayed."

Information If the **Enable to Multiple Coil Check** Check Box is selected in the compile options, a search for multiple coils will be performed during compilation and the results will be displayed as warnings in the Output Pane.

5.5 Forcing Coils ON and OFF

You can force a specified coil ON or OFF from the Ladder Editor.

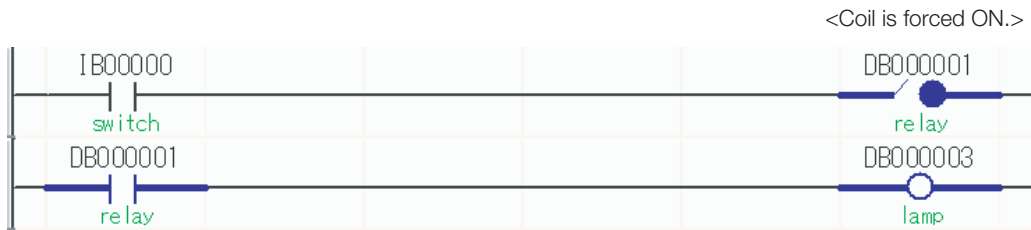
The coil will output ON or OFF regardless of the output of the instruction to the left of the coil.

In the following programming example, you can simulate turning ON the switch (IB00000) by forcing the DB000001 relay ON even though the physical switch does not exist.

5.5.1 Forcing Coils ON or OFF from a Ladder Program

You can monitor a program by forcing specified coil objects ON or OFF in the Ladder Editor.

1. Select the coil to force ON or OFF.
2. Select **Debug – Force ON** or **Force OFF** from the menu bar.
The selected coil will be forced ON or OFF.




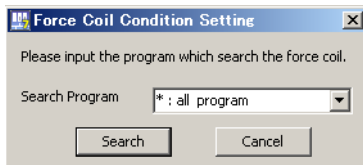
Information Select **Debug – Disable Force** from the menu bar to cancel forced ON or forced OFF status.

5.5.2 Changing the Forced ON/OFF Status from the Force Coil List Pane

The Force Coil List Pane lists the ON/OFF status of the forced coils in the ladder program. You can also change and cancel the ON, OFF, or canceled status of the forced coils in the entire ladder program.

Searching for Forced Coils in the Force Coil List Pane

1. Display the Force Coil List Pane.
Note: You can show and hide the Force Coil List Pane by selecting **View – Other Windows – Force Coil List** from the menu bar.
2. Select **Debug – Force Coil List** from the menu bar.
Note: In the above case, all programs will be searched for forced coils. To specify a program for the search, press the Forced Coil Condition Setting Button () to display the Forced Coil Condition Setting Dialog Box.



The search results will be displayed in the Force Coil List Pane.

Forcing State	Coil	Program	Variable	Comment	Execution Step
ON	-(S ON)-	H01 : common settings for axes	MB000200		40
OFF	-(R OFF)-	H01 : common settings for axes	MB000200		42
ON	-(ON)-	H : High-speed Main Program	MB000100		6
ON	-(ON)-	H : High-speed Main Program	DB000001	relay	8
ON	-(ON)-	H : High-speed Main Program	MB000100		12
OFF	-(OFF)-	H : High-speed Main Program	MB000100		14

3. Select the check boxes for the coils to force ON or OFF.

Forcing State	Coil	Program	Variable	Comment	Execution Step
ON	-(S ON)-	H01 : common settings for axes	MB000200		40
OFF	-(R OFF)-	H01 : common settings for axes	MB000200		42
ON	-(ON)-	H : High-speed Main Program	MB000100		6
ON	-(ON)-	H : High-speed Main Program	DB000001	relay	8
ON	-(ON)-	H : High-speed Main Program	MB000100		12
OFF	-(OFF)-	H : High-speed Main Program	MB000100		14

Information


1. If you right-click in the list in the Force Coil List Pane, you can use the pop-up menu to select **Check All** or **Uncheck All** to select or clear the selections of the all of the **Forcing State** Check Boxes.
2. If you select or double-click a search result row in the Force Coil List Pane, you can jump to the corresponding coil in the ladder program. Alternatively, you can right-click in the list in the Force Coil List Pane, and select **Go to** from the pop-up menu. If the program is not open, it will be opened automatically and the display will jump to the corresponding coil in the program.
3. If you right-click in the list in the Force Coil List Pane and select **Cross Reference** from the pop-up menu, or select **Debug – Cross Reference** from the menu bar, the register that is set for the coil will be checked for cross references and the results will be displayed in the Cross Reference Pane.
4. If you edit the ladder program while the search results are displayed, the coils in the edited program will be displayed in gray.

Names and Descriptions of the Force Coil List Pane Items

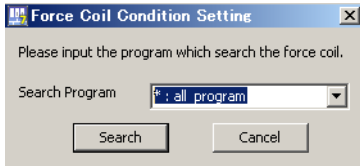
The Force Coil List Pane consists of a list where the forced coils are displayed, and a toolbar that is used to search and repeat searches for forced coils, and to change the forced status of coils.


Forcing State ①	Coil ②	Program ③	Variable ④	Comment ⑤	Execution Step ⑥
ON	-(S ON)-	H01 : common settings for axes	MB000200		40
OFF	-(R OFF)-	H01 : common settings for axes	MB000200		42
ON	-(ON)-	H : High-speed Main Program	MB000100		6
ON	-(ON)-	H : High-speed Main Program	DB000001	relay	8
ON	-(ON)-	H : High-speed Main Program	MB000100		12
OFF	-(OFF)-	H : High-speed Main Program	MB000100		14

◆ **Toolbar**

- **Forced Coil Condition Setting Button** ()

Click this button to display the Forced Coil Condition Setting Dialog Box. Specify the program to search for forced coils.



- **Search Again Button** ()

Click this button to repeat the forced coil search in the program that was specified in the Force Coil Condition Setting Dialog Box.

- **Force Reset Button** ()


Click this button to cancel the forced status of the selected coils.

- **Force ON Button** ()

Click this button to force ON the selected coils.

- **Force OFF Button** ()

Click this button to force OFF the selected coils.

- **Display Variable Button** ()

Click this button to switch the display of the register that is used by the coil between a register or a variable.

◆ **List**

① **Forcing State**

This column displays the forced ON or OFF status of the coils that were found.

② **Coil**

This column displays the coils that were found.
There are six types of coils.

Coil Type	Coil Symbol	
	ON	OFF
Coil	-/ (ON)-	-/ (OFF)-
Set Coil	-/ (S ON)-	-/ (S OFF)-
Reset Coil	-/ (R ON)-	-/ (R OFF)-

③ **Program**

This column displays the names of the programs where the coils were found.

④ **Variable**

This column displays the variables or registers that are set for the coils that were found.

⑤ **Comment**

This column displays the comments of the variables.

⑥ **Execution Step**

This column displays the execution step numbers of the coils that were found.

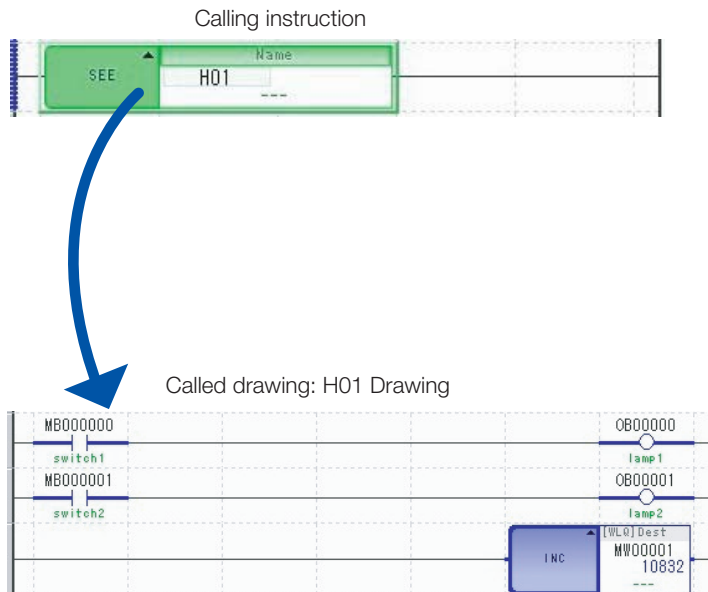
⑦ **Check Boxes**

The coils with selected check boxes will be subject to forcing operations (ON, OFF, or Cancel). You can use the toolbar buttons and also the pop-up menu to force the status of all selected coils to ON, OFF, or canceled.

5.6 Viewing Called Programs

You can open a drawing that is called with an SEE instruction or a FUNC instruction.

Select the SEE instruction object or FUNC instruction object for the program to view, and select **Debug – Open Program** from the menu bar.



5.7 Register Lists

You can monitor the current values of the registers in a continuous area (register map) on any of the Register List 1, 2, and 3 Panes. Realtime monitoring is possible if the Machine Controller is connected. You can edit the values.

- Information**
 - The register map will show the data in the project file even for a direct connection. If you use a project link connection, the data in the Machine Controller is accessed. When the register map is displayed, the displayed results do not always match the project file of the linked project.
 - If you display the register map when using a project link connection, first always transfer the data to the project file by reading the data from the Machine Controller.
 - The register list can display S, I, O, M, C, D, and G registers. However, C registers are read-only. They can be read but not written.

5.7.1 Displaying the Register Map

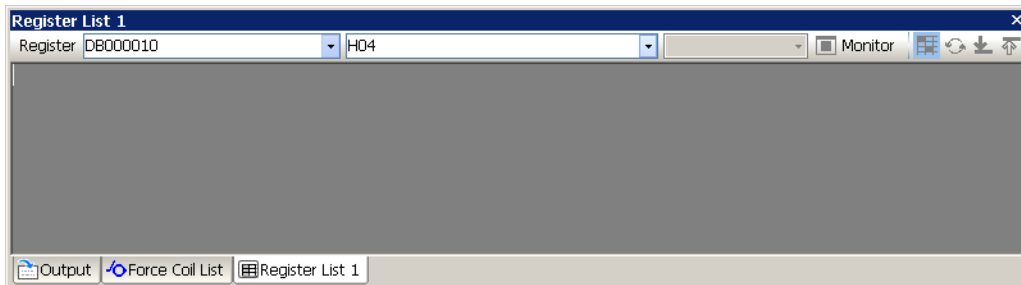
The following table gives the meaning of the background colors in the register map.

Green	Indicates a register that is used in a ladder program.
Red	Indicates a redundant register (i.e., a register that is used for more than one data type).

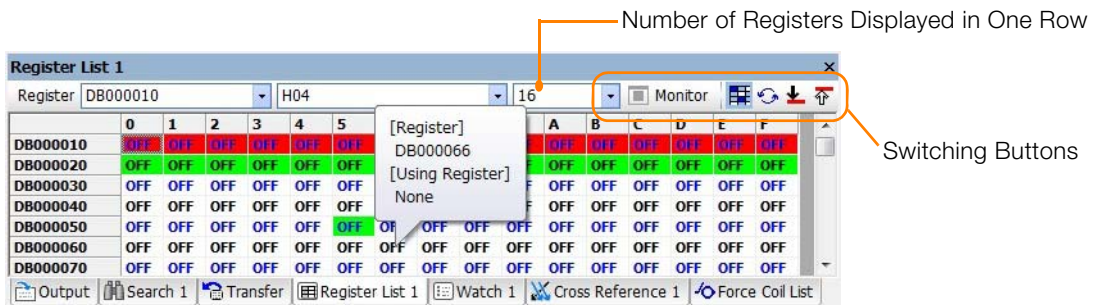
Use the following procedure to display the register map.

- Click one of the tabs for the Register List 1, 2 or 3 Panes. Select **Monitor – Register List** from the Launcher. The Register List 1 Pane will be displayed.

Note: You can show or hide the Register List 1, 2, and 3 Panes by selecting **View – Register List – Register List 1**, **View – Register List – Register List 2**, or **View – Register List – Register List 3** from the menu bar.
- Enter the address of the register for which to display a register map in the **Register Box**. When displaying a list of D registers, enter the program number as shown below.



- Press the **Enter Key**. The specified register will be displayed in the top row of the register map.



Example of Displaying the D Register Map and Balloon

Register	0	1	2	3	4	5	6	7
MW00014	0	0	0	0	0	0	0	0
MW00022	0	0	0	0	0	0	0	0
MW00030	0	0	0	0	0	0	0	0
MW00038	0	0	0	0	0	0	0	0
MW00046	0	0	0	0	0	0	0	0
MW00054	0	0	0	0	0	0	0	0
MW00062	0	0	0	0	0	0	0	0

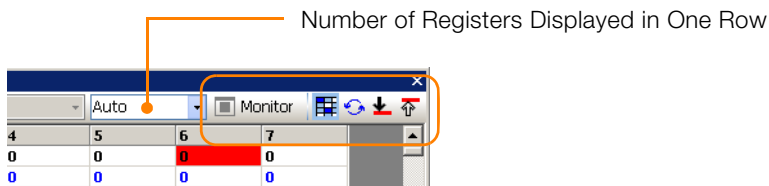
Example of an M Register Map

Information

- If you move the cursor over the register map, a balloon will show the register and the status of the register at the cursor position.
- You can change the number of registers displayed in one row. The five buttons on the top right of the pane are used to switch the displayed contents.
- If you right-click the register list, you can select **Decimal**, **Hexadecimal**, **BIN**, or **ASCII** from the pop-up menu to change the data type of the values. However, the B and F data types cannot be changed.
- The display color alternate between blue and black for every other row.
- The **Monitor** Icon is enabled only when the Machine Controller is online.

5.7.2 Switching the Register Map Display

You can change the number of registers that is displayed in one row. You can use the five buttons on the top right to switch the displayed contents of the register map.



◆ Number of Registers Displayed in One Row

You can set the number of registers displayed in a row to between 1 and 16 either by direct numeric input or by selection from a list. For bit registers, the number is always 16 and cannot be changed. If you select **Auto**, the number of displayed registers will be set automatically based on the size of the Register List Pane.

◆ Monitor ON ()/OFF () Button

This button is enabled only in Online Mode. Click this button to turn monitoring ON and OFF. When monitoring is ON, the register data will be updated and displayed continuously. When monitoring is OFF, the data will not be updated.

◆ Register Map Show ()/Hide () Button

Click this button to show and hide the register map.

Show mode: Registers that are used in the ladder program are displayed with a green background, and registers that are used for more than one data type are displayed with a red background.

Hide mode: All registers are displayed with a white background.

◆ Register Map Refresh Button ()

Click this button to refresh the values in the register map.

Information

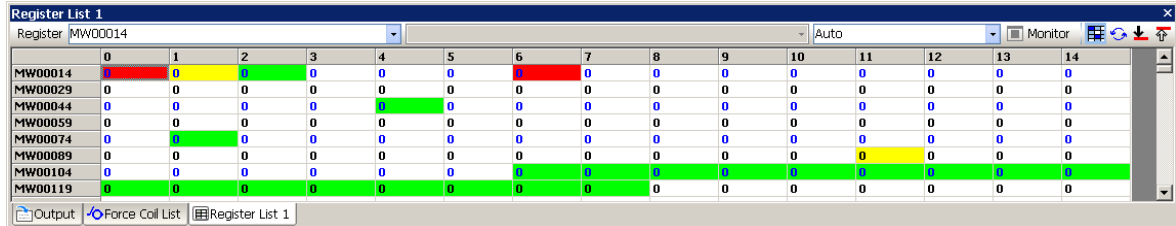
This button is disabled when the above Register Map Show/Hide Button is in Hide () status.

◆ Redundant Register Search Button ( / )

This button searches for and displays redundant registers. The [↑] Button searches for redundant registers upward, and the [↓] Button searches downward.

If the same register is found, it will be displayed in the Register List Pane with a blue background.

Information This button is disabled when the Register Map Show/Hide Button is in Hide () status.



The screenshot shows a window titled "Register List 1" with a dropdown menu set to "Register MW00014". Below the menu is a grid with 15 columns (0-14) and 10 rows of register data. The data values are 0 or 1, and cells are color-coded: red for 1 in column 0, yellow for 1 in column 1, green for 1 in column 2, and blue for 1 in column 3. Other cells are white with a 0. The grid is as follows:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
MW00014	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MW00029	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MW00044	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MW00059	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MW00074	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MW00089	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MW00104	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MW00119	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5.7.3 Editing Data

You can perform the following editing operations by double-clicking cells on the register map or by pressing the F2 Key to display the text cursor.

- Directly entering data
- Deleting data (setting the data to 0)
- Copying and pasting data

Press the **Enter** Key to confirm the change. If the Machine Controller is online, any changes in the data immediately affect the operation of the Machine Controller.

5.8 Tuning Panel

The Tuning Panel allows you to display and edit the current value of pre-registered variables. In addition to the current values, the Tuning Panel also displays comments and visual status indicators.

You can use the Tuning Panel to control and check the operation of your application.

You can adjust the **Visual monitor** Column to display data according to specific conditions.

Variable	Comment	Current v...	Unit	Visual monitor
DW00010 [L]		0		0
X.Ready(IB80000)	X~Motion controller operation ready	OFF		○
Y.Ready(IB80800)	Y~Motion controller operation ready	OFF		○
X.Position.Monitor,A...	X~Machine coordinate feedback posi...	0		■
Y.Position.Monitor,A...	Y~Machine coordinate feedback posit...	0		■
DW00010 [L]		0		0
MB300000	Servo ON PB	OFF		○
MB300001	Alarm reset PB	OFF		○
DW00010 [L]		0		0
DB000010 [H02.01]		OFF		○
DB000011 [H02.01]		OFF		○

5.9 Enabling and Disabling Ladder Programs

Individual drawings in ladder programming can be enabled or disabled.

The diagram is disabled.

- Disabled ladder drawings are not processed.
- ON/OFF control of coils and instruction execution are not processed.

This feature is used to temporarily disable ladder drawings that contain processing to turn ON the power supply to servomotors or jog processing for servomotors. This allows you to check the operation of individual servomotors with the test run operation of the MPE720 or the module configuration definition.

The motor cannot be controlled from the MPE720 because the ladder drawing is being executed.

The motor can be controlled from MPE720 as required.

5.10 Watching

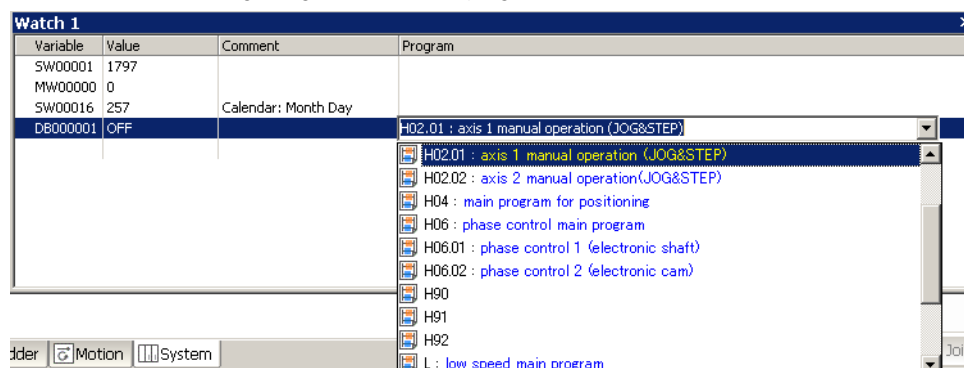
You can monitor the values and comments of the specified S, I, O, M, C, and D registers on the Watch 1, 2, and 3 Panes. Realtime monitoring is possible if the Machine Controller is connected. You can edit the values.

Information When a project link is used, the data registered in the Watch Pane is saved only to the Machine Controller. To apply the watch data to the project file, transfer all of the data from the Machine Controller.

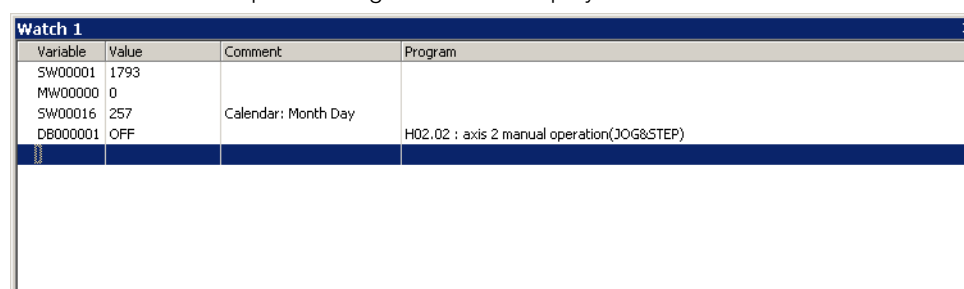
5.10.1 Displaying Watch Data

1. Click one of the tabs for the Watch 1, 2 or 3 Panes.
Select **Monitor – Watch** from the Launcher. The Watch 1 Pane will be displayed.
Note: You can show or hide the Watch 1, 2, and 3 Panes by selecting **View – Watch – Watch 1**, **View – Watch – Watch 2**, or **View – Watch – Watch 3** from the menu bar.
2. Double-click the **Variable** Column or press F2 to display the text cursor, and then enter the register or variable register to monitor.

Note: 1. You can also drag or copy registers from the ladder program or from the Variable Pane.
2. When monitoring D registers, enter the program number as shown below.



3. Press the Enter Key.
The contents of the specified register will be displayed.



If you right-click a row, you can select **Decimal**, **Hexadecimal**, **BIN**, or **ASCII** from the pop-up menu to change the data type of the **Value** Column.

5.10.2 Editing the Value Column

Double-click the **Value** Column or press F2 to display the text cursor. You can enter the value directly or paste a value.

After entering the data, press the **Enter** Key to confirm the change.

Information If the Machine Controller is online, any changes in the data immediately affect the operation of the Machine Controller.

5.11 Security

MPE720 version 7 has the following security features. You can use these security features for data protection by specifying access privileges for individual projects and program drawings.

- **User Administration (User Name and Password Setting)**

You can register and change the name of the users who can open projects.

If the setting is performed while the Machine Controller is online, the setting will provide access privileges to the Machine Controller.

- **Project Password Setting**

You can set a password for opening a project file.

- **Program Password Setting**

You can set a password for opening ladder programs and motion programs. A password can be set for each program.

- **Online Security Setting**

You can set a security key (i.e., a password) and privilege levels for reading data from a Machine Controller. This allows you to restrict the ability to read the program data from the Machine Controller or the ability to open the programs to users who have the specified level of privilege or a higher privilege.

5.12 Tracing

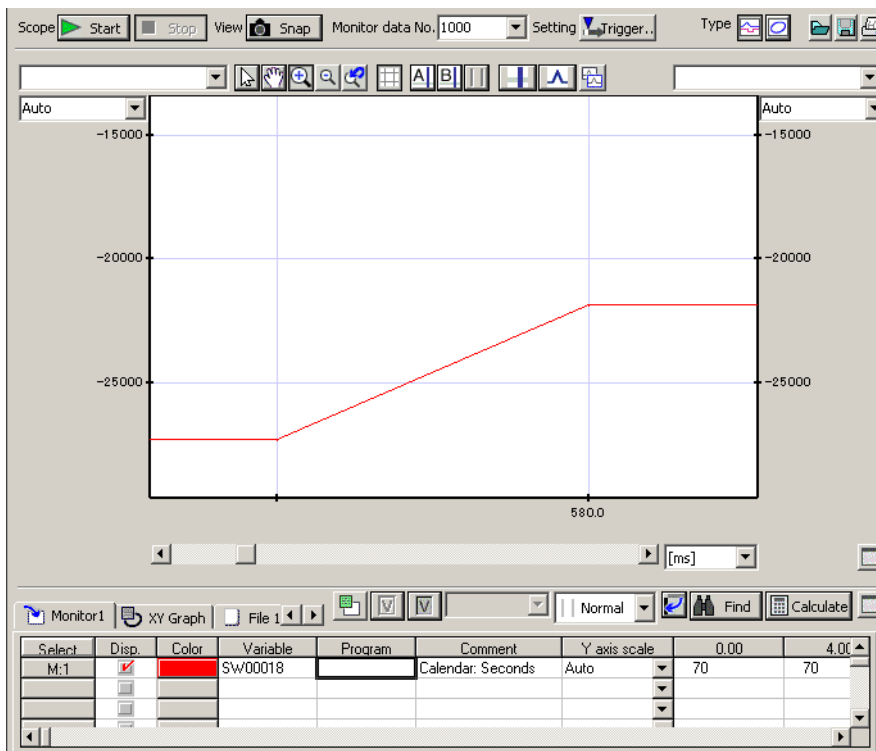
MPE720 version 7 has three trace modes.

- **Real-Time Trace**
You can monitor specified registers on a graph in real time.
- **Trace Manager**
You can have the Machine Controller collect data for specified registers during a specified time period, and perform operations on that data and plot it on a graph. This allows you to analyze register data that is acquired during specific time periods to debug ladder programs.
- **XY Trace**
This trace mode acquires the position data of the X axis and Y axis every scan, and displays the data in a 2-dimensional graph.

All three modes support exporting the trace data to CSV files.

Use tracing to check operation and to debug the ladder programs and motion programs.

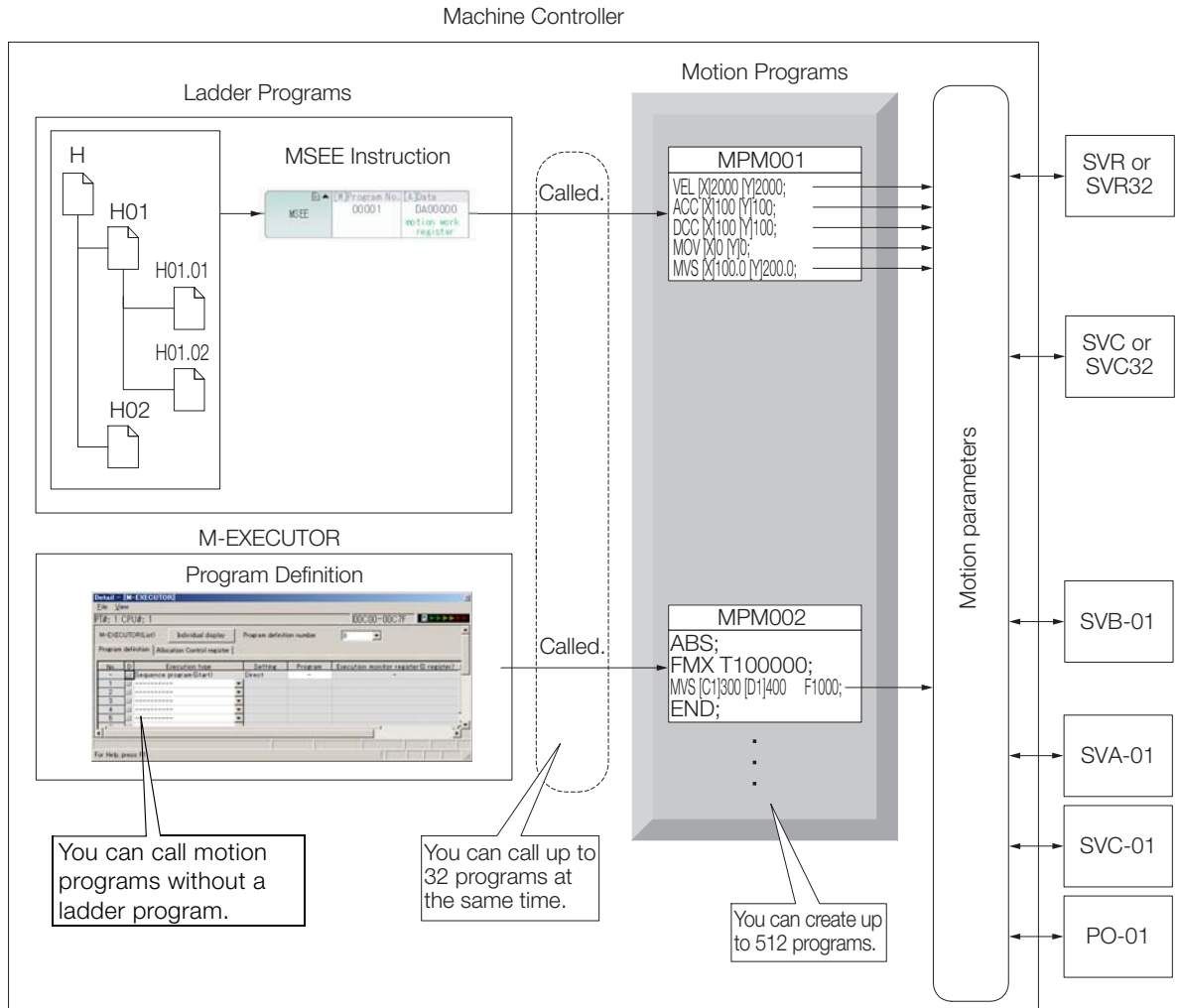
A typical pane for data tracing is shown below.



5.13 Advanced Programming

5.13.1 Motion Programs

A motion program is written in a text-based motion language. In addition to basic motion control and operations, motion programs can also be used to easily program complex movements, such as linear interpolation and circular interpolation. You can execute motion programs either by placing MSEE instructions in ladder programming in DWG.H (high-speed scan process drawings), or by registering the motion programs in Program Definition Tab Page for the M-EXECUTOR Module.



Refer to the following manual for details on motion programs.
 📖 MP3000 Series Motion Programming Manual (Manual No. SIEP C880725 14)

System Service Registers

Appendix **A**

This appendix describes the system service registers that are part of the system registers that are provided with the Machine Controller system.

A.1	Overview of System Registers	A-2
A.2	Common to All Drawings	A-3
A.3	Exclusive to DWG.H (High-speed Scan Process Drawings) . .	A-4
A.4	Exclusive to DWG.L (Low-speed Scan Process Drawings) . .	A-5
A.5	Scan Execution Status and Calendar	A-6
A.6	System Program Software Numbers and Remaining Program Memory Capacity . .	A-7

A.1 Overview of System Registers

System registers are provided by the Machine Controller system. They can be used to read system error information, the current operating status, and other information.

	Contents
SW000000	System Service Registers
SW000030	System Status
SW000050	System Error Status
SW000080	Overview of User Operation Error Status
SW000090	System Service Execution Status
SW000110	Detailed User Operation Error Status
SW000190	Alarm Counter and Alarm Clear
SW000200	System I/O Error Status
SW000504	Reserved for system.
SW000652	CF Card-related System Registers (MP2200-series CPU-02 and CPU-03 only)
SW000698	Interrupt Status
SW000800	Module Information
SW001312	Reserved for system.
SW001411	MPU-01 Module System Status
SW002048	Reserved for system.
SW003200	Motion Program Information
SW005200 to SW008191	Reserved for system.

The System Service Registers are grouped into the following five categories.

- Common to All Drawings
- Exclusive to DWG.H (high-speed scan process drawings)
- Exclusive to DWG.L (low-speed scan process drawings)
- Scan Execution Status and Calendar
- System Program Software Numbers and Remaining Program Memory Capacity

A.2 Common to All Drawings

Name	Register Address	Remarks
Reserved for system.	SB000000	Not used.
High-speed Scan	SB000001	This register is ON for only the first scan after the high-speed scan starts.
Low-speed Scan	SB000003	This register is ON for only the first scan after the low-speed scan starts.
Always ON	SB000004	Always ON (1).
Reserved for system.	SB000005, SB000006	Not used.
High-speed Scan in Progress	SB000007	ON (1) during execution of the high-speed scan.
Reserved for system.	SB000008 to SB00000F	Not used.

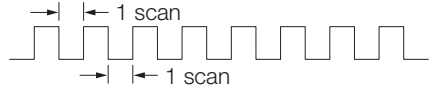
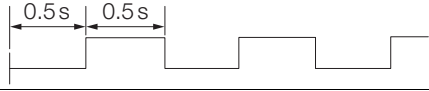
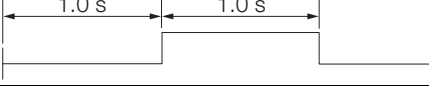
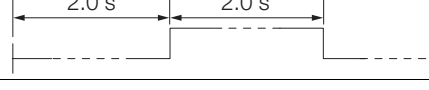
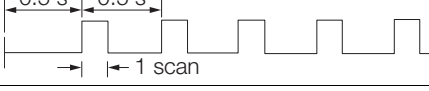
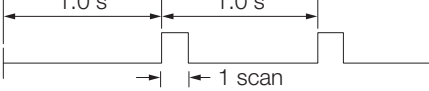
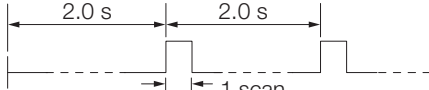
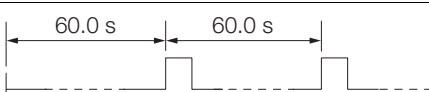



A.3 Exclusive to DWG.H (High-speed Scan Process Drawings)

Operation starts when the high-speed scan starts.

Name	Register Address	Remarks
1-scan Flicker Relay	SB000010	
0.5-s Flicker Relay	SB000011	
1.0-s Flicker Relay	SB000012	
2.0-s Flicker Relay	SB000013	
0.5-s Sampling Relay	SB000014	
1.0-s Sampling Relay	SB000015	
2.0-s Sampling Relay	SB000016	
60.0-s Sampling Relay	SB000017	
1.0 s After Start of Scan Relay	SB000018	
2.0 s After Start of Scan Relay	SB000019	
5.0 s After Start of Scan Relay	SB00001A	

A.4 Exclusive to DWG.L (Low-speed Scan Process Drawings)

Operation starts when the low-speed scan starts.

Name	Register Address	Remarks
1-scan Flicker Relay	SB000030	
0.5-s Flicker Relay	SB000031	
1.0-s Flicker Relay	SB000032	
2.0-s Flicker Relay	SB000033	
0.5-s Sampling Relay	SB000034	
1.0-s Sampling Relay	SB000035	
2.0-s Sampling Relay	SB000036	
60.0-s Sampling Relay	SB000037	
1.0 s After Start of Scan Relay	SB000038	
2.0 s After Start of Scan Relay	SB000039	
5.0 s After Start of Scan Relay	SB00003A	

A.5 Scan Execution Status and Calendar

Name	Register Address	Remarks
High-speed Scan Set Value	SW00004	This is the high-speed scan set value (0.1 ms).
Current High-speed Scan Time	SW00005	This is the current value of the high-speed scan (0.1 ms).
High-speed Scan Maximum Value	SW00006	This is the maximum value of the high-speed scan (0.1 ms).
High-speed Scan Set Value 2	SW00007	This is the high-speed scan set value (1 μ s).
Current High-speed Scan Time 2	SW00008	This is the current value of the high-speed scan (1 μ s).
High-speed Scan Maximum Value 2	SW00009	This is the maximum value of the high-speed scan (1 μ s).
Low-speed Scan Set Value	SW00010	This is the low-speed scan set value (0.1 ms).
Current Low-speed Scan Time	SW00011	This is the current value of the low-speed scan (0.1 ms).
Low-speed Scan Maximum Value	SW00012	This is the maximum value of the low-speed scan (0.1 ms).
Reserved for system.	SW00013	Not used.
Current Scan Time	SW00014	This is the current value of the scan that is currently being executed (0.1 ms).
Calendar: Year	SW00015	1999: 0099 (BCD) (last two digits only)
Calendar: Month Day	SW00016	December 31: 1231 (BCD)
Calendar: Hour and Minutes	SW00017	23:59: 2359 (BCD)
Calendar: Seconds	SW00018	59 s: 59 (BCD)
Calendar: Week	SW00019	0: Sunday, 1: Monday, 2: Tuesday, 3: Wednesday, 4: Thursday, 5: Friday, and 6: Saturday

A.6**System Program Software Numbers and Remaining Program Memory Capacity**

Name	Register Address	Remarks
System Program Software Number	SW00020	Sxxxx (xxxx is replaced with the BCD value.)
System Number	SW00021 to SW00025	Not used.
Remaining Program Memory Capacity	SL00026	Bytes
Total Memory Capacity	SL00028	Bytes

Sample Programs

Appendix **B**

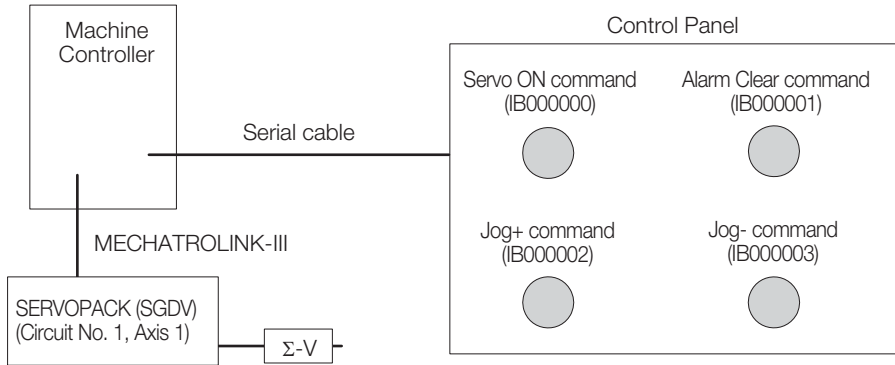
This appendix describes ladder programming examples that perform test runs.

- B.1 Jogging from the Control PanelB-2**
- B.2 Motion Program ControlB-3**
- B.3 Simple Synchronized Operation of Two Axes with a Virtual Axis . .B-4**

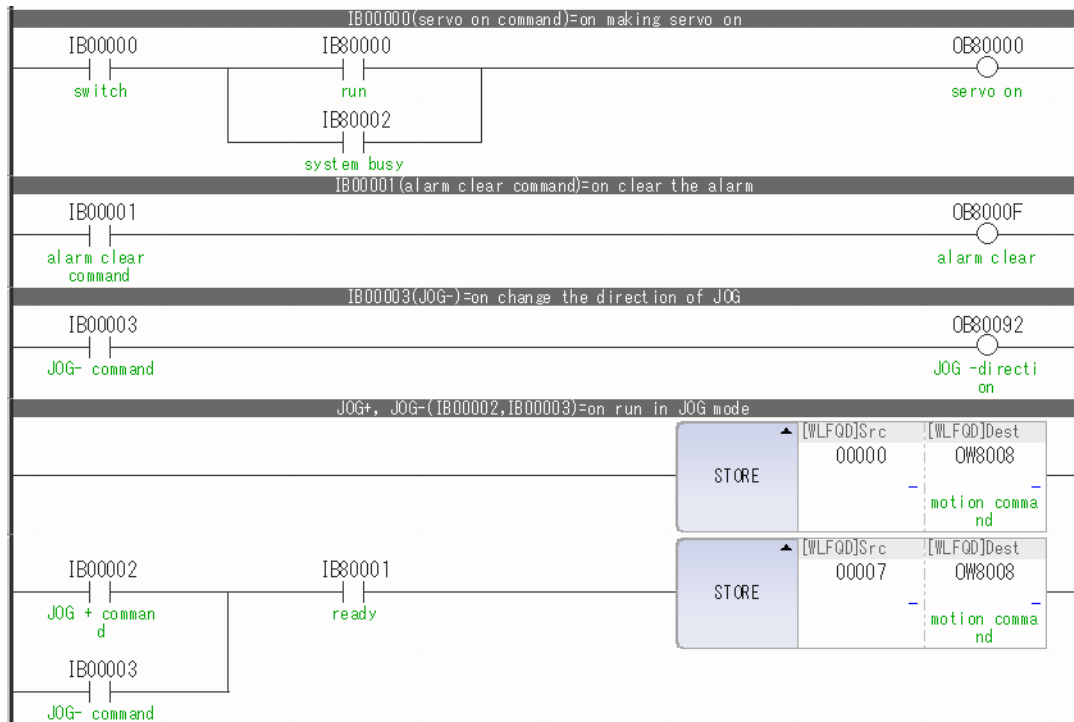
B.1 Jogging from the Control Panel

The following configuration and ladder programming example illustrate how to control a motor from switches on a control panel when the motor and control panel are connected to a Machine Controller.

■ Configuration Example

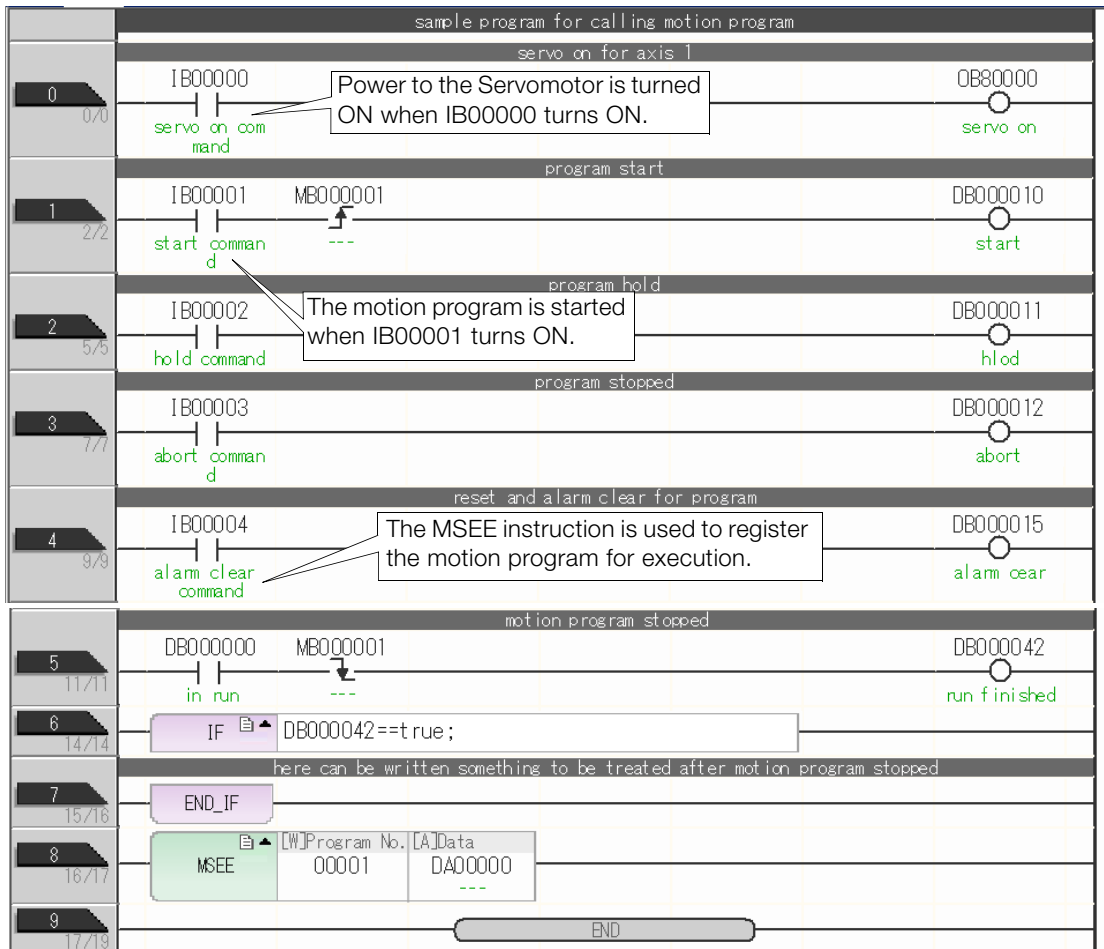


■ Ladder Programming Example



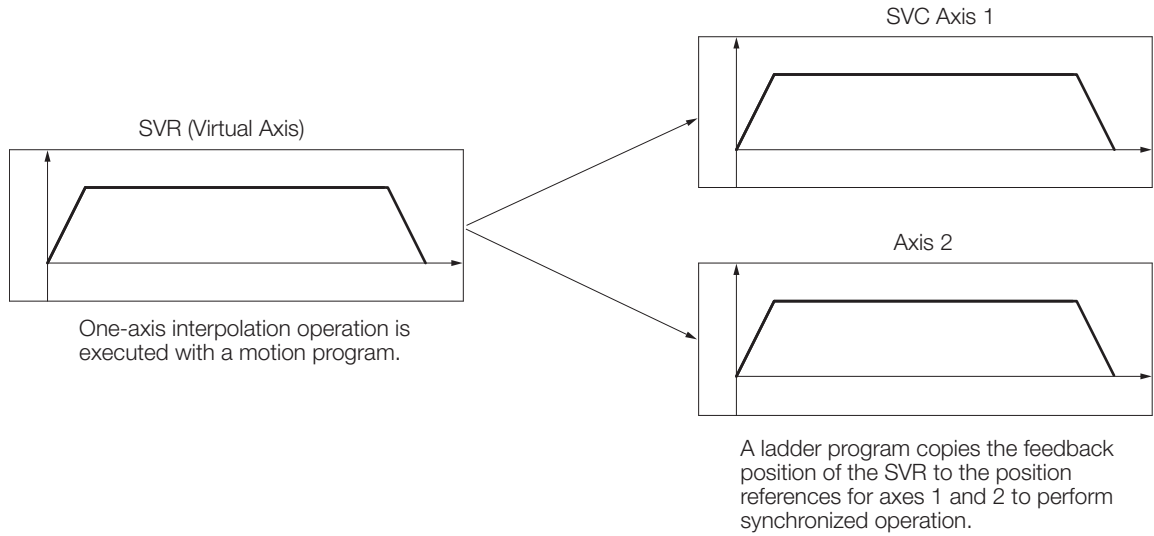
B.2 Motion Program Control

The following ladder programming example demonstrates how to control execution of a motion program.



B.3 Simple Synchronized Operation of Two Axes with a Virtual Axis

A motion program moves an SVR (virtual axis) and a ladder program distributes the feedback position of the SVR to two physical axes to perform synchronized operation with two axes.

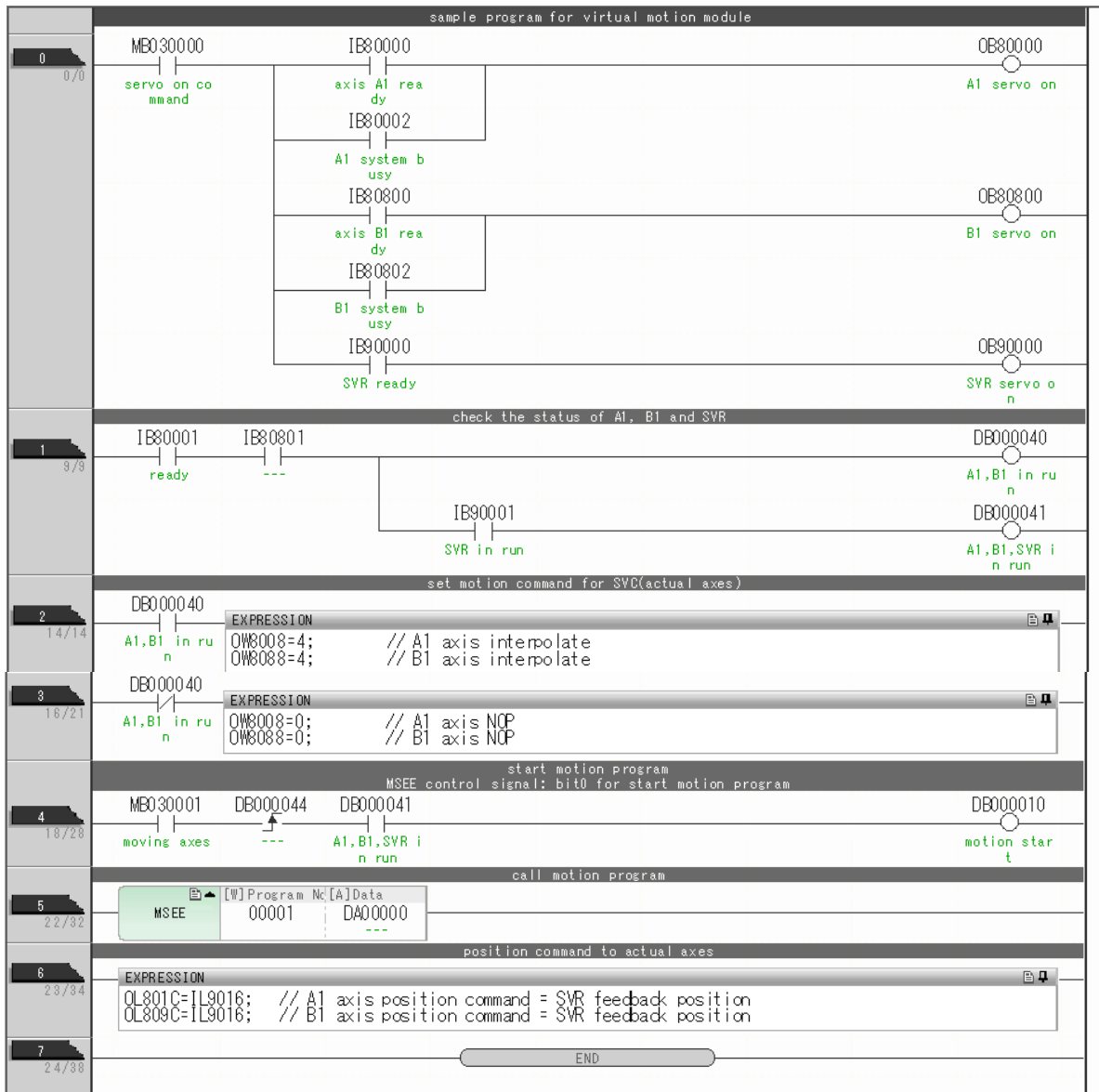


The motion programming example and ladder programming example for the above operation are given below.

■ Motion Programming Example

```
FMX T1000K;           "Set maximum interpolation speed K = 1,000.  
INC;                  "Incremental Mode  
IAC T500;             "Interpolation acceleration time = 500 ms  
IDC T500;             "Interpolation deceleration time = 500 ms  
MVS [SVR] 1000K F10000K; "Interpolation for travel distance of 1,000,000  
END;
```

■ Ladder Programming Example



This programming example does not include recovery processing for axis errors. If you decide to incorporate this programming example into your application, be sure to add the necessary programming to ensure safe operation in the event of an axis error.

Format for EXPRESSION Instructions

Appendix C

This appendix describes the format for EXPRESSION instructions.

C.1 Elements That You Can Use in Numeric Expressions . . . C-2

C.1.1	Operators	C-2
C.1.2	Operands	C-3
C.1.3	Instructions That You Can Use with EXPRESSION Instructions	C-4

C.2 Notational Limitations C-5

C.2.1	Arithmetic and Logic Operators	C-5
C.2.2	Comparison Operators	C-5
C.2.3	Logic Operators	C-5
C.2.4	Substitution Operator	C-6
C.2.5	Functions	C-6
C.2.6	Parentheses	C-6

C.1 Elements That You Can Use in Numeric Expressions

Numeric expressions that can be used in EXPRESSION instructions include operators, operands (constants and variables), and functions. This section describes each of these elements.

C.1.1 Operators

Types of Operators and Usable Operators

The following list gives the types of operators and usable operators.

Type	Usable Operators	
Arithmetic and Logic Operators	+	Add
	-	Subtract
	*	Multiply
	/	Divide
	%	Remainder
	&	Bit-wise AND
		Bit-wise OR
	++	Extended Add
Logic Operators (Usable only with bit data)	--	Extended Subtract
	&&	Inclusive AND
		Inclusive OR
Comparison Operators	!	Logical NOT
	==	Equal to right-side value
	!=	Unequal to right-side value
	>	Greater than right-side value
	>=	Greater than or equal to right-side value
	<	Less than right-side value
Substitution Operator	<=	Less than or equal to right-side value
	=	Substitutes left-side value with right-side value
Reserved Words	true	TRUE for a logical expression
	false	FALSE for a logical expression
Control Instructions	IF, ELSE, and IEND	ELSE can be omitted.

Order of Evaluation

Operators are evaluated according to their processing priority and the order in which operands are grouped, as listed below.

Priority	Operators	Description	Grouping Order	
High	[] ()	Expression	Left to right	
	- !	Unary	Right to left	
↑	* / %	Multiplication, division, and remainder	Left to right	
	+ - ++ --	Addition, subtraction, extended addition, and extended subtraction		
	< > <= >=	Relational		
	== !=	Equivalence		
	&	Bit-wise AND		
↓		Bit-wise OR		
	&&	Inclusive AND		
		Inclusive OR		
Low				

Note: Operators on the same line have the same processing priority and are evaluated according to their grouping order.

C.1.2 Operands

Constants

Integers or real numbers may be used as a constant.

- An integer may be any number that can be expressed within the range of a 64-bit integer (quadruple-precision integers).
(-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)
- A real number may be any number that can be expressed within the range of 64-bit data (double-precision real numbers).
± (2.225E-308 to 1.798E+308)

Information Hexadecimal numbers must be expressed using the `0x□□□□` notation when used in the `EXPRESSION`, `IF`, or `WHILE` instruction. The `H□□□□` notation will result in an error.
Example: `H012F ... NG` `0x012F ... OK`
The `H□□□□` notation must be used for all other instructions, such as the `STORE` instruction.

Variables


The `EXPRESSION` instruction allows you to assign arbitrary variable names that are allowed in C language to registers in the Machine Controller.

Although the C language does not have Boolean variables, bit registers in the Machine Controller are treated as Boolean variables. Boolean variables are either `TRUE` or `FALSE` and can be used only in logical expressions.

■ Limitations on Variable Names

The following limitations apply to variable names.

- Variable names must start with a non-numeric character.
- For ASCII characters, only alphabetic characters, underscores, and numbers may be used.
- The following variable names cannot be used because they are already used as function names.

 *C.1.3 Instructions That You Can Use with `EXPRESSION` Instructions on page C-4*

Example `AbcOK`
`Get_input()OK`
`1abNG`
`SinNG`

C.1.3 Instructions That You Can Use with EXPRESSION Instructions

The following list gives the instructions that can be used with EXPRESSION instructions.

Instruction	Description	Example	Reserved Word
+	Add	MW00001 = MW00002 + MW00003	√
-	Subtract	MW00001 = MW00002 - MW00003	√
*	Multiply	MW00001 = MW00002 × MW00003	√
/	Divide	MW00001 = MW00002 / MW00003	√
%	Remainder	MW00001 = MW00002 % MW00003	√
&	Bit-wise AND	MW00001 = MW00002 & 4096	√
	Bit-wise OR	MW00001 = MW00002 4096	√
++	Extended Add	MW00001 = MW00002 ++ MW00003	√
--	Extended Subtract	MW00001 = MW00002 -- MW00003	√
&&	Inclusive AND	MB000010 = MB000011 && MB000012	√
	Inclusive OR	MB000010 = MB000011 MB000012	√
!	Logical NOT	MB000010 = !MB000011	√
==	Equal to right-side value	MB000010 = MB000011 == true	√
>=	Right-side value is greater than or equal to left-side value	MB000010 = MW00020 >= MW00021	√
>	Right-side value is greater than left-side value	MB000010 = MW00020 > MW00021	√
<	Right-side value is less than left-side value	MB000010 = MW00020 < MW00021	√
<=	Right-side value is less than or equal to left-side value	MB000010 = MW00020 <= MW00021	√
=	Substitute left-side value with right-side value	MW00001 = MW00002	√
true	TRUE	MB000010 = MB000011 == true	√
false	FALSE	MB000010 = MB000011 == false	√
sin()	SIN	MW00001 = sin(MW00002)	√
cos()	COS	MF00002 = cos(MF00004)	√
atan()	ARCTAN	MF00002 = atan(MF00004)	√
tan()	TAN	MF00002 = tan(MF00004)	√
()	Parentheses	MW00001 = (MW00002 + MW00003) / MW00004	√
asin()	ARCSIN	MF00002 = asin(MF00004)	√
acos()	ARCCOS	MF00002 = acos(MF00004)	√
sqrt()	SQRT	MW00001 = sqrt(MW00002)	√
abs()	ABS	MW00001 = abs(MW00002)	√
exp()	EXP	MF00002 = exp(MF00004)	√
log()	LOG natural logarithm	MF00002 = log(MF00004)	√
log10()	LOG10 common logarithm	MF00002 = log10(MF00004)	√

C.2 Notational Limitations

Several limitations apply when combining operands and operators to form numeric expressions. An expression is not recognized as a numeric expression unless it meets these conditions.

This section describes these limitations.

C.2.1 Arithmetic and Logic Operators

These operators can be used with integer and real number operands. The unary minus operator can be used only once. Bit operations can be performed only on integer data. Bit operands cannot be used for arithmetic operations. No automatic data type conversion is performed even if the calculation result exceeds the range of the assigned register. Therefore, the user must assign the appropriate data type to the register.

Example

```
MW00001 = MW00002 + MW00003OK
MW00001 = MW00002 / 345OK
MF00002 = (MW00004 + MF00002) / (ML00018 + MW00008)OK
MW00001 = MW00002 & 4096OK
MB000010 = MB000011 - MB000012NG
MW00001 = MB000011 * MW00001NG
```



Important

To perform bit operations, match the data types on the left and right sides of the operator. If the operation is performed using different data types, the intended result may not be obtained.

```
ML00000 = MW00002 | ML00004NG
ML00000 = ML00002 | ML00004OK

MQ00000 = 0xFFFF0000 & MQ00004NG
MQ00000 = 0x00000000FFFF0000 & MQ00004OK
```

C.2.2 Comparison Operators

These operators can be used with integer and real number operands. The left side must be a bit data register. To use an integer bit operand in a comparison operation with the == or != operator, compare it with TRUE or FALSE.

Example

```
MB000010 = MW00002 != MW00003OK
MB000010 = MF00002 < 99.99OK
MB000010 = MW00002 >= MW00003OK
MB000010 = MB000011 == trueOK
MB000010 = MB000011 != 0NG
MB000010 = MB000011 == 1NG
```

C.2.3 Logic Operators

These operators can be used with bit operands.

Example

```
MB000010 = MB000011 && MB000012OK
MB000010 = !MB000011OK
MB000010 = (MW000020 >= 50) && MB000011OK
MB000010 = MW00001 || MW00002 NG
MB000010 = !MW00001NG
```

C.2.4 Substitution Operator

Real number and integer registers can be substituted with either real number or integer data, even if the data type differs. When you substitute an integer register with a real number register, a round-off error will occur.

Bit registers can be substituted only with logical values, such as another bit register or a TRUE/FALSE. If you substitute a bit register with a non-logical value, that value will be compared against 0 or 0.0 and the TRUE or FALSE outcome will be converted to a code before it is substituted.

Bit data cannot be substituted into non-bit registers.

Example MW00001 = MW00002;OK
 MF00000 = MW00002 / 345;OK
 MB000010 = MB000010; OK
 MW00010 = MB0000101;NG
 MW00001 = true;NG

C.2.5 Functions

The arguments and return values for functions depend on the specifications of the functions in the Machine Controller.

Therefore, if the input for the sin(), cos(), and tan() functions is an integer or integer register, the output value will be returned as an integer. If the input is a real number or a real number register, the output value will be returned as a real number.

The argument for the tan() function is a real number so an integer register input will be treated as a real number.

Example MW00001 = sin(MW00002);OK
 MF00002 = cos(MF00000×3.14);OK
 MW00001 = -atan(MF00002);OK

C.2.6 Parentheses

■ Grouping

You can group multiple expressions by enclosing them with parenthesis ().

Example MW00001 = -(MW00002 + 10) / (MW00003 – MW00005); OK

■ Arrays

You can specify arrays by using square brackets [], just like with the C language.

Example MW00001 = MW00002[100];OK
 MW00001 = MW00002[MW00003];OK
 MB000010 = MB000020[0];OK

Precautions on Motion Parameters

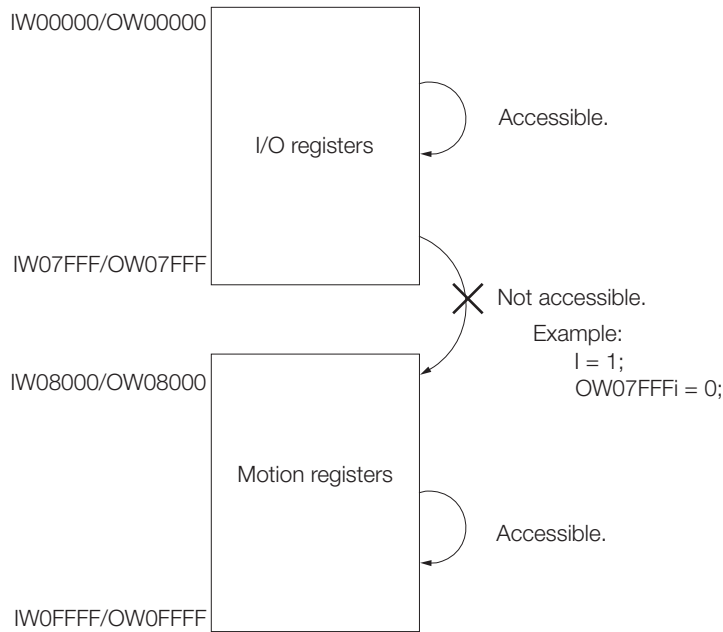
Appendix **D**

This appendix provides precautions on motion parameters.

The following precautions apply to using motion parameters.

■ **Do Not Use a Subscript to Reference a Motion Register from an I/O Register**

I/O registers and motion registers are not assigned to consecutive memory locations. When using a subscript, access registers within the range of I/O registers or within the range of motion registers.



■ **Do Not Use a Subscript to Reference a Motion Register in a Different Circuit**

Motion registers on different circuits are not assigned to consecutive memory locations.

When using a subscript, access registers within the range of motion registers for each circuit.

If the circuit number is the same, it is possible to access motion registers for different axes.

Circuit No.	Axis 1	Axis 2	...	Axis 16
1	OW08000 to OW0807F	OW08080 to OW080FF	...	OW08780 to OW087FF
3	OW09000 to OW0907F	OW09080 to OW090FF	...	OW09780 to OW097FF
5	OW0A000 to OW0A07F	OW0A080 to OW0A0FF	...	OW0A780 to OW0A7FF
7	OW0B000 to OW0B07F	OW0B080 to OW0B0FF	...	OW0B780 to OW0B7FF
9	OW0C000 to OW0C07F	OW0C080 to OW0C0FF	...	OW0C780 to OW0C7FF
11	OW0D000 to OW0D07F	OW0D080 to OW0D0FF	...	OW0D780 to OW0D7FF
13	OW0E000 to OW0E07F	OW0E080 to OW0E0FF	...	OW0E780 to OW0E7FF
15	OW0F000 to OW0F07F	OW0F080 to OW0F0FF	...	OW0F780 to OW0F7FF
17	OW18000 to OW1807F	OW18080 to OW180FF	...	OW18780 to OW187FF
19	OW19000 to OW1907F	OW19080 to OW190FF	...	OW19780 to OW197FF
21	OW1A000 to OW1A07F	OW1A080 to OW1A0FF	...	OW1A780 to OW1A7FF
23	OW1B000 to OW1B07F	OW1B080 to OW1B0FF	...	OW1B780 to OW1B7FF
25	OW1C000 to OW1C07F	OW1C080 to OW1C0FF	...	OW1C780 to OW1C7FF
27	OW1D000 to OW1D07F	OW1D080 to OW1D0FF	...	OW1D780 to OW1D7FF
29	OW1E000 to OW1E07F	OW1E080 to OW1E0FF	...	OW1E780 to OW1E7FF
31	OW1F000 to OW1F07F	OW1F080 to OW1F0FF	...	OW1F780 to OW1F7FF

Circuit 1	Axis 1 (IW08000 to IW0807F, OW08000 to OW0807F)
	Axis 2 (IW08080 to IW080FF, OW08080 to OW080FF)
	⋮
	Axis 16 (IW08780 to IW087FF, OW08780 to OW087FF)

Accessible.

Example:

I = 1;
OW0807Fi = 0;

Circuit 3	Axis 1 (IW09000 to IW0907F, OW09000 to OW0907F)
	Axis 2 (IW09080 to IW090FF, OW09080 to OW090FF)
	⋮
	Axis 16 (IW09780 to IW097FF, OW09780 to OW097FF)

Not accessible.

Example:

I = 1;
OW087FFi = 0;

Machine Controller Specifications

Appendix E

This appendix provides the specifications for programs for the Motion Controller.

The following table gives the specifications for programs for the Machine Controller.

Specification	CPU Unit/CPU Module	Remarks
Number of Programs	512 max.	You can create a combined total of 512 motion programs and sequence programs.
Number of Groups	16 groups	—
Number of Tasks	32 tasks max. (This is the number of simultaneously executable motion programs.)	—
Number of Parallel Forks per Task	8 parallel forks max. (select from these 4 modes) <ul style="list-style-type: none"> • 4 main program forks, 2 subprogram forks • 8 main program forks, 1 subprogram forks • 2 main program forks, 4 subprogram forks • 1 main program forks, 8 subprogram forks 	Change the mode using the MPE720.
Execution Registration	<ul style="list-style-type: none"> • Use the MSEE instruction from a ladder program. • Use the M-EXECUTOR. 	—
Starting Method	Program execution starts on the rising edge of bit 0 (Request for Start of Program Operation) in the control signals.	—
Override for Positioning Speeds	Can be specified from 0.01% to 327.67%.	—
Operating Modes	Absolute Mode and Incremental Mode	The mode is changed with the ABS and INC instructions.
Reference Unit	<ul style="list-style-type: none"> • SVC/SVC32/SVC-01/SVB-01/SVR/SVR32 pulse, mm, deg, inch, μm • SVA-01/PO-01 pulse, mm, deg, inch 	—
Minimum Reference Unit	<ul style="list-style-type: none"> • pulse 1 • mm, deg, inch, μm 1, 0.1, 0.01, 0.001, 0.0001, 0.00001 	—
Reference Range	-2147483648 to +2147483647 (32-bit signed data)	—
Number of Simultaneously Controlled Axes per Task	<ul style="list-style-type: none"> • Positioning, Linear Interpolation, Zero Point Return, Skip Function, and Set-time Positioning 32 axes max. • Circular Interpolation 2 axes • Helical Interpolation 3 axes • External Positioning 1 axis 	—
Number of Simultaneously Controlled Cameras	32 cameras max.	—
Number of Simultaneously Controlled Mechanisms	4 mechanisms max.	—

Continued on next page.

Continued from previous page.

	Specification	CPU Unit/CPU Module	Remarks
Sequence Programs	Number of Programs	512 max. (There are three settings for the execution timing: startup processing, high-speed scan processing, or low-speed scan processing.)	You can create a combined total of 512 motion programs and sequence programs.
	Number of Tasks	32 tasks max. (This is the number of simultaneously executable sequence programs.)	–
	Number of Parallel Forks per Task	The PFORK instruction cannot be used.	–
	Execution Registration	Use the M-EXECUTOR.	–
	Starting Method	Automatically started by the system.	The system starts sequence programs that are registered in the M-EXECUTOR.
Accessible Registers	M Registers	1,048,576 words	These registers are backed up with a battery.
	S Registers	65,535 words	These registers are backed up with a battery.
	G Registers	2,097,152 words	These registers are shared by all programs. They are not backed up with a battery.
	I Registers	65,536 words + Setting parameters + Registers for CPU interface	–
	O Registers	65,536 words + Monitor parameters + Registers for CPU interface	–
	C Registers	16,384 words	–
	D Registers	Can be specified from 0 to 16,384 words.	These are internal registers that are unique within each DWG. They can be referenced only within the local drawing.

Error Codes

Appendix **F**

This appendix describes the error codes that correspond to the storage operation instructions.

Error Code	Description	Instructions in Which This Error Occurs
0000 hex	No error	–
8000 hex	Param is outside range of registers	FOPEN, FCLOSE, FREAD, FWRITE, FSEEK, FGETS, FPUTS, FCOPY, FREMOVE, FRENAME, DCREATE, DREMOVE, FTTPUT
8101 hex	Drive number out of range error	FTTPUT
810B hex	Text string error (NULL character not detected)	FPUTS
810C hex	File or directory name error	FOPEN, FCOPY, FREMOVE, FRENAME, DCREATE, DREMOVE, FTTPUT
810D hex	FTP transmission error	FTTPUT
8110 hex	Invalid file handler	FOPEN, FCLOSE, FREAD, FWRITE, FSEEK, FGETS, FPUTS
8111 hex	Size out of range error	FREAD, FWRITE
8113 hex	Storage or read destination registers out of range error	FOPEN, FREAD, FWRITE, FGETS, FCOPY, FREMOVE, FRENAME, DCREATE, DREMOVE, FTTPUT
8114 hex	Offset out of range error	FSEEK
8115 hex	Origin out of range error	FSEEK
8116 hex	Open type out of range error	FOPEN
8201 hex	No USB memory device	FOPEN, FCOPY, FREMOVE, FRENAME, DCREATE, DREMOVE, FTTPUT
8202 hex	Cannot open file (e.g., invalid path, inaccessible, or insufficient space)	FOPEN
8203 hex	File seek error (inaccessible)	FSEEK
8204 hex	File write error (inaccessible or insufficient space)	FWRITE, FPUTS
8205 hex	File read error (inaccessible)	FREAD, FGETS
8206 hex	File close failure (inaccessible)	FCLOSE
8207 hex	Cannot save file (invalid path, inaccessible, or insufficient space)	FCOPY, FRENAME
8208 hex	File or directory deletion failure	FREMOVE, DREMOVE
8209 hex	Cannot create directory (invalid path, inaccessible, or insufficient space)	DCREATE
820A hex	Cannot open write-protected file	FOPEN
820B hex	Exceeded number of files that can be opened simultaneously	FOPEN, FCOPY, FREMOVE, FRENAME
820C hex	Exceeded number of workspaces that can be used simultaneously	DCREATE, DREMOVE, FTTPUT
820D hex	Cannot execute processing because target file is being used in another instruction	FOPEN, FCLOSE, FREAD, FWRITE, FSEEK, FGETS, FPUTS, FCOPY, FREMOVE, FRENAME
820E hex	File already open	FOPEN
820F hex	Preparation for storage operation processing not completed	FOPEN, FCLOSE, FREAD, FWRITE, FSEEK, FGETS, FPUTS, FCOPY, FREMOVE, FRENAME, DCREATE, DREMOVE, FTTPUT
8210 hex	Directory was specified	FOPEN, FREMOVE, FCOPY, FRENAME, FTTPUT
8211 hex	Attempted to overwrite file or directory	FCOPY, FRENAME, DCREATE

Continued on next page.

Continued from previous page.

Error Code	Description	Instructions in Which This Error Occurs
8212 hex	File or directory does not exist	FOPEN, FCOPY, FREMOVE, FRENAME, DREMOVE, FTTPUT

Index

Symbols

registers - - - - - 3-4

Numerics

10-ms OFF-Delay Timer (TOFF (10 ms)) - - - - - 4-21
 10-ms ON-Delay Timer (TON (10 ms)) - - - - - 4-19
 1-ms OFF-Delay Timer (TOFF (1 ms))- - - - - 4-18
 1-ms ON-Delay Timer (TON (1 ms))- - - - - 4-16
 1-s OFF-Delay Timer (TOFF (1 s)) - - - - - 4-24
 1-s ON-Delay Timer (TON (1 s)) - - - - - 4-22

A

Absolute Value (ABS) - - - - - 4-56
 Add (ADD (+)) - - - - - 4-35
 Add Time (TMADD) - - - - - 4-48
 address - - - - - 3-8
 Arc Cosine (ACOS) - - - - - 4-99
 Arc Sine (ASIN) - - - - - 4-98
 Arc Tangent (ATAN) - - - - - 4-100
 arithmetic operators - - - - - C-2, C-5
 ASCII Conversion 1 (ASCII) - - - - - 4-60
 ASCII Conversion 2 (BINASC) - - - - - 4-61
 ASCII Conversion 3 (ASCBIN) - - - - - 4-62

B

background - - - - - 1-10
 Basic Function Instructions - - - - - 4-8, 4-93
 batch transfer - - - - - 2-12
 BCD Conversion (BCD) - - - - - 4-58
 Binary Conversion (BIN) - - - - - 4-57
 Binary Search (BSRCH) - - - - - 4-116
 bit- - - - - 3-8
 Bit Rotate Left (ROTL) - - - - - 4-104
 Bit Rotate Right (ROTR) - - - - - 4-105
 Bit Shift Left (SHFTL) - - - - - 4-118
 Bit Shift Right (SHFTR) - - - - - 4-120
 Byte Swap (BSWAP) - - - - - 4-122
 Byte-to-word Expansion (BEXTD) - - - - - 4-113

C

Call Extended Program (XCALL) - - - - - 4-83
 Call Motion Program (MSEE) - - - - - 4-76
 Call Sequence Program (SEE) - - - - - 4-75
 Call User Function (FUNC) - - - - - 4-78
 calling user functions - - - - - 1-17
 checking for multiple coils - - - - - 5-13
 checking the operation of the ladder programs - - - - - 2-13
 child drawings - - - - - 1-7
 Clear Queue Table Pointer (QTBLC) - - - - - 4-202

Clear Table Block (TBLCL) - - - - - 4-187
 Coil (COIL) - - - - - 4-29
 Common Logarithm (LOG) - - - - - 4-103
 comparison operators - - - - - C-2, C-5
 compiling the program - - - - - 2-8
 connecting the hardware - - - - - 2-3
 constant registers - - - - - 3-3
 constants - - - - - C-3
 control instructions - - - - - C-2
 controlling the execution of drawings - - - - - 1-9
 Copy Word (COPYW) - - - - - 4-121
 Cosine (COS) - - - - - 4-95
 Counter (COUNTER) - - - - - 4-204
 creating a project - - - - - 2-4
 creating ladder programs - - - - - 2-7
 creating table data - - - - - 1-18
 creating user functions - - - - - 1-15
 cross reference criteria - - - - - 5-10
 cross references - - - - - 5-10

D

D registers - - - - - 3-4
 Data Manipulation Instructions - - - - - 4-8
 data registers - - - - - 3-2
 Data Shift Instructions - - - - - 4-104
 data types - - - - - 3-8
 DDC Instructions - - - - - 4-8, 4-123
 Dead Zone A (DZA) - - - - - 4-123
 Dead Zone B (DZB) - - - - - 4-124
 Decrement (DEC) - - - - - 4-47
 dialog box
 Replace - - - - - 5-6
 Replace in the Project - - - - - 5-8
 Search - - - - - 5-5
 Search in Project - - - - - 5-7
 Direct Input String (INS) - - - - - 4-79
 Direct Output String (OUTS) - - - - - 4-81
 displaying watch data - - - - - 5-23
 Divide (DIV (+)) - - - - - 4-42
 double-length integer - - - - - 3-8
 double-precision real number - - - - - 3-8
 DWG.A - - - - - 1-8
 DWG.H - - - - - 1-8
 DWG.I - - - - - 1-8
 DWG.L - - - - - 1-8

E

enabling and disabling ladder diagrams - - - - - 5-22
 Equal (=) - - - - - 4-69
 Exchange (XCHG) - - - - - 4-110
 Exclusive OR (XOR) - - - - - 4-66
 execution processing of drawings - - - - - 1-9

- Exponential (EXP) - - - - - 4-101
Export (EXPORT/EXPORTL/EXPORTLE) - - - - - 4-248
Expression (EXPRESSION) - - - - - 4-91
EXPRESSION instructions - - - - - C-2
Extended Add (ADDX (++)) - - - - - 4-36
Extended Subtract (SUBX --)) - - - - - 4-39
- F**
- Falling-edge Detection Coil (OFFP-COIL) - - - - - 4-31
Falling-edge NC Contact (OFFP-NCC) - - - - - 4-15
Falling-edge NO Contact (OFFP-NOC) - - - - - 4-13
Falling-edge Pulses (OFF-PLS) - - - - - 4-27
First-in First-out (FINFOUT) - - - - - 4-207
First-order Lag (LAG) - - - - - 4-142
Flash Operation (FLASH-OP) - - - - - 4-233
FOR Construct (FOR,END_FOR) - - - - - 4-86
Force OFF - - - - - 5-14
Force ON - - - - - 5-14
forcing coils ON or OFF - - - - - 5-14
 from a ladder program - - - - - 5-14
 from the Force Coil List Pane - - - - - 5-14
 searching for forced coils in the Force Coil
 List Pane - - - - - 5-14
function external registers - - - - - 3-5
Function Generator (FGN) - - - - - 4-147
function input registers - - - - - 3-4
function internal registers - - - - - 3-5
function output registers - - - - - 3-4
- G**
- G registers - - - - - 3-2
global registers - - - - - 3-2
going online - - - - - 2-6
grandchild drawings - - - - - 1-7
Greater Than (>) - - - - - 4-72
Greater Than or Equal (\geq) - - - - - 4-71
- H**
- high-speed drawing operation mode settings - - - - - 1-11
- I**
- IF Construct (IF, END_IF) - - - - - 4-88
IF-ELSE Construct (IF, ELSE, END_IF) - - - - - 4-90
Import (IMPORT/IMPORTL/IMPORTLE) - - - - - 4-240
Inclusive AND (AND) - - - - - 4-64
Inclusive OR (OR) - - - - - 4-65
Increment (INC) - - - - - 4-46
index registers (i, j) - - - - - 3-12
individual transfer - - - - - 2-12
input registers - - - - - 3-3
inserting instructions - - - - - 2-7
inserting rungs - - - - - 2-7
installing MPE720 version 7 - - - - - 2-3
- integer - - - - - 3-8
Integer Remainder (MOD) - - - - - 4-43
Inverse Function Generator (IFGN) - - - - - 4-151
Invert Sign (INV) - - - - - 4-54
- L**
- ladder drawings - - - - - 1-7
 execution timing - - - - - 1-3
ladder language instructions - - - - - 4-6
Ladder Pane - - - - - 1-6
ladder program - - - - - 1-2
Ladder Program Editor - - - - - 1-6
ladder program runtime monitoring - - - - - 5-4
Less Than (<) - - - - - 4-67
Less Than or Equal (\leq) - - - - - 4-68
Linear Accelerator/Decelerator 1 (LAU) - - - - - 4-155
Linear Accelerator/Decelerator 2 (SLAU) - - - - - 4-161
local registers - - - - - 3-4
Logic Operation Instructions - - - - - 4-7
Logic Operations and Comparison Instructions - - - - - 4-64
logic operators - - - - - C-2, C-5
- M**
- motion programs - - - - - 5-26
Move Bit (MOVB) - - - - - 4-106
Move Table Block (TBLMV) - - - - - 4-190
Move Word (MOVW) - - - - - 4-108
Multiply (MUL (x)) - - - - - 4-41
- N**
- Natural Logarithm (LN) - - - - - 4-102
NC Contact (NCC) - - - - - 4-14
NO Contact (NOC) - - - - - 4-11
Not Equal (\neq) - - - - - 4-70
Numeric Operation Instructions - - - - - 4-7, 4-34
- O**
- One's Complement (COM) - - - - - 4-55
online security setting - - - - - 5-24
operands - - - - - C-3
operation error drawings - - - - - 1-7
operators - - - - - C-2
output registers - - - - - 3-3
- P**
- parent drawings - - - - - 1-7
Parity Conversion (PARITY) - - - - - 4-59
PD Control (PD) - - - - - 4-133
Phase Lead Lag (LLAG) - - - - - 4-144
PI Control (PI) - - - - - 4-128
PID Control (PID) - - - - - 4-137
preparation for devices to be connected - - - - - 2-3
privilege levels - - - - - 5-24

Program Control Instructions - - - - - 4-7
 program password setting - - - - - 5-24
 project password setting - - - - - 5-24
 Pulse Width Modulation (PWM) - - - - - 4-170

Q

quadruple-length integer - - - - - 3-8

R

Range Check (RCHK) - - - - - 4-73
 Read Data Trace (DTRC-RD/DTRC-RDE) - - - - - 4-212
 Read Motion Register (MOTREG-R) - - - - - 4-238
 Read Queue Table (QTBLR and QTBLRI) - - - - - 4-194
 Read SERVOPACK Parameter (MLNK-SVR)- - - - - 4-228
 Read Table Block (TBLBR)- - - - - 4-173
 real number - - - - - 3-8
 Real Remainder (REM) - - - - - 4-45
 real-time trace- - - - - 5-25
 Receive Message (MSG-RCV)- - - - - 4-220
 Receive Message Extended (MSG-RCVE) - - - - - 4-221
 Register Lists - - - - - 2-13, 5-18
 register map - - - - - 5-18
 register types - - - - - 3-8
 Relay Circuit Instructions- - - - - 4-6, 4-11
 Replace Dialog Box- - - - - 5-6
 Replace in the Project Dialog Box - - - - - 5-8
 replacing in project files - - - - - 5-8
 replacing text in programs - - - - - 5-6
 reserved words - - - - - C-2
 Reset Coil (R-COIL)- - - - - 4-33
 Reverse Coil (REV-COIL)- - - - - 4-30
 Rising-edge Detection Coil (ONP-COIL) - - - - - 4-31
 Rising-edge NC Contact (ONP-NCC)- - - - - 4-14
 Rising-edge NO Contact (ONP-NOC) - - - - - 4-12
 Rising-edge Pulses (ON-PLS) - - - - - 4-25

S

saving the ladder program to flash memory - - - - - 2-16
 scheduling the execution of high-speed and
 low-speed scan process drawings - - - - - 1-10
 Search Dialog Box - - - - - 5-5
 Search for Table Column (TBLSRC)- - - - - 4-184
 Search for Table Row (TBLRSL)- - - - - 4-181
 Search in Project Dialog Box - - - - - 5-7
 searching and replacing - - - - - 5-5
 searching and replacing in programs - - - - - 5-5
 searching in programs - - - - - 5-5
 searching in project files - - - - - 5-7
 security features - - - - - 5-24
 security key - - - - - 5-24
 self configuration - - - - - 2-5
 Send Message (MSG-SND) - - - - - 4-216

Send Message Extended (MSG-SNDE) - - - - - 4-218
 Set Coil (S-COIL) - - - - - 4-32
 Setting the High-speed and Low-speed Times - - - - - 1-10
 Sine (SIN) - - - - - 4-94
 Sort (SORT) - - - - - 4-117
 Spend Time (SPEND) - - - - - 4-52
 Square Root (SQRT) - - - - - 4-93
 Standard System Function Instructions - - - - - 4-9
 Store (STORE) - - - - - 4-34
 structure of register addresses - - - - - 3-8
 substitution operator- - - - - C-2, C-6
 Subtract (SUB (-)) - - - - - 4-38
 Subtract Time (TMSUB) - - - - - 4-50
 switching the register map display - - - - - 5-19
 system configuration example - - - - - 2-3
 System Function Instructions - - - - - 4-204
 system registers - - - - - 3-2
 System Service Registers - - - - - A-2

T

Tab Page to Edit Ladder Program - - - - - 1-6
 table data - - - - - 1-18
 Table Initialization (SETW) - - - - - 4-111
 Table Manipulation Instructions - - - - - 4-8, 4-173
 Tangent (TAN)- - - - - 4-97
 Trace (TRACE) - - - - - 4-210
 trace manager - - - - - 5-25
 tracing - - - - - 5-25
 Tuning Panel - - - - - 5-21

U

Upper/Lower Limit (LIMIT)- - - - - 4-126
 user administration - - - - - 5-24
 user functions - - - - - 1-13

V

Variable Pane - - - - - 1-6
 variables- - - - - C-3
 viewing called programs- - - - - 5-17
 Visual monitor - - - - - 5-21

W

watching - - - - - 5-23
 WHILE Construct (WHILE, END_WHILE) - - - - - 4-84
 Word-to-byte Compression (BPRESS)- - - - - 4-114
 Write Motion Register (MOTREG-W) - - - - - 4-236
 Write Queue Table (QTBLW and QTBLWI) - - - - - 4-198
 Write SERVOPACK Parameter (MLNK-SVW) - - - - - 4-223
 Write Table Block (TBLBW) - - - - - 4-177
 writing the ladder programs - - - - - 2-11

X

XY trace - - - - - 5-25

Revision History

The date of publication, revision number, and web revision number are given at the bottom right of the back cover. Refer to the following example.

MANUAL NO. SIEP C880725 13A <0>-1
 Published in Japan June 2012

┌─── WEB revision number
 │
 └─── Revision number
 └─── Date of publication

Date of Publication	Rev. No.	Web Rev. No.	Section	Revised Contents
September 2019	<7>	0	Chapter 4	Partly revised.
			Back cover	Revision: Format
May 2019	<6>	0	All chapters	Partly revised.
			Chapter 4	Addition: Storage operation instructions and string operation instructions
			Back cover	Revision: Address
October 2017	<5>	0	Chapter 3	Addition: Usable range of local registers
				Addition: Setting for D Registers
			4.8	Revision: Expression of dead zone set value for Dead Zone A and Dead Zone B
			4.10	Revision: Trace (TRACE), Write SERVOPACK Parameter (MLNK-SVW), Read SERVOPACK Parameter (MLNK-SVR), Export (EXPORT/EXPORTL/EXPORTLE)
			C.1	Addition: Information on extended addition and extended subtraction
			C.2	Addition: Important information on arithmetic operators
July 2017	<4>	1	1.3	Revision: Maximum number of drawings for DWGH and DWGL
			4.10	Deletion: Data number listed in I/O item of IMPORTLE table
			Back cover	Revision: Address
February 2017		0	–	Same changes as for SIEP C880725 13D<3>-1 for the Web
			4.10	Addition: Information on reading data traces (DTRC-RDE), importing (IMPORTLE), and exporting (EXPORTLE)
			Back cover	Revision: Address
September 2016	<3>	1	4.8	Addition: Information on the scan time set value
December 2015		0	4.2	Revision: Information on OFF-Delay Timer (TOFF (1 ms))
			4.8	Revision: Specifications for P gain, I gain, and D gain in the parameter tables for the real-number PI, PD, and PID instructions
				Revision: Programming examples for PI, PD, and PID control
			4.9	Revision: Table data for Write Table Block (TBLBW)
			4.10	Addition: Read SERVOPACK Parameter (MLNK-SVR) Addition: Flash Operation (FLASH-OP)
Back cover	Revision: Address			
June 2015	<2>	2	4.2, 4.5	Addition: Precaution for user functions
			Front cover, back cover	Revision: Format
February 2015		1	4.2	Addition: Timing charts and notes on combining instructions
			Back cover	Revision: Address
September 2014		0	All chapters	Addition: Information related to the MP3300.
			Preface	Revision: PL contents.
			4.5	Revision: RSSEL parameter for JNS and OUTS instructions Rack numbers changed from "1 to 4" to "1 to 7" and slot numbers changed from "0 to 8" to "0 to 9."
			4.10	Addition: Information related to the IMPORTL and EXPORTL instructions.
			Back cover	Revision: Address
September 2012	<1>	0	All chapters	Fully revised.
			4.10	Addition: Write SERVOPACK Parameter (MLNK-SVW)
			Back cover	Revision: Address
June 2012	<0>	1	Appendix C.1	Addition: Control instructions as operators.
March 2012	–	–	–	First edition

Machine Controller MP3000 Series

Ladder Program

PROGRAMMING MANUAL

IRUMA BUSINESS CENTER (SOLUTION CENTER)

480, Kamifujisawa, Iruma, Saitama, 358-8555, Japan
Phone: +81-4-2962-5151 Fax: +81-4-2962-6138
<http://www.yaskawa.co.jp>

YASKAWA AMERICA, INC.

2121, Norman Drive South, Waukegan, IL 60085, U.S.A.
Phone: +1-800-YASKAWA (927-5292) or +1-847-887-7000 Fax: +1-847-887-7310
<http://www.yaskawa.com>

YASKAWA ELÉTRICO DO BRASIL LTDA.

777, Avenida Piraporinha, Diadema, São Paulo, 09950-000, Brasil
Phone: +55-11-3585-1100 Fax: +55-11-3585-1187
<http://www.yaskawa.com.br>

YASKAWA EUROPE GmbH

Hauptstraße 185, 65760 Eschborn, Germany
Phone: +49-6196-569-300 Fax: +49-6196-569-398
<http://www.yaskawa.eu.com> E-mail: info@yaskawa.eu.com

YASKAWA ELECTRIC KOREA CORPORATION

35F, Three IFC, 10 Gukjegeumyung-ro, Yeongdeungpo-gu, Seoul, 07326, Korea
Phone: +82-2-784-7844 Fax: +82-2-784-8495
<http://www.yaskawa.co.kr>

YASKAWA ASIA PACIFIC PTE. LTD.

30A, Kallang Place, #06-01, 339213, Singapore
Phone: +65-6282-3003 Fax: +65-6289-3003
<http://www.yaskawa.com.sg>

YASKAWA ELECTRIC (THAILAND) CO., LTD.

59, 1st-5th Floor, Flourish Building, Soi Ratchadapisek 18, Ratchadapisek Road, Huaykwang, Bangkok, 10310, Thailand
Phone: +66-2-017-0099 Fax: +66-2-017-0799
<http://www.yaskawa.co.th>

YASKAWA ELECTRIC (CHINA) CO., LTD.

22F, Link Square 1, No.222, Hubin Road, Shanghai, 200021, China
Phone: +86-21-5385-2200 Fax: +86-21-5385-3299
<http://www.yaskawa.com.cn>

YASKAWA ELECTRIC (CHINA) CO., LTD. BEIJING OFFICE

Room 1011, Tower W3 Oriental Plaza, No.1, East Chang An Ave.,
Dong Cheng District, Beijing, 100738, China
Phone: +86-10-8518-4086 Fax: +86-10-8518-4082

YASKAWA ELECTRIC TAIWAN CORPORATION

12F, No. 207, Sec. 3, Beishin Rd., Shindian Dist., New Taipei City 23143, Taiwan
Phone: +886-2-8913-1333 Fax: +886-2-8913-1513 or +886-2-8913-1519
<http://www.yaskawa.com.tw>

YASKAWA

YASKAWA ELECTRIC CORPORATION

In the event that the end user of this product is to be the military and said product is to be employed in any weapons systems or the manufacture thereof, the export will fall under the relevant regulations as stipulated in the Foreign Exchange and Foreign Trade Regulations. Therefore, be sure to follow all procedures and submit all relevant documentation according to any and all rules, regulations and laws that may apply.

Specifications are subject to change without notice for ongoing product modifications and improvements.

© 2012 YASKAWA ELECTRIC CORPORATION

MANUAL NO. SIEP C880725 13H <7>-0

Published in Japan September 2019

18-10-15

Original instructions