Cat. No. O013-E1-03

FQM1 Series
FQM1-CM002
FQM1-MMP22
FQM1-MMA22

Flexible Motion Controller

INSTRUCTIONS REFERENCE MANUAL

OMRON

FQM1 Series

FQM1-CM002
FQM1-MMP22
FQM1-MMA22

Flexible Motion Controller

Instructions Reference Manual

Revised April 2008

Notice:

OMRON products are manufactured for use according to proper procedures by a qualified operator and only for the purposes described in this manual.

The following conventions are used to indicate and classify precautions in this manual. Always heed the information provided with them. Failure to heed precautions can result in injury to people or damage to property.

⚠DANGER Indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury. Additionally, there may be severe property damage.

⚠WARNING Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury. Additionally, there may be severe property damage.

⚠Caution Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury, or property damage.

OMRON Product References

All OMRON products are capitalized in this manual. The word “Unit” is also capitalized when it refers to an OMRON product, regardless of whether or not it appears in the proper name of the product.

The abbreviation “Ch,” which appears in some displays and on some OMRON products, often means “word” and is abbreviated “Wd” in documentation in this sense.

The abbreviation “CM” means Coordinator Module and the abbreviation “MM” means Motion Control Module.

Visual Aids

The following headings appear in the left column of the manual to help you locate different types of information.

Note Indicates information of particular interest for efficient and convenient operation of the product.

1,2,3... 1. Indicates lists of one sort or another, such as procedures, checklists, etc.

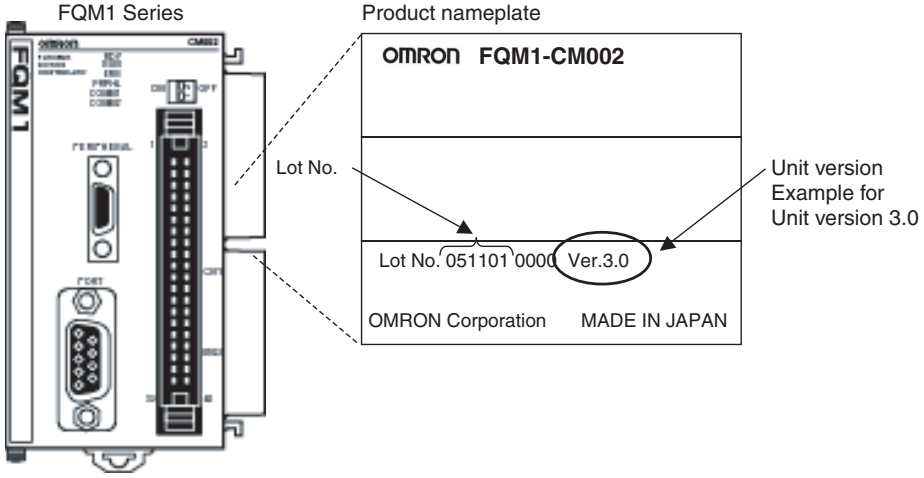
Unit Versions of FQM1 Series Flexible Motion Controller

Unit Versions

The FQM1 Series Controllers have “unit versions”, which are used to manage the differences in functionality associated with upgrades to the Coordinator Modules and Motion Control Modules.

Notation of Unit Versions on Products

The unit version is listed just to the right of the lot number on the nameplate of the Module, as shown below.



Unit Versions and Model Numbers

Name	Unit Ver. 2.0	Unit Ver. 3.0
Coordinator Module	FQM1-CM001	FQM1-CM002
Motion Control Module	FQM1-MMA21 FQM1-MMP21	FQM1-MMA22 FQM1-MMP22

Note The Ver. 2.0 Modules (FQM1-CM001, FQM1-MMA21, and FQM1-MMP21) can be used together with the Ver. 3.0 Modules (FQM1-CM002, FQM1-MMA22, and FQM1-MMP22).

Version Upgrade Guide

■ Functional Improvements from Version 3.0 to Version 3.1

Previous version (unit version 3.0)	Unit version 3.1 or later
Not UL listed	UL listed Note: For an FQM1-series Controller to conform to the UL listing, the system must be configured with an XW2B-80J7-1A Relay Unit and XW2Z-□□□J-A□□ Connecting Cable.

■ Functional Improvements from Version 3.1 to Version 3.2

Previous version (unit version 3.1)	Unit version 3.2 or later
Not in previous version	When PULS(886) is used in electronic cam mode (ring), the pulse output can be set to pass through 0 in the CW direction or CCW direction.
When PULS(886) is used in electronic cam mode (linear or ring), the user set the present operation's reference position and pulse output frequency in the instruction's operands.	When PULS(886) is used in electronic cam mode (linear or ring), a new option can be selected to automatically calculate the pulse output frequency based on the previous reference value and the present operation's reference value.
Not in previous version	Two cyclic refreshing areas (up to 25 words each for output and input) can be added. These areas are primarily used as interface areas between the Coordinator Module and the base FB in the Motion Control Module. When the base FB is not being used, these areas can be used as work words.
Mounting CJ-series Units • Basic I/O Units (except the CJ1W-INT01 and CJ1W-IDP01) • CPU Bus Units: CJ1W-SPU01 and CJ1W-NCF71 • Special I/O Units: CJ1W-SRM21 • Communications Units: CJ1W-DRM21	The following Units can be mounted, in addition to the Units listed on the left. CPU Bus Units: CJ1W-ADG41 Special I/O Units: CJ1W-NC113/213/413/133/ 233/ 433, CJ1W-V600C11/V600C12 Note: The FQM1 Controllers do not support the IORD(222) and IOWR(223) instructions.
Not in previous version	When the counter reset method is set to Phase-Z signal + software reset in the system settings, an interrupt task can be started when the counter is reset.
When the 20-MHz clock is specified in the system settings for the pulse output function, the output frequency range is 400 Hz to 1 MHz.	When the 20-MHz clock is specified in the system settings, a new option can be selected to set an output frequency range of 1 Hz to 1 MHz.
When the high-speed analog sampling function is used with counter 1 as the sampling timing counter, the multiplier is always 1x, regardless of the counter 1 multiplier setting (1x, 2x, or 4x).	The sampling timing counter uses the same 1x, 2x, or 4x multiplier setting that is set for counter 1.
The VIRTUAL AXIS (AXIS (981)) instruction's calculation cycle can be set to 0.5 ms, 1 ms, or 2 ms.	The calculation cycle settings have been expanded. The cycle can be set to 0.5 ms, 1 ms, 2 ms, 3 ms, or 4 ms. The following conditions were removed from the conditions detected as errors when the instruction is executed. • Target position (travel amount in relative mode) = 0 • Target position (target position in absolute mode) = Present position • Target frequency < Deceleration rate

■ **Functional Improvements from Version 3.2 to Version 3.3**

Previous version (unit version 3.2)	Unit version 3.3 or later
OMNUC W-series Absolute Encoders can be used.	Absolute Encoders of OMNUC G-series Servomotors can be now be used (in addition to the Absolute Encoders of W-series Servomotors).
CJ-series Units can be mounted.	In addition to the Units that could previously be mounted, the following Special I/O Units can now be mounted. • Analog Output Units: CJ1W-DA08V, CJ1W-DA08C, CJ1W-DA041, and CJ1W-DA021 • Analog Input Units: CJ1W-AD081-V1 and CJ1W-041-V1 • Analog I/O Unit: CJ1W-MAD42
The offset and gain of an analog output can be adjusted separately.	In addition to the previous functions, the default adjustment data can now be registered as the offset value for the analog output offset/gain adjustment function when adjusting the gain. This feature is useful for connecting to a Servo Driver, adjusting the offset using the Servo Driver, and then adjusting only the gain.

TABLE OF CONTENTS

PRECAUTIONS xvii

1 Intended Audience xviii
2 General Precautions xviii
3 Safety Precautions xviii
4 Conformance to EC Directives xxiii
5 Data Backup xxvi

SECTION 1
Introduction 1

1-1 General Instruction Characteristics 2
1-2 Instruction Execution Checks 8

SECTION 2
Summary of Instructions 11

2-1 Instruction Classifications by Function 12
2-2 Instruction Functions 18
2-3 Alphabetical List of Instructions by Mnemonic 69
2-4 List of Instructions by Function Code 78

SECTION 3
Instructions 87

3-1 Notation and Layout of Instruction Descriptions 92
3-2 Sequence Input Instructions 95
3-3 Sequence Output Instructions 117
3-4 Sequence Control Instructions 134
3-5 Timer and Counter Instructions 152
3-6 Comparison Instructions 167
3-7 Data Movement Instructions 199
3-8 Data Shift Instructions 225
3-9 Increment/Decrement Instructions 265
3-10 Symbol Math Instructions 281
3-11 Conversion Instructions 331
3-12 Logic Instructions 351
3-13 Special Math Instructions 368
3-14 Floating-point Math Instructions 380
3-15 Double-precision Floating-point Instructions 425
3-16 Table Data Processing Instructions 467
3-17 Data Control Instructions 475
3-18 Subroutines 491
3-19 Interrupt Control Instructions 508
3-20 High-speed Counter/Pulse Output Instructions 521
3-21 Step Instructions 562

TABLE OF CONTENTS

3-22 I/O Refresh Instruction 580
3-23 Serial Communications Instructions 582
3-24 Debugging Instructions 596
3-25 Failure Diagnosis Instructions 600
3-26 Other Instructions 606
3-27 Block Programming Instructions 608
3-28 Function Block Instructions 618

SECTION 4
Instruction Execution Times and Number of Steps 621
4-1 FQM1 Instruction Execution Times and Number of Steps 622

Index 639

Revision History 645

About this Manual:

This manual describes the ladder diagram programming instructions of the Coordinator Module and Motion Control Modules of the FQM1-series Flexible Motion Controllers.

Please read this manual and all related manuals listed in the table on the next page and be sure you understand information provided before attempting to program or use FQM1-series Flexible Motion Controllers in a control system.

Name	Cat. No.	Contents
FQM1 Series FQM1-CM002, FQM1-MMP22, FQM1-MMA22 Flexible Motion Controllers Operation Manual	O012	This manual provides an overview of and describes the following information for the FQM1-series Flexible Motion Controllers: features, system configuration, system design, installation, wiring, maintenance, I/O memory allocation, troubleshooting, etc.
FQM1 Series FQM1-CM002, FQM1-MMP22, FQM1-MMA22 Flexible Motion Controllers Instructions Reference Manual (this manual)	O013	Describes the ladder diagram programming instructions supported by FQM1-series Flexible Motion Controllers. Use this manual together with the *Operation Manual* (Cat. No. O012).
SYSMAC WS02-CXPC1-E-V7 CX-Programmer Operation Manual Version 7.x	W446	Provides information on how to use the CX-Programmer (except for function block functionality).
SYSMAC WS02-CXPC1-E-V7 CX-Programmer Operation Manual Version 7.x Function Blocks	W447	Provides specifications and operating procedures for function blocks.
SYSMAC CXONE-AL□□C-E CX-One FA Integrated Tool Package Setup Manual	W445	Provides an overview of the CX-One FA Integrated Tool and installation procedures.

Section 1 provides information on general instruction characteristics as well as the errors that can occur during instruction execution.

Section 2 provides summaries of instructions used with the FQM1.

Section 3 describes each of the instructions that can be used in programming the FQM1.

Section 4 provides instruction execution times and the number of steps for each FQM1 instruction.

Read and Understand this Manual

Please read and understand this manual before using the product. Please consult your OMRON representative if you have any questions or comments.

Warranty and Limitations of Liability

WARRANTY

OMRON's exclusive warranty is that the products are free from defects in materials and workmanship for a period of one year (or other period if specified) from date of sale by OMRON.

OMRON MAKES NO WARRANTY OR REPRESENTATION, EXPRESS OR IMPLIED, REGARDING NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR PARTICULAR PURPOSE OF THE PRODUCTS. ANY BUYER OR USER ACKNOWLEDGES THAT THE BUYER OR USER ALONE HAS DETERMINED THAT THE PRODUCTS WILL SUITABLY MEET THE REQUIREMENTS OF THEIR INTENDED USE. OMRON DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED.

LIMITATIONS OF LIABILITY

OMRON SHALL NOT BE RESPONSIBLE FOR SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS OR COMMERCIAL LOSS IN ANY WAY CONNECTED WITH THE PRODUCTS, WHETHER SUCH CLAIM IS BASED ON CONTRACT, WARRANTY, NEGLIGENCE, OR STRICT LIABILITY.

In no event shall the responsibility of OMRON for any act exceed the individual price of the product on which liability is asserted.

IN NO EVENT SHALL OMRON BE RESPONSIBLE FOR WARRANTY, REPAIR, OR OTHER CLAIMS REGARDING THE PRODUCTS UNLESS OMRON'S ANALYSIS CONFIRMS THAT THE PRODUCTS WERE PROPERLY HANDLED, STORED, INSTALLED, AND MAINTAINED AND NOT SUBJECT TO CONTAMINATION, ABUSE, MISUSE, OR INAPPROPRIATE MODIFICATION OR REPAIR.

Application Considerations

SUITABILITY FOR USE

OMRON shall not be responsible for conformity with any standards, codes, or regulations that apply to the combination of products in the customer's application or use of the products.

At the customer's request, OMRON will provide applicable third party certification documents identifying ratings and limitations of use that apply to the products. This information by itself is not sufficient for a complete determination of the suitability of the products in combination with the end product, machine, system, or other application or use.

The following are some examples of applications for which particular attention must be given. This is not intended to be an exhaustive list of all possible uses of the products, nor is it intended to imply that the uses listed may be suitable for the products:

- Outdoor use, uses involving potential chemical contamination or electrical interference, or conditions or uses not described in this manual.
- Nuclear energy control systems, combustion systems, railroad systems, aviation systems, medical equipment, amusement machines, vehicles, safety equipment, and installations subject to separate industry or government regulations.
- Systems, machines, and equipment that could present a risk to life or property.

Please know and observe all prohibitions of use applicable to the products.

NEVER USE THE PRODUCTS FOR AN APPLICATION INVOLVING SERIOUS RISK TO LIFE OR PROPERTY WITHOUT ENSURING THAT THE SYSTEM AS A WHOLE HAS BEEN DESIGNED TO ADDRESS THE RISKS, AND THAT THE OMRON PRODUCTS ARE PROPERLY RATED AND INSTALLED FOR THE INTENDED USE WITHIN THE OVERALL EQUIPMENT OR SYSTEM.

PROGRAMMABLE PRODUCTS

OMRON shall not be responsible for the user's programming of a programmable product, or any consequence thereof.

Disclaimers

CHANGE IN SPECIFICATIONS

Product specifications and accessories may be changed at any time based on improvements and other reasons.

It is our practice to change model numbers when published ratings or features are changed, or when significant construction changes are made. However, some specifications of the products may be changed without any notice. When in doubt, special model numbers may be assigned to fix or establish key specifications for your application on your request. Please consult with your OMRON representative at any time to confirm actual specifications of purchased products.

DIMENSIONS AND WEIGHTS

Dimensions and weights are nominal and are not to be used for manufacturing purposes, even when tolerances are shown.

PERFORMANCE DATA

Performance data given in this manual is provided as a guide for the user in determining suitability and does not constitute a warranty. It may represent the result of OMRON's test conditions, and the users must correlate it to actual application requirements. Actual performance is subject to the OMRON Warranty and Limitations of Liability.

ERRORS AND OMISSIONS

The information in this manual has been carefully checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical, or proofreading errors, or omissions.

PRECAUTIONS

This section provides general precautions for using the FQM1-series Flexible Motion Controllers and related devices.

The information contained in this section is important for the safe and reliable application of the FQM1-series Flexible Motion Controller. You must read this section and understand the information contained before attempting to set up or operate a control system using the FQM1-series Flexible Motion Controller.

1 Intended Audience xviii
2 General Precautions xviii
3 Safety Precautions xviii
3-1 Operating Environment Precautions xix
3-2 Application Precautions xx
4 Conformance to EC Directives xxiii
4-1 Applicable Directives xxiii
4-2 Concepts xxiii
4-3 Conformance to EC Directives xxiii
4-4 EMC Directive Conformance Conditions xxiii
4-5 Relay Output Noise Reduction Methods xxiv
5 Data Backup xxvi

1 Intended Audience

This manual is intended for the following personnel, who must also have knowledge of electrical systems (an electrical engineer or the equivalent).

- Personnel in charge of installing FA systems.
- Personnel in charge of designing FA systems.
- Personnel in charge of managing FA systems and facilities.

2 General Precautions

The user must operate the product according to the performance specifications described in the operation manuals.

Before using the product under conditions which are not described in the manual or applying the product to nuclear control systems, railroad systems, aviation systems, vehicles, combustion systems, medical equipment, amusement machines, safety equipment, petrochemical plants, and other systems, machines, and equipment that may have a serious influence on lives and property if used improperly, consult your OMRON representative.

Make sure that the ratings and performance characteristics of the product are sufficient for the systems, machines, and equipment, and be sure to provide the systems, machines, and equipment with double safety mechanisms.

⚠ **WARNING** It is extremely important that the FQM1 be used for the specified purpose and under the specified conditions, especially in applications that can directly or indirectly affect human life. You must consult with your OMRON representative before applying an FQM1 System to the above-mentioned applications.

3 Safety Precautions

⚠ **WARNING** Do not attempt to take any Modules apart while the power is being supplied. Doing so may result in electric shock.

⚠ **WARNING** Do not touch any of the terminals or terminal blocks while the power is being supplied. Doing so may result in electric shock.

⚠ **WARNING** Do not attempt to disassemble, repair, or modify any Modules. Any attempt to do so may result in malfunction, fire, or electric shock.

⚠ **WARNING** Provide safety measures in external circuits, i.e., not in the Flexible Motion Controller (referred to as the “FQM1”), to ensure safety in the system if an abnormality occurs due to malfunction of the FQM1 or another external factor affecting the FQM1 operation. Not doing so may result in serious accidents.

- Emergency stop circuits, interlock circuits, limit circuits, and similar safety measures must be provided in external control circuits.
- The FQM1 will turn OFF all outputs when its self-diagnosis function detects any error or when a severe failure alarm (FALS) instruction is executed. As a countermeasure for such errors, external safety measures must be provided to ensure safety in the system.
- The FQM1 outputs may remain ON or OFF due to destruction of the output transistors. As a countermeasure for such problems, external safety measures must be provided to ensure safety in the system.

- When the 24-VDC output (service power supply to the FQM1) is overloaded or short-circuited, the voltage may drop and result in the outputs being turned OFF. As a countermeasure for such problems, external safety measures must be provided to ensure safety in the system.

⚠ **WARNING** Fail-safe measures must be taken by the customer to ensure safety in the event of incorrect, missing, or abnormal signals caused by broken signal lines, momentary power interruptions, or other causes. Not doing so may result in serious accidents.

⚠ **Caution** Execute online edit only after confirming that no adverse effects will be caused by extending the cycle time. Otherwise, the input signals may not be readable.

⚠ **Caution** User programs and parameters written to the Coordinator Module or Motion Control Module will be automatically backed up in the FQM1 flash memory (flash memory function). The contents of I/O memory (including the DM Area), however, are not written to flash memory. Part of the DM Area used as a holding area when recovering from a power interruption is backed up using a super capacitor, but correct values will not be maintained if an error occurs that prevents memory backup. As a countermeasure for such problems, take appropriate measures in the program using the Memory Not Held Flag (A316.14) when externally outputting the contents of the DM Area.

⚠ **Caution** Confirm safety at the destination Module before transferring a program to another Module or editing the I/O area. Doing either of these without confirming safety may result in injury.

⚠ **Caution** Tighten the screws on the terminal block of the AC Power Supply Unit to the torque specified in the operation manual. The loose screws may result in burning or malfunction.

⚠ **Caution** Do not touch the Power Supply Unit while the power is ON, and immediately after turning OFF the power. Touching hot surfaces may result in burning.

⚠ **Caution** Pay careful attention to the polarities (+/-) when wiring the DC power supply. A wrong connection may cause malfunction of the system.

3-1 Operating Environment Precautions

⚠ **Caution** Do not operate the control system in the following places:

- Locations subject to direct sunlight
- Locations subject to temperatures or humidity outside the range specified in the specifications
- Locations subject to condensation as the result of severe changes in temperature
- Locations subject to corrosive or flammable gases
- Locations subject to dust (especially iron dust) or salts
- Locations subject to exposure to water, oil, or chemicals
- Locations subject to shock or vibration

⚠ **Caution** Take appropriate and sufficient countermeasures when installing systems in the following locations:

- Locations subject to static electricity or other forms of noise
- Locations subject to strong electromagnetic fields
- Locations subject to possible exposure to radioactivity
- Locations close to power supplies

⚠ **Caution** The operating environment of the FQM1 System can have a large effect on the longevity and reliability of the system. Improper operating environments can lead to malfunction, failure, and other unforeseeable problems with the FQM1 System. Make sure that the operating environment is within the specified conditions at installation and remains within the specified conditions during the life of the system.

3-2 Application Precautions

⚠ **WARNING** Always heed these precautions. Failure to abide by the following precautions could lead to serious or possibly fatal injury.

- Always connect to a ground of 100 Ω or less when installing the FQM1. Not doing so may result in electric shock.
- Always connect to a ground of 100 Ω or less when short-circuiting the functional ground and line ground terminals of the Power Supply Unit, in particular.
- Always turn OFF the power supply to the FQM1 before attempting any of the following. Not turning OFF the power supply may result in malfunction or electric shock.
  - Mounting or dismounting Power Supply Unit, Coordinator Module, Motion Control Module, I/O Control Module, CJ-series Units, and End Module
  - Assembling the Modules
  - Setting DIP switches
  - Connecting or wiring the cables
  - Connecting or disconnecting the connectors

⚠ **Caution** Failure to abide by the following precautions could lead to faulty operation of the FQM1 or the system, or could damage the FQM1. Always heed these precautions.

- Always use the CX-Programmer (Programming Device for Windows) to create new cyclic tasks and interrupt tasks.
- The user program, parameter area data, and part of the DM Area in the Coordinator Module and Motion Control Modules is backed up in the built-in flash memory. Do not turn OFF the power supply to the FQM1 while the user program or parameter area data is being transferred. The data will not be backed up if the power is turned OFF.
- The FQM1 will start operating in RUN mode when the power is turned ON with the default settings (i.e., if the operating mode at power ON (startup mode) setting in the System Setup is disabled).
- Configure the external circuits so that the control power supply turns ON after the power supply to the FQM1 turns ON. If the power is turned ON in the opposite order, the built-in outputs and other outputs may momentarily malfunction and the control outputs may temporarily not operate correctly.

- Outputs may remain ON due to a malfunction in the built-in transistor outputs or other internal circuits. As a countermeasure for such problems, external safety measures must be provided to ensure the safety of the system.
- Part of the DM Area (data memory) in the Motion Control Module is held using the super capacitor. Corrupted memory may prevent the correct values from being saved, however. Take appropriate measures in the ladder program whenever the Memory Not Held Flag (A316.14) turns ON, such as resetting the data in the DM Area.
- Part of the DM Area in the Coordinator Module is backed up in the built-in flash memory when transferring data from the CX-Programmer. Do not turn OFF the power to the FQM1 while data is being transferred. The data will not be backed up if the power is turned OFF.
- Confirm that no adverse effect will occur in the system before attempting any of the following. Not doing so may result in an unexpected operation.
  - Changing the operating mode of the FQM1 (including setting the operating mode at startup)
  - Force-setting/force-resetting any bit in memory
  - Changing the present value of any word or any set value in memory
- Install external breakers and take other safety measures against short-circuiting in external wiring. Insufficient safety measures against short-circuiting may result in burning.
- Be sure that all the terminal screws and cable connector screws are tightened to the torque specified in the relevant manuals. Incorrect tightening torque may result in malfunction.
- Mount the Modules only after checking the connectors and terminal blocks completely.
- Before touching the Module, be sure to first touch a grounded metallic object in order to discharge any static built-up. Not doing so may result in malfunction or damage.
- Be sure that the terminal blocks, connectors, and other items with locking devices are properly locked into place. Improper locking may result in malfunction.
- Wire correctly according to the specified procedures.
- Always use the power supply voltage specified in the operation manuals. An incorrect voltage may result in malfunction or burning.
- Take appropriate measures to ensure that the specified power with the rated voltage and frequency is supplied. Be particularly careful in places where the power supply is unstable. An incorrect power supply may result in malfunction.
- Leave the dust protective label attached to the Module when wiring. Removing the label may result in malfunction.
- Remove the dust protective label after the completion of wiring to ensure proper heat dissipation. Leaving the label attached may result in malfunction.
- Use crimp terminals for wiring. Do not connect bare stranded wires directly to terminals. Connection of bare stranded wires may result in burning.
- Do not apply voltages to the built-in inputs in excess of the rated input voltage. Excess voltages may result in burning.

- Do not apply voltages or connect loads to the built-in outputs in excess of the maximum switching capacity. Excess voltage or loads may result in burning.
- Disconnect the functional ground terminal when performing withstand voltage tests. Not disconnecting the functional ground terminal may result in burning.
- Wire correctly and double-check all the wiring or the setting switches before turning ON the power supply. Incorrect wiring may result in burning.
- Check that the DIP switches and data memory (DM) are properly set before starting operation.
- Check the user program for proper execution before actually running it on the Module. Not checking the program may result in an unexpected operation.
- Resume operation only after transferring to the new Module the contents of the DM Areas, programs, parameters, and data required for resuming operation. Not doing so may result in an unexpected operation.
- Do not pull on the cables or bend the cables beyond their natural limit. Doing either of these may break the cables.
- Do not place objects on top of the cables. Doing so may break the cables.
- Use the dedicated connecting cables specified in operation manuals to connect the Modules. Using commercially available RS-232C computer cables may cause failures in external devices or the Coordinator Module.
- Do not connect pin 6 (+5V) on the RS-232C port on the Coordinator Module to any external device other than the NT-AL001 or CJ1W-CIF11 Conversion Adapter. Doing so may result in damage to the external device and the Coordinator Module.
- When replacing parts, be sure to confirm that the rating of a new part is correct. Not doing so may result in malfunction or burning.
- When transporting or storing the product, cover the PCBs with electrically conductive materials to prevent LSIs and ICs from being damaged by static electricity, and also keep the product within the specified storage temperature range.
- Do not touch the mounted parts or the rear surface of PCBs because PCBs have sharp edges such as electrical leads.
- When connecting the Power Supply Unit, Coordinator Module, Motion Control Module, I/O Control Module, CJ-series Units, and End Module, slide the upper and lower sliders until a click sound is heard to lock them securely. Desired functionality may not be achieved unless Modules are securely locked in place.
- Be sure to mount the End Module supplied with the Coordinator Module to the rightmost Module. Unless the End Module is properly mounted, the FQM1 will not function properly.
- Make sure that parameters are set correctly. Incorrect parameter settings may result in unexpected operations. Make sure that equipment will not be adversely affected by the parameter settings before starting or stopping the FQM1.

4 Conformance to EC Directives

4-1 Applicable Directives

- EMC Directives
- Low Voltage Directive

4-2 Concepts

EMC Directives

OMRON devices that comply with EC Directives also conform to the related EMC standards so that they can be more easily built into other devices or the overall machine. The actual products have been checked for conformity to EMC standards (see the following note). Whether the products conform to the standards in the system used by the customer, however, must be checked by the customer.

EMC-related performance of the OMRON devices that comply with EC Directives will vary depending on the configuration, wiring, and other conditions of the equipment or control panel on which the OMRON devices are installed. The customer must, therefore, perform the final check to confirm that devices and the overall machine conform to EMC standards.

Note Applicable EMC (Electromagnetic Compatibility) standards are as follows:

EMS (Electromagnetic Susceptibility): EN61000-6-2
EMI (Electromagnetic Interference): EN61000-6-4
(Radiated emission: 10-m regulations)

Low Voltage Directive

Always ensure that devices operating at voltages of 50 to 1,000 V AC and 75 to 1,500 V DC meet the required safety standards for the Motion Controller (EN61131-2).

4-3 Conformance to EC Directives

The FQM1-series Flexible Motion Controllers comply with EC Directives. To ensure that the machine or device in which the Motion Controller is used complies with EC Directives, the Motion Controller must be installed as follows:

1,2,3...

1. The Motion Controller must be installed within a control panel.
2. You must use reinforced insulation or double insulation for the DC power supplies used for the communications power supply and I/O power supplies.
3. Motion Controllers complying with EC Directives also conform to the Common Emission Standard (EN61000-6-4). Radiated emission characteristics (10-m regulations) may vary depending on the configuration of the control panel used, other devices connected to the control panel, wiring, and other conditions. You must therefore confirm that the overall machine or equipment complies with EC Directives.

4-4 EMC Directive Conformance Conditions

The immunity testing condition of the Motion Controller Modules is as follows:

Overall accuracy of FQM1-MMA22 analog I/O: +4%/–2%

4-5 Relay Output Noise Reduction Methods

The FQM1-series Flexible Motion Controller conforms to the Common Emission Standards (EN61000-6-4) of the EMC Directives. However, noise generated by relay output switching may not satisfy these Standards. In such a case, a noise filter must be connected to the load side or other appropriate countermeasures must be provided external to the Motion Controller.

Countermeasures taken to satisfy the standards vary depending on the devices on the load side, wiring, configuration of machines, etc. Following are examples of countermeasures for reducing the generated noise.

Countermeasures

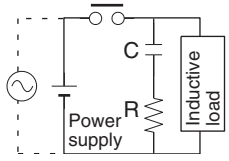
(Refer to EN61000-6-4 for more details.)

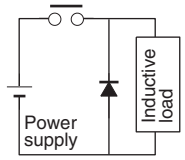
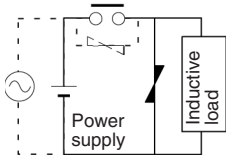
Countermeasures are not required if the frequency of load switching for the whole system with the Motion Controller included is less than 5 times per minute.

Countermeasures are required if the frequency of load switching for the whole system with the Motion Controller included is more than 5 times per minute.

Countermeasure Examples

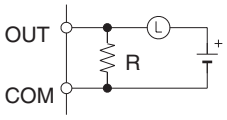
When switching an inductive load, connect a surge protector, diodes, etc., in parallel with the load or contact as shown below.

Circuit	Current		Characteristic	Required element
	AC	DC		
CR method 	Yes	Yes	If the load is a relay or solenoid, there is a time lag between the moment the circuit is opened and the moment the load is reset. If the supply voltage is 24 or 48 V, insert the surge protector in parallel with the load. If the supply voltage is 100 to 200 V, insert the surge protector between the contacts.	The capacitance of the capacitor must be 1 to 0.5 μF per contact current of 1 A and resistance of the resistor must be 0.5 to 1 Ω per contact voltage of 1 V. These values, however, vary with the load and the characteristics of the relay. Decide these values from experiments, and take into consideration that the capacitance suppresses spark discharge when the contacts are separated and the resistance limits the current that flows into the load when the circuit is closed again. The dielectric strength of the capacitor must be 200 to 300 V. If the circuit is an AC circuit, use a capacitor with no polarity.

Circuit	Current		Characteristic	Required element
	AC	DC		
Diode method 	No	Yes	The diode connected in parallel with the load changes energy accumulated by the coil into a current, which then flows into the coil so that the current will be converted into Joule heat by the resistance of the inductive load. This time lag, between the moment the circuit is opened and the moment the load is reset, caused by this method is longer than that caused by the CR method.	The reversed dielectric strength value of the diode must be at least 10 times as large as the circuit voltage value. The forward current of the diode must be the same as or larger than the load current. The reversed dielectric strength value of the diode may be two to three times larger than the supply voltage if the surge protector is applied to electronic circuits with low circuit voltages.
Varistor method 	Yes	Yes	The varistor method prevents the imposition of high voltage between the contacts by using the constant voltage characteristic of the varistor. There is time lag between the moment the circuit is opened and the moment the load is reset. If the supply voltage is 24 or 48 V, insert the varistor in parallel with the load. If the supply voltage is 100 to 200 V, insert the varistor between the contacts.	---

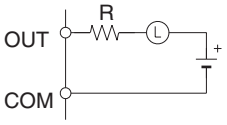
When switching a load with a high inrush current such as an incandescent lamp, suppress the inrush current as shown below.

Countermeasure 1



Providing a dark current of approx. one-third of the rated value through an incandescent lamp

Countermeasure 2



Providing a limiting resistor

The following Unit and Cables can be used with the FQM1-series Flexible Motion Controller.

Name	Model	Cable length
Relay Unit	XW2B-80J7-1A	---
Controller Connecting Cables	XW2Z-050J-A28	0.5 m
	XW2Z-100J-A28	1 m
	XW2Z-050J-A30	0.5 m
	XW2Z-100J-A30	1 m
	XW2Z-050J-A31	0.5 m
	XW2Z-100J-A31	1 m

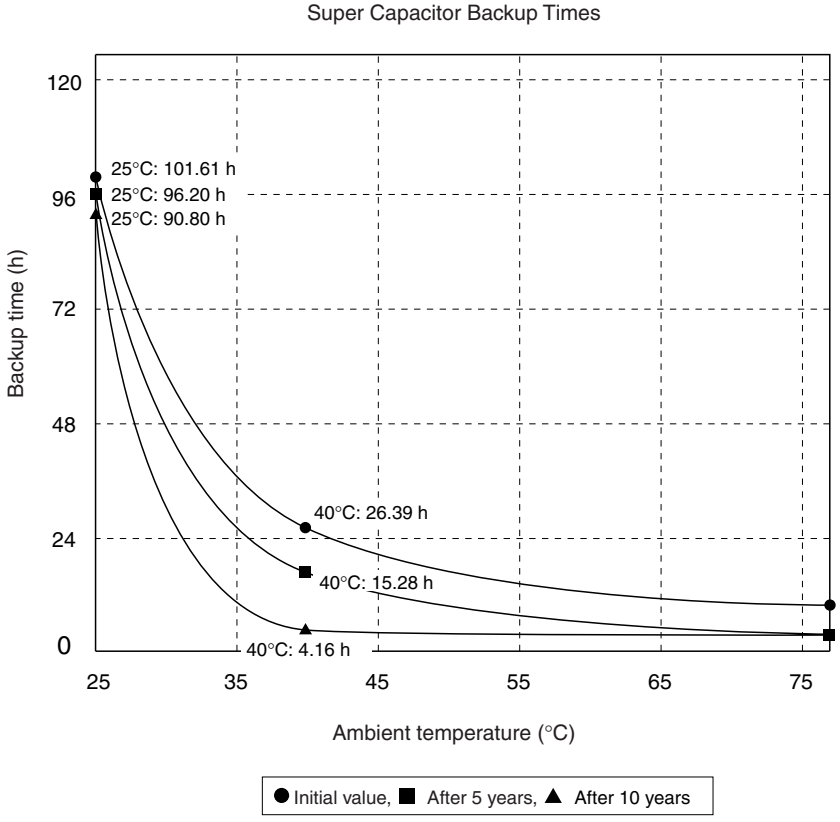
5 Data Backup

The user programs, I/O memories, and other data in the Coordinator Module and Motion Control Modules is backed up either by a super capacitor or flash memory, as listed in the following table.

Module	Data	Data backup
Coordinator Module	Error log	RAM with super capacitor
Motion Control Module	DM Area words D30000 to D32767 Error log	
Coordinator Module	User program System Setup DM Area words D20000 to D32767	Flash memory
Motion Control Module	User program System Setup DM Area words D00000 to D29999 (Auxiliary Area bit must be set.)	

The data backup time of the super capacitor is given in the following table and shown in the following graph.

Temperature	Initial	After 5 years	After 10 years
Ta = 25°C	101.61 hours (4.23 days)	96.2 hours (4.01days)	90.8 hours (3.78 days)
Ta = 40°C	26.39 hours (1.09 days)	15.28 hours	4.16 hours



Note 1. The times give above assume that the capacitor is completely charged. Power must be supply to the FQM1 for at least 20 minutes to completely charge the capacitor.

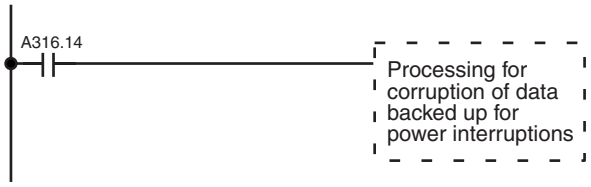
2. The backup time of the super capacitor is reduced as the capacitor ages. It is also affected by the ambient temperature. Use portion of the DM Area backed up by the super capacitor only for data that is to be held during momentary power interruptions. For operating parameters and other long-term data, use the portion of DM Area stored in flash memory in the Coordinator Module and transfer it to the Motion Control Modules before starting operation.

The data in the DM Area and error log will become unstable or corrupted if the power to the system is OFF for longer than the backup time.

If the power supply is to be turned OFF for an extended period of time, use D20000 to D32767 in the Coordinator Module and D00000 to D29999 in the Motion Control Module, which is backed up in flash memory, to store data.

Otherwise, the Memory Not Held Flag (A316.14) can be used as the input condition for programming using data in areas stored for power interruptions to perform suitable processing.

A316.14: Turns ON when power is turned ON if data stored for power interruptions in the DM Area or error log is corrupted.



Backing Up DM Area Data in Flash Memory

DM Area words D20000 to D32767 for the Coordinator Module and D00000 to D29999 for the Motion Control Modules are read from flash memory when the power supply is turned ON. (A Setup parameter must be set to read DM Area data for the Motion Control Modules.)

SECTION 1
Introduction

This section provides information on general instruction characteristics as well as the errors that can occur during instruction execution.

1-1 General Instruction Characteristics 2
1-1-1 Program Capacity 2
1-1-2 Differentiated Instructions 2
1-1-3 Instruction Variations 3
1-1-4 Instruction Location and Execution Conditions 3
1-1-5 Inputting Data in Operands 4
1-1-6 Data Formats 7
1-2 Instruction Execution Checks 8
1-2-1 Errors Occurring at Instruction Execution 8
1-2-2 Fatal Errors (Program Errors) 9

1-1 General Instruction Characteristics

1-1-1 Program Capacity

The program capacity tells the size of the user program area each Module and is expressed as the number of program steps. The number of steps required in the user program area for each instruction varies from 1 to 7 steps, depending upon the instruction and the operands used with it.

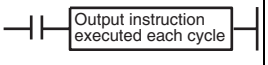
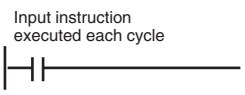
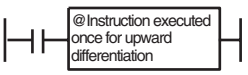
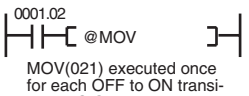
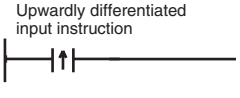
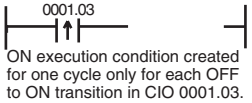
Model		Model	Program capacity
Coordinator Module		FQM1-CM002	10K steps
Motion Control Modules	Pulse I/O	FQM1-MMP22	
	Analog I/O	FQM1-MMA22	

Note The number of steps in a program is not the same as the number of instructions, i.e., each instruction contains from 1 to 7 steps. For example, LD and OUT require 1 step each, but MOV(021) requires 3 steps. The number of steps required by an instruction is also increased by one step for each double-length operand used in it. For example, MOVL(498) normally requires 3 steps, but 4 steps will be required if a constant is specified for the source word operand, S. Refer to *SECTION 4 Instruction Execution Times and Number of Steps* for the number of steps required for each instruction.

1-1-2 Differentiated Instructions

Most instructions in the FQM1 are provided with both non-differentiated and upwardly differentiated variations, and some are also provided with downwardly differentiated variations.

- A non-differentiated instruction is executed every time it is scanned.
- An upwardly differentiated instruction is executed only once after its execution condition goes from OFF to ON.
- A downwardly differentiated instruction is executed only once after its execution condition goes from ON to OFF.

Variation	Instruction type	Operation	Format	Example
Non-differentiated	Output instructions (instructions requiring an execution condition)	The instruction is executed every cycle while the execution condition is true (ON).		
	Input instructions (instructions used as execution conditions)	The bit processing (such as read, comparison, or test) is performed every cycle. The execution condition is true while the result is ON.		
Upwardly differentiated (with @ prefix)	Output instructions	The instruction is executed just once when the execution condition goes from OFF to ON.		
	Input instructions (instructions used as execution conditions)	The bit processing (such as read, comparison, or test) is performed every cycle. The execution condition is true for one cycle when the result goes from OFF to ON.		

Variation	Instruction type	Operation	Format	Example
Downwardly differentiated (with % prefix)	Output instructions	The instruction is executed just once when the execution condition goes from ON to OFF.	%Instruction executed once for downward differentiation	0001.02 %SET SET executed once for each ON to OFF transition in CIO 0001.02.
	Input instructions (instructions used as execution conditions)	The bit processing (such as read, comparison, or test) is performed every cycle. The execution condition is true for one cycle when the result goes from ON to OFF.	Downwardly differentiated input instruction	0001.03 ON execution condition created for one cycle only for each ON to OFF transition in CIO 0001.03.

Note The downwardly differentiated option (%) is available only for the LD, AND, OR, and RSET instructions. To create downwardly differentiated variations of other instructions, control the execution of the instruction with work bits controlled with DIFD(014).

1-1-3 Instruction Variations

The variation prefixes (@ and %) can be added to certain instructions to create a differentiated instruction.

Variation		Prefix	Operation
Differentiation	Upwardly differentiated	@	Creates an upwardly differentiated instruction.
	Downwardly differentiated	%	Creates a downwardly differentiated instruction.

@ MOV
Instruction mnemonic
Up-differentiation variation

1-1-4 Instruction Location and Execution Conditions

The following table shows the locations in which instructions can be programmed. The table also shows when an instruction requires an execution condition and when it does not. Refer to *SECTION 2 Summary of Instructions* for details on specific instructions.

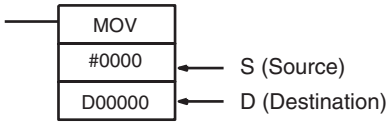
Instruction type		Location	Execution condition	Format	Examples
Input	Instructions that start logic conditions (load instructions)	At the left bus or at the start of an instruction block	Not required		LD and input comparison instructions such as LD >
	Connecting instructions	Between a starting instruction and output instruction	Required		AND, OR, and input comparison instructions, such as AND >
Output		At the right bus	Required		The majority of instructions (such as OUT and MOV)
			Not required		Instructions such as END, JME, and ILC

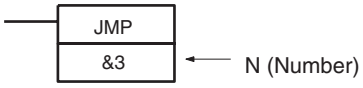
In addition to these instructions, the FQM1 is equipped with block programming instructions. Refer to the description of the block programming instructions for details.

Note If an execution condition does not precede an instruction that requires one, a program error will occur when the program is checked from the CX-Programmer.

1-1-5 Inputting Data in Operands

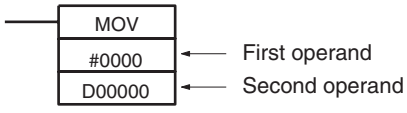
Operands are parameters that are set in advance with the I/O memory addresses or constants to be used when the instruction is executed. There are basically three kinds of operands: Source operands, destination operands, and numbers.





Operand		Usual code	Contents	
Source	Address containing the data or the data itself	S	Source operand	Source data other than control data
		C	Control data	Control data with a bit or bits controlling instruction execution
Destination	Address where the data will be stored	D	---	
Number	Contains a number, such as a jump number or subroutine number.	N	---	

Note An instruction's operands may also be referred to by their position in the instruction (first operand, second operand, ...). The codes used for the operand vary with the specific function of the operand.



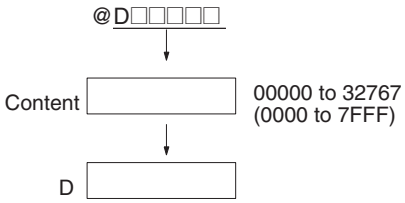
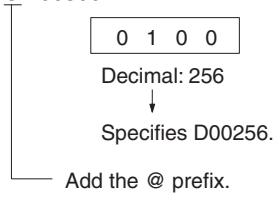
Specifying Bit Addresses

Description	Example	Instruction example
To specify a bit address, specify the word address and bit address directly. □□□□ . □□ Bit number (00 to 15) Word address **Note** The word address + bit number format is not used for Timer/Counter Completion Flags or Task Flags.	0001.02 Bit 02 Word CIO 0001	0001 02

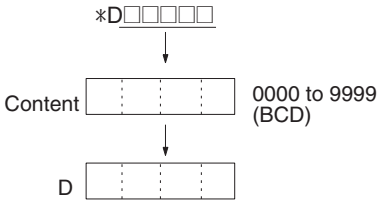
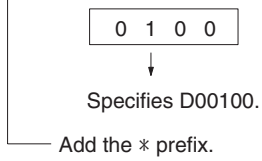
Specifying Word Addresses

Description	Example	Instruction example
To specify a word address, specify the word address directly. Each word contains 16 bits. □□□□ Word address	0003 Word CIO 0003 D00200 Word D00200	MOV 0003 D00200

Specifying Indirect DM Addresses in Binary Mode

Description	Example	Instruction example
When the @ prefix is input before a DM address, the contents of that word specifies another word that is used as the operand. The contents can be 0000 to 7FFF (0 to 32,767), corresponding to the desired word address in the DM Area. 	---	---
When the contents of @D□□□□□ is between 0000 and 7FFF (00000 to 32,767), the corresponding word between D00000 and D32767 is specified.	@D00300 	MOV #0001 @D00300

Specifying Indirect DM Addresses in BCD Mode

Method	Description	Example	Instruction example
Indirect DM addressing (BCD mode)	When the * prefix is input before a DM address, the BCD contents of that word specify another word that is used as the operand. The contents can be 0000 to 9999, corresponding to the desired word address in the DM Area. 	*D00200 	MOV #0001 *D00200

Specifying Constants

Method	Applicable operands	Data format	Code	Range
Constant, 16-bit data	All binary data and binary data within a range	Unsigned binary	#	#0000 to #FFFF
		Signed decimal	±	–32,768 to +32,767
		Unsigned decimal	&	&0 to &65,535
	All BCD data and BCD data within a range	BCD	#	#0000 to #9999
Constant, 32-bit data	All binary data and binary data within a range	Unsigned binary	#	#0000 0000 to #FFFF FFFF
		Signed decimal	+ –	–2,147,483,648 to +2,147,483,647
		Unsigned decimal	&	&0 to &4,294,967,295
	All BCD data and BCD data within a range	BCD	#	#0000 0000 to #9999 9999

Specifying Text Strings

Method	Description	Code	Examples
Text strings	Text is stored in ASCII (1 byte/character excluding special characters) in the order from the higher to lower byte and lowest to highest word. If there is an odd number of characters, 00 (NULL) is stored in the higher byte of the last word in the range. If there is an even number of characters, 0000 (two NULLs) are stored in the word after the last in the range.		"ABCDE" "A" "B" "C" "D" "E" NUL || 41 42 43 44 45 00 "ABCD" "A" "B" "C" "D" NUL NUL || 41 42 43 44 00 00

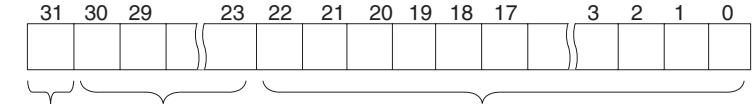
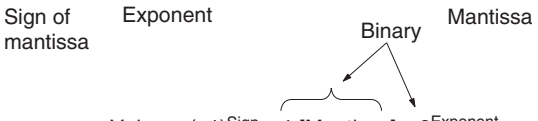
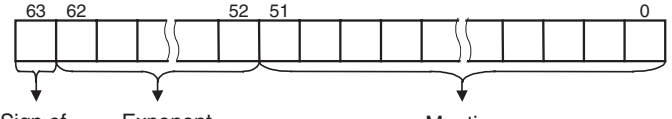
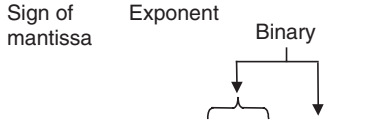
The following diagram shows the characters that can be expressed in ASCII.

Rightmost bit \ Leftmost bit	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	0	@	P	‘	p				—	タ	ミ		
1			!	1	A	Q	a	q			。	ア	チ	ム		
2			”	2	B	R	b	r			「	イ	ツ	メ		
3			#	3	C	S	c	s			」	ウ	テ	モ		
4			$	4	D	T	d	t			、	エ	ト	ヤ		
5			%	5	E	U	e	u			・	オ	ナ	ユ		
6			&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7			’	7	G	W	g	w			ァ	キ	ヌ	ラ		
8			(	8	H	X	h	x			ィ	ク	ネ	リ		
9			)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
A			*	:	J	Z	j	z			ェ	コ	ハ	レ		
B			+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C			,	<	L	¥	l	|			ャ	シ	フ	ワ		
D			−	=	M	]	m	}			ュ	ス	ヘ	ン		
E			.	>	N	^	n	~			ョ	セ	ホ	゛		
F			/	?	O	_	o				ッ	ソ	マ	°		

1-1-6 Data Formats

The following table shows the data formats that can be used in the FQM1.

Name	Format	Decimal range	Hexadecimal range
Unsigned binary data	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 Binary: 2^{15} 2^{14} 2^{13} 2^{12} 2^{11} 2^{10} 2^{9} 2^{8} 2^{7} 2^{6} 2^{5} 2^{4} 2^{3} 2^{2} 2^{1} 2^{0} Decimal: 32768 16384 8192 4096 2048 1024 512 256 128 64 32 16 8 4 2 1 Hexa-decimal: 2^{3} 2^{2} 2^{1} 2^{0} 2^{3} 2^{2} 2^{1} 2^{0} 2^{3} 2^{2} 2^{1} 2^{0} 2^{3} 2^{2} 2^{1} 2^{0}	0 to 65,535	0000 to FFFF
Signed binary data	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 Binary: 2^{15} 2^{14} 2^{13} 2^{12} 2^{11} 2^{10} 2^{9} 2^{8} 2^{7} 2^{6} 2^{5} 2^{4} 2^{3} 2^{2} 2^{1} 2^{0} Decimal: -32768 16384 8192 4096 2048 1024 512 256 128 64 32 16 8 4 2 1 Hexa-decimal: 2^{3} 2^{2} 2^{1} 2^{0} 2^{3} 2^{2} 2^{1} 2^{0} 2^{3} 2^{2} 2^{1} 2^{0} 2^{3} 2^{2} 2^{1} 2^{0} Sign bit: 0: Positive, 1: Negative	−32,768 to +32,767	8000 to 7FFF
BCD data	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 BCD: 2^{3} 2^{2} 2^{1} 2^{0} 2^{3} 2^{2} 2^{1} 2^{0} 2^{3} 2^{2} 2^{1} 2^{0} 2^{3} 2^{2} 2^{1} 2^{0} Decimal: 0 to 9 | 0 to 9 | 0 to 9 | 0 to 9	0 to 9,999	0000 to 9999

Name	Format	Decimal range	Hexadecimal range
Floating-point decimal	  Value = $(-1)^{Sign}$ x 1.[Mantissa] x $2^{Exponent}$ Sign (bit 31) 1: negative or 0: positive Mantissa The mantissa includes 23 bits from bit 00 to bit 22 and indicates this portion below the decimal point in 1.□□□..... in binary. Exponent The exponent includes 8 bits from bit 23 to bit 30 and indicates n plus 127 in 2^n in binary. **Note** This format conforms to IEEE754 standards for single-precision floating-point data and is used only with instructions that convert or calculate floating-point data. It can be used to set or monitor from the I/O memory Edit and Monitor Screen on the CX-Programmer. As such, users do not need to know this format although they do need to know that the formatting takes up two words.	---	---
Double-precision floating-point decimal	  Value = $(-1)^{Sign}$ x 1.[Mantissa] x $2^{Exponent}$ Sign (bit 63) 1: negative or 0: positive Mantissa The mantissa includes 52 bits from bit 00 to bit 51 and indicates this portion below the decimal point in 1.□□□..... in binary. Indicates this value. Exponent The exponent includes 11 bits from bit 52 to bit 62 and indicates n plus 127 in 2_n in binary. The actual value will be 2^{n-1023}. **Note** This format conforms to IEEE754 standards for double-precision floating-point data and is used only with instructions that convert or calculate floating-point data. It can be used to set or monitor from the I/O memory Edit and Monitor Screen on the CX-Programmer. As such, users do not need to know this format although they do need to know that the formatting takes up four words.		

Signed Binary Numbers

Negative signed-binary numbers are expressed as the 2's complement of the absolute hexadecimal value. For a decimal value of –12,345, the absolute value is equivalent to 3039 hexadecimal. The 2's complement is 10000 – 3039 (both hexadecimal) or CFC7.

To convert from a negative signed binary number (CFC7) to decimal, take the 2's complement of that number (10000 – CFC7 = 3039), convert to decimal (3039 hexadecimal = 12,345 decimal), and add a minus sign (–12,345).

1-2 Instruction Execution Checks

1-2-1 Errors Occurring at Instruction Execution

An instruction's operands and placement are checked when an instruction is input from the CX-Programmer or a program check is performed from the CX-

Programmer, but these are not final checks. The following errors can occur when an instruction is executed.

Error	Flag	Fatal/Non-fatal
Instruction Processing Error	ER Flag ON	Non-fatal
Illegal Instruction Error	Illegal Instruction Error Flag (A295.14)	Fatal (program error)
UM (User Program Memory) Overflow Error	UM Overflow Error Flag (A295.15)	Fatal (program error)

1-2-2 Fatal Errors (Program Errors)

Program execution will be stopped when one of the following program errors occurs. All errors for which the Error Flag or Access Error Flag turns ON is treated as a program error. The following table lists program errors. The System Setup can be set to stop program execution when one of these errors occurs.

Error type	Description	Related flags
No END Instruction	There is no END(001) instruction in the program.	No END Error Flag (A295.11)
Task Error	An interrupt was generated but the corresponding interrupt task does not exist.	Task Error Flag (A295.12)
Instruction Processing Error	The CPU attempted to execute an instruction, but the data provided in the instruction's operand was incorrect.	Error (ER) Flag, Instruction Processing Error Flag (A295.08)
Differentiation Overflow Error	Differentiated instructions were repeatedly inserted and deleted during online editing (over 131,072 times).	Differentiation Overflow Error Flag (A295.13)
UM Overflow Error	The last address in UM (user program memory) has been exceeded.	UM Overflow Error Flag (A295.15)
Illegal Instruction Error	The program contains an instruction that cannot be executed.	Illegal Instruction Error Flag (A295.14)

SECTION 2
Summary of Instructions

This section provides a summary of instructions used with the FQM1.

2-1 Instruction Classifications by Function 12
2-2 Instruction Functions 18
2-2-1 Sequence Input Instructions 18
2-2-2 Sequence Output Instructions 20
2-2-3 Sequence Control Instructions 23
2-2-4 Timer and Counter Instructions 26
2-2-5 Comparison Instructions 28
2-2-6 Data Movement Instructions 31
2-2-7 Data Shift Instructions 34
2-2-8 Increment/Decrement Instructions 38
2-2-9 Symbol Math Instructions 39
2-2-10 Conversion Instructions 44
2-2-11 Logic Instructions 47
2-2-12 Special Math Instructions 49
2-2-13 Floating-point Math Instructions 49
2-2-14 Double-precision Floating-point Instructions 53
2-2-15 Table Data Processing Instructions 57
2-2-16 Data Control Instructions 58
2-2-17 Subroutine Instructions 60
2-2-18 Interrupt Control Instructions 61
2-2-19 High-speed Counter and Pulse Output Instructions 63
2-2-20 Step Instructions 64
2-2-21 I/O Refresh Instructions 64
2-2-22 Serial Communications Instructions 65
2-2-23 Debugging Instructions 65
2-2-24 Failure Diagnosis Instructions 66
2-2-25 Other Instructions 66
2-2-26 Block Programming Instructions 66
2-2-27 Special Function Block Instructions 68
2-3 Alphabetical List of Instructions by Mnemonic 69
2-4 List of Instructions by Function Code 78

2-1 Instruction Classifications by Function

The following table lists the FQM1 instructions by function. (The instructions appear by order of their function in *Section 3 Instructions*.)

Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Sequence input instructions	**---**	LD	LOAD	LD NOT	LOAD NOT	AND	AND
		AND NOT	AND NOT	OR	OR	OR NOT	OR NOT
		AND LD	AND LOAD	OR LD	OR LOAD	NOT	NOT
		UP	CONDITION ON	DOWN	CONDITION OFF	---	---
	Bit test	LD TST	LD BIT TEST	LD TSTN	LD BIT TEST NOT	AND TST	AND BIT TEST NOT
		AND TSTN	AND BIT TEST NOT	OR TST	OR BIT TEST	OR TSTN	OR BIT TEST NOT
Sequence output instructions	**---**	OUT	OUTPUT	OUT NOT	OUTPUT NOT	KEEP	KEEP
		DIFU	DIFFERENTIATE UP	DIFD	DIFFERENTIATE DOWN	OUTB	SINGLE BIT OUTPUT
	Set/Reset	SET	SET	RSET	RESET	SETA	MULTIPLE BIT SET
		RSTA	MULTIPLE BIT RESET	SETB	SINGLE BIT SET	RSTB	SINGLE BIT RESET
Sequence control instructions	**---**	END	END	NOP	NO OPERATION	---	---
	Interlock	IL	INTERLOCK	ILC	INTERLOCK CLEAR	---	---
	Jump	JMP	JUMP	JME	JUMP END	CJP	CONDITIONAL JUMP
		CJPN	CONDITIONAL JUMP	JMP0	MULTIPLE JUMP	JME0	MULTIPLE JUMP END
	Repeat	FOR	FOR-NEXT LOOPS	BREAK	BREAK LOOP	NEXT	FOR-NEXT LOOPS
Timer and counter instructions (BCD)	**Timer (with timer numbers)**	TIM	TIMER	TIMH	HIGH-SPEED TIMER	TMHH	ONE-MS TIMER
	Counter (with counter numbers)	CNT	COUNTER	CNTR	REVERSIBLE TIMER	---	---
Comparison instructions	**Symbol comparison**	LD, AND, OR + =, <>, <, <=, >, >=	Symbol comparison (unsigned)	LD, AND, OR + =, <>, <, <=, >, >= + L	Symbol comparison (double-word, unsigned)	LD, AND, OR + =, <>, <, <=, >, >= +S	Symbol comparison (signed)
		LD, AND, OR + =, <>, <, <=, >, >= + SL	Symbol comparison (double-word, signed)	---	---	---	---
	Data comparison (Condition Flags)	CMP	UNSIGNED COMPARE	CMPL	DOUBLE UNSIGNED COMPARE	CPS	SIGNED BINARY COMPARE
		CPSL	DOUBLE SIGNED BINARY COMPARE	ZCP	AREA RANGE COMPARE	ZCPL	DOUBLE AREA RANGE COMPARE
	Table compare	MCMP	MULTIPLE COMPARE	TCMP	TABLE COMPARE	BCMP	UNSIGNED BLOCK COMPARE
		BCMP2	EXPANDED BLOCK COMPARE	---	---	---	---

Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Data movement instructions	**Single/ double-word**	MOV	MOVE	MOVL	DOUBLE MOVE	MVN	MOVE NOT
		MVNL	DOUBLE MOVE NOT	---	---	---	---
	Bit/digit	MOVB	MOVE BIT	MOVD	MOVE DIGIT	---	---
	Exchange	XCHG	DATA EXCHANGE	XCGL	DOUBLE DATA EXCHANGE	---	---
	Block/bit transfer	XFRB	MULTIPLE BIT TRANSFER	XFER	BLOCK TRANSFER	BSET	BLOCK SET
	Distribute/collect	DIST	SINGLE WORD DISTRIBUTE	COLL	DATA COLLECT	---	---
	Index register	MOVR	MOVE TO REGISTER	MOVRW	MOVE TIMER/ COUNTER PV TO REGISTER	---	---
Data shift instructions	**1-bit shift**	SFT	SHIFT REGISTER	SFTR	REVERSIBLE SHIFT REGISTER	ASLL	DOUBLE SHIFT LEFT
		ASL	ARITHMETIC SHIFT LEFT	ASR	ARITHMETIC SHIFT RIGHT	ASRL	DOUBLE SHIFT RIGHT
	0000 hex asynchronous	ASFT	ASYNCHRONOUS SHIFT REGISTER	---	---	---	---
	Word shift	WSFT	WORD SHIFT	---	---	---	---
	1-bit rotate	ROL	ROTATE LEFT	ROLL	DOUBLE ROTATE LEFT	RLNC	ROTATE LEFT WITHOUT CARRY
		RLNL	DOUBLE ROTATE LEFT WITHOUT CARRY	ROR	ROTATE RIGHT	RORL	DOUBLE ROTATE RIGHT
		RRNC	ROTATE RIGHT WITHOUT CARRY	RRNL	DOUBLE ROTATE RIGHT WITHOUT CARRY	---	---
	1 digit shift	SLD	ONE DIGIT SHIFT LEFT	SRD	ONE DIGIT SHIFT RIGHT	---	---
	Shift n-bit data	NASL	SHIFT N-BIT DATA LEFT	NSLL	DOUBLE SHIFT N-BIT DATA LEFT	NASR	SHIFT N-BIT DATA RIGHT
		NSRL	DOUBLE SHIFT N-BIT DATA RIGHT	---	---	---	---
Increment/ decrement instructions	**BCD**	++B	INCREMENT BCD	++BL	DOUBLE INCREMENT BCD	– –B	DECREMENT BCD
		– –BL	DOUBLE DECREMENT BCD	---	---	---	---
	Binary	++	INCREMENT BINARY	++L	DOUBLE INCREMENT BINARY	– –	DECREMENT BINARY
		– –L	DOUBLE DECREMENT BINARY	---	---	---	---

Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Symbol math instructions	**Binary add**	+	SIGNED BINARY ADD WITHOUT CARRY	+L	DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+C	SIGNED BINARY ADD WITH CARRY
		+CL	DOUBLE SIGNED BINARY ADD WITH CARRY	---	---	---	---
	BCD add	+B	BCD ADD WITHOUT CARRY	+BL	DOUBLE BCD ADD WITHOUT CARRY	+BC	BCD ADD WITH CARRY
		+BCL	DOUBLE BCD ADD WITH CARRY	---	---	---	---
	Binary subtract	−	SIGNED BINARY SUBTRACT WITHOUT CARRY	−L	DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	−C	SIGNED BINARY SUBTRACT WITH CARRY
		−CL	DOUBLE SIGNED BINARY WITH CARRY	---	---	---	---
	BCD subtract	−B	BCD SUBTRACT WITHOUT CARRY	−BL	DOUBLE BCD SUBTRACT WITHOUT CARRY	−BC	BCD SUBTRACT WITH CARRY
		−BCL	DOUBLE BCD SUBTRACT WITH CARRY	---	---	---	---
	Binary multiply	*	SIGNED BINARY MULTIPLY	*L	DOUBLE SIGNED BINARY MULTIPLY	*U	UNSIGNED BINARY MULTIPLY
		*UL	DOUBLE UNSIGNED BINARY MULTIPLY	---	---	---	---
	BCD multiply	*B	BCD MULTIPLY	*BL	DOUBLE BCD MULTIPLY	---	---
	Binary divide	/	SIGNED BINARY DIVIDE	/L	DOUBLE SIGNED BINARY DIVIDE	/U	UNSIGNED BINARY DIVIDE
		/UL	DOUBLE UNSIGNED BINARY DIVIDE	---	---	---	---
	BCD divide	/B	BCD DIVIDE	/BL	DOUBLE BCD DIVIDE	---	---
Conversion instructions	**BCD/Binary convert**	BIN	BCD-TO-BINARY	BINL	DOUBLE BCD-TO-DOUBLE BINARY	BCD	BINARY-TO-BCD
		BCDL	DOUBLE BINARY-TO-DOUBLE BCD	NEG	2'S COMPLEMENT	NEGL	DOUBLE 2'S COMPLEMENT
	ASCII/HEX convert	ASC	ASCII CONVERT	HEX	ASCII TO HEX	---	---

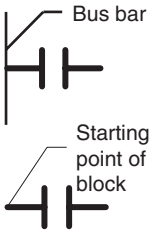
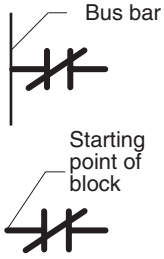
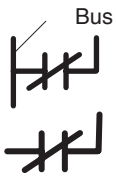
Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Logic instructions	**Logical AND/OR**	ANDW	LOGICAL AND	ANDL	DOUBLE LOGICAL AND	ORW	LOGICAL OR
		ORWL	DOUBLE LOGICAL OR	XORW	EXCLUSIVE OR	XORL	DOUBLE EXCLUSIVE OR
		XNRW	EXCLUSIVE NOR	XNRL	DOUBLE EXCLUSIVE NOR	---	---
	Complement	COM	COMPLE-MENT	COML	DOUBLE COMPLE-MENT	---	---
Special math instructions	**---**	APR	ARITHMETIC PROCESS	BCNT	BIT COUNTER	AXIS	VIRTUAL AXIS
Floating-point math instructions	**Floating point/ binary convert**	FIX	FLOATING TO 16-BIT	FIXL	FLOATING TO 32-BIT	FLT	16-BIT TO FLOATING
		FLTL	32-BIT TO FLOATING	---	---	---	---
	Floating- point basic math	+F	FLOATING-POINT ADD	–F	FLOATING-POINT SUBTRACT	/F	FLOATING-POINT DIVIDE
		*F	FLOATING-POINT MULTIPLY	---	---	---	---
	Floating- point trigonometric	RAD	DEGREES TO RADIANS	DEG	RADIANS TO DEGREES	SIN	SINE
		COS	COSINE	TAN	TANGENT	ASIN	ARC SINE
		ACOS	ARC COSINE	ATAN	ARC TAN-GENT	---	---
	Floating- point math	SQRT	SQUARE ROOT	EXP	EXPONENT	LOG	LOGARITHM
		PWR	EXPONEN-TIAL POWER	---	---	---	---
	Symbol compari-son and conver-sion*	LD, AND, OR + =, <>, <, <=, >, >= + F	Symbol com-parison (sin-gle-precision floating point)	---	---	---	---

Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Double-precision floating- point instructions*	**Floating point/ binary convert**	FIXD	DOUBLE FLOATING TO 16-BIT	FIXLD	DOUBLE FLOATING TO 32-BIT	DBL	16-BIT TO DOUBLE FLOATING
		DBLL	32-BIT TO DOUBLE FLOATING	---	---	---	---
	Floating- point basic math	+D	DOUBLE FLOATING-POINT ADD	–D	DOUBLE FLOATING-POINT SUBTRACT	/D	DOUBLE FLOATING-POINT DIVIDE
		*D	DOUBLE FLOATING-POINT MULTIPLY	---	---	---	---
	Floating- point trigonometric	RADD	DOUBLE DEGREES TO RADIANS	DEGD	DOUBLE RADIANS TO DEGREES	SIND	DOUBLE SINE
		COSD	DOUBLE COSINE	TAND	DOUBLE TANGENT	ASIND	DOUBLE ARC SINE
		ACOSD	DOUBLE ARC COSINE	ATAND	DOUBLE ARC TANGENT	---	---
	Floating- point math	SQRTD	DOUBLE SQUARE ROOT	EXPD	DOUBLE EXPONENT	LOGD	DOUBLE LOGARITHM
		PWRD	DOUBLE EXPONENTIAL POWER	---	---	---	---
	Symbol comparison	LD, AND, OR + =, <>, <, <=, >, >= + D	Symbol comparison (double-precision floating point)	---	---	---	---
Table data processing instructions	**Record-to-word processing**	MAX	FIND MAXIMUM	MIN	FIND MINIMUM	---	---
Data control instructions	**---**	SCL	SCALING	SCL2	SCALING 2	SCL3	SCALING 3
		AVG	AVERAGE	---	---	---	---
Subroutines instructions	**---**	SBS	SUBROUTINE CALL	MCRO	MACRO	SBN	SUBROUTINE ENTRY
		RET	SUBROUTINE RETURN	JSB	JUMP TO SUBROUTINE	---	---
Interrupt control instructions	**---**	MSKS	SET INTERRUPT MASK	MSKR	READ INTERRUPT MASK	CLI	CLEAR INTERRUPT
		DI	DISABLE INTERRUPTS	EI	ENABLE INTERRUPTS	STIM	INTERVAL TIMER
High-speed counter/ pulse output instructions	**---**	INI	MODE CONTROL	PRV	HIGH-SPEED COUNTER PV READ	---	---
		CTBL	COMPARISON TABLE LOAD	SPED	SPEED OUTPUT	PULS	SET PULSES
		PLS2	PULSE OUTPUT	ACC	ACCELERATION CONTROL	---	---
Step instructions	**---**	STEP	STEP DEFINE	SNXT	STEP START		
I/O Refresh instructions	**---**	IORF	I/O REFRESH	---	---	---	---
Serial communications instructions	**---**	TXD	TRANSMIT	RXD	RECEIVE	STUP	CHANGE SERIAL PORT SETUP
Debugging instructions	**---**	TRSM	TRACE MEMORY SAMPLING	---	---	---	---

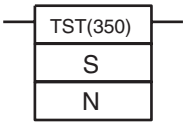
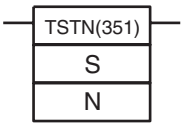
Classification	Sub-class	Mnemonic	Instruction	Mnemonic	Instruction	Mnemonic	Instruction
Failure diagnosis instructions	**---**	FAL	FAILURE ALARM	FALS	SEVERE FAILURE ALARM	---	---
Other instructions	**---**	STC	SET CARRY	CLC	CLEAR CARRY	---	---
Block programming instructions	**Define block program area**	BPRG	BLOCK PROGRAM BEGIN	BEND	BLOCK PROGRAM END	---	---
	IF branch processing	IF *bit_address*	CONDITIONAL BLOCK BRANCHING	IF NOT *bit_address*	CONDITIONAL BLOCK BRANCHING (NOT)	ELSE	CONDITIONAL BLOCK BRANCHING (ELSE)
		IEND	CONDITIONAL BLOCK BRANCHING END	---	---	---	---
Special Function Block Instructions	**---**	GETID	GET VARIABLE ID	---	---	---	---

2-2 Instruction Functions

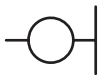
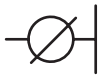
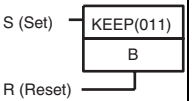
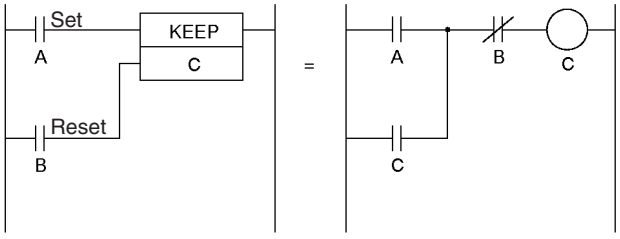
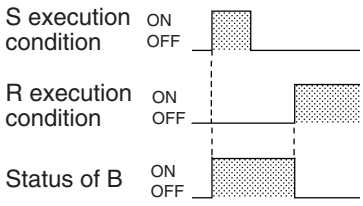
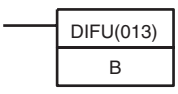
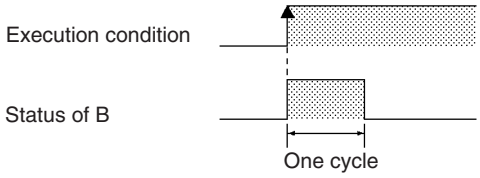
2-2-1 Sequence Input Instructions

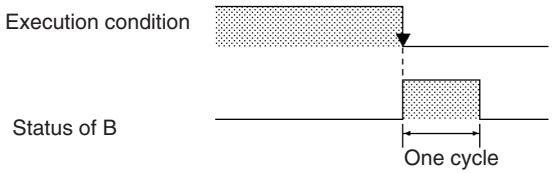
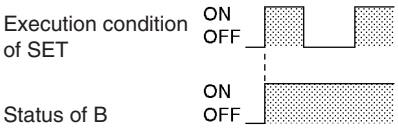
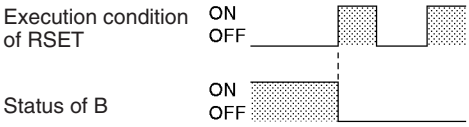
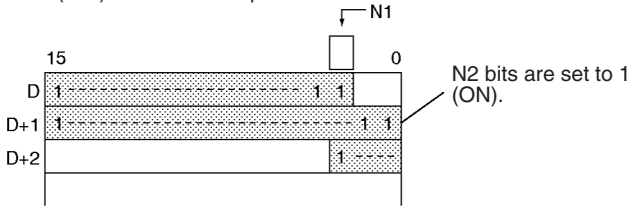
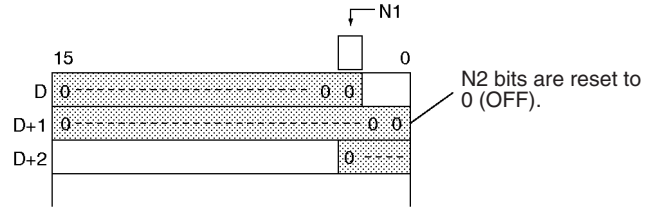
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
LOAD LD @LD %LD		Indicates a logical start and creates an ON/OFF execution condition based on the ON/OFF status of the specified operand bit.	Start of logic Not required	95
LOAD NOT LD NOT @LD NOT %LD NOT		Indicates a logical start and creates an ON/OFF execution condition based on the reverse of the ON/OFF status of the specified operand bit.	Start of logic Not required	97
AND AND @AND %AND		Takes a logical AND of the status of the specified operand bit and the current execution condition.	Continues on rung Required	99
AND NOT AND NOT @AND NOT %AND NOT		Reverses the status of the specified operand bit and takes a logical AND with the current execution condition.	Continues on rung Required	101
OR OR @OR %OR		Takes a logical OR of the ON/OFF status of the specified operand bit and the current execution condition.	Continues on rung Required	102
OR NOT OR NOT @OR NOT %OR NOT		Reverses the status of the specified bit and takes a logical OR with the current execution condition	Continues on rung Required	104
AND LOAD AND LD		Takes a logical AND between logic blocks. LD to Logic block A LD to Logic block B AND LD ····· Serial connection between logic block A and logic block B.	Continues on rung Required	105

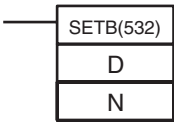
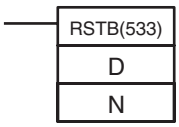
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
OR LOAD OR LD	Logic block Logic block	Takes a logical OR between logic blocks. LD to: Logic block A LD to: Logic block B OR LD: Parallel connection between logic block A and logic block B.	Continues on rung Required	107
NOT NOT 520	---	Reverses the execution condition.	Continues on rung Required	111
CONDITION ON UP 521	UP(521)	UP(521) turns ON the execution condition for one cycle when the execution condition goes from OFF to ON.	Continues on rung Required	112
CONDITION OFF DOWN 522	DOWN(522)	DOWN(522) turns ON the execution condition for one cycle when the execution condition goes from ON to OFF.	Continues on rung Required	112
BIT TEST LD TST 350	TST(350) S N **S**: Source word **N**: Bit number	LD TST(350), AND TST(350), and OR TST(350) are used in the program like LD, AND, and OR; the execution condition is ON when the specified bit in the specified word is ON and OFF when the bit is OFF.	Continues on rung Not required	113
BIT TEST LD TSTN 351	TSTN(351) S N **S**: Source word **N**: Bit number	LD TSTN(351), AND TSTN(351), and OR TSTN(351) are used in the program like LD NOT, AND NOT, and OR NOT; the execution condition is OFF when the specified bit in the specified word is ON and ON when the bit is OFF.	Continues on rung Not required	113
BIT TEST AND TST 350	AND TST(350) S N **S**: Source word **N**: Bit number	LD TST(350), AND TST(350), and OR TST(350) are used in the program like LD, AND, and OR; the execution condition is ON when the specified bit in the specified word is ON and OFF when the bit is OFF.	Continues on rung Required	113
BIT TEST AND TSTN 351	AND TSTN(351) S N **S**: Source word **N**: Bit number	LD TSTN(351), AND TSTN(351), and OR TSTN(351) are used in the program like LD NOT, AND NOT, and OR NOT; the execution condition is OFF when the specified bit in the specified word is ON and ON when the bit is OFF.	Continues on rung Required	113

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
BIT TEST OR TST 350	 **S**: Source word **N**: Bit number	LD TST(350), AND TST(350), and OR TST(350) are used in the program like LD, AND, and OR; the execution condition is ON when the specified bit in the specified word is ON and OFF when the bit is OFF.	Continues on rung Required	113
BIT TEST OR TSTN 351	 **S**: Source word **N**: Bit number	LD TSTN(351), AND TSTN(351), and OR TSTN(351) are used in the program like LD NOT, AND NOT, and OR NOT; the execution condition is OFF when the specified bit in the specified word is ON and ON when the bit is OFF.	Continues on rung Required	113

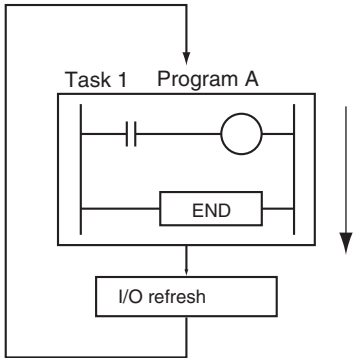
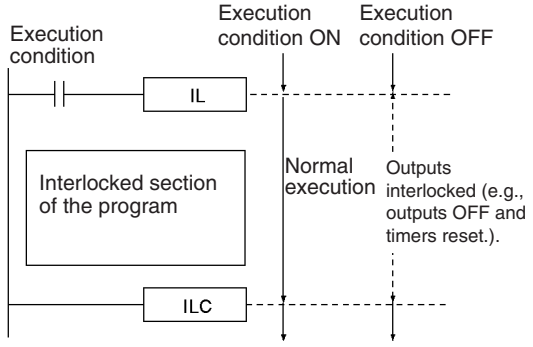
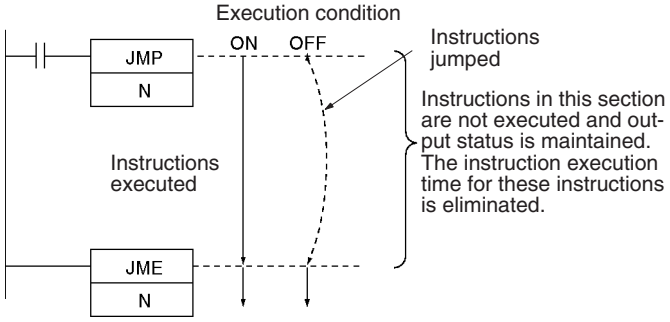
2-2-2 Sequence Output Instructions

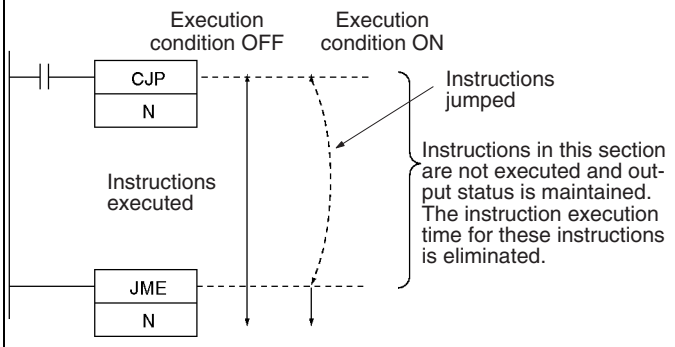
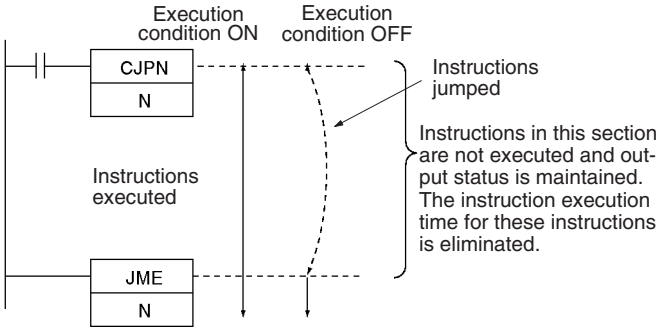
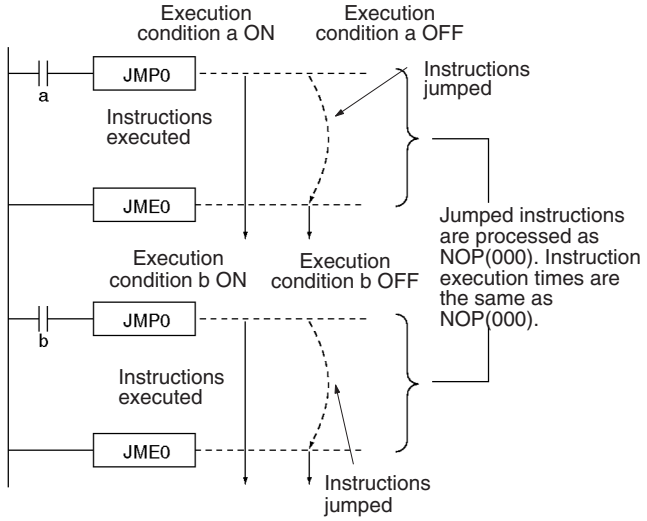
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
OUTPUT OUT		Outputs the result (execution condition) of the logical processing to the specified bit.	Output Required	117
OUTPUT NOT OUT NOT		Reverses the result (execution condition) of the logical processing, and outputs it to the specified bit.	Output Required	118
KEEP KEEP 011	 **B**: Bit	Operates as a latching relay.  	Output Required	119
DIFFERENTIATE UP DIFU 013	 **B**: Bit	DIFU(013) turns the designated bit ON for one cycle when the execution condition goes from OFF to ON (rising edge). 	Output Required	122

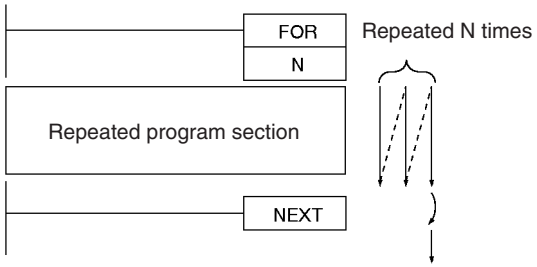
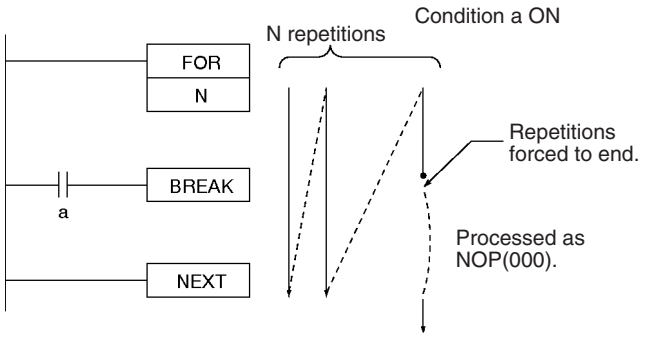
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DIFFERENTIATE DOWN DIFD 014	DIFD(014) B **B**: Bit	DIFD(014) turns the designated bit ON for one cycle when the execution condition goes from ON to OFF (falling edge). 	Output Required	122
SINGLE BIT OUTPUT OUTB @OUTB	OUTB(534) D N **D**: Word address **N**: Bit number	OUTB(534) outputs the result (execution condition) of the logical processing to the specified bit. Unlike the OUT instruction, OUTB(534) can be used to control a bit in a DM or EM word.	Output Required	132
SET SET @SET %SET	SET B **B**: Bit	SET turns the operand bit ON when the execution condition is ON. 	Output Required	125
RESET RSET @RSET %RSET	RSET B **B**: Bit	RSET turns the operand bit OFF when the execution condition is ON. 	Output Required	125
MULTIPLE BIT SET SETA @SETA 530	SETA(530) D N1 N2 **D**: Beginning word **N1**: Beginning bit **N2**: Number of bits	SETA(530) turns ON the specified number of consecutive bits. 	Output Required	126
MULTIPLE BIT RESET RSTA @RSTA 531	RSTA(531) D N1 N2 **D**: Beginning word **N1**: Beginning bit **N2**: Number of bits	RSTA(531) turns OFF the specified number of consecutive bits. 	Output Required	126

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SINGLE BIT SET SETB @SETB	 **D**: Word address **N**: Bit number	SETB(532) turns ON the specified bit in the specified word when the execution condition is ON. Unlike the SET instruction, SETB(532) can be used to set a bit in a DM or EM word.	Output Required	129
SINGLE BIT RESET RSTB @RSTB	 **D**: Word address **N**: Bit number	RSTB(533) turns OFF the specified bit in the specified word when the execution condition is ON. Unlike the RSET instruction, RSTB(533) can be used to reset a bit in a DM or EM word.	Output Required	129

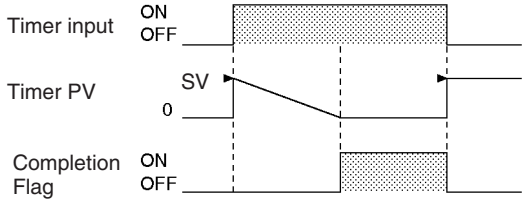
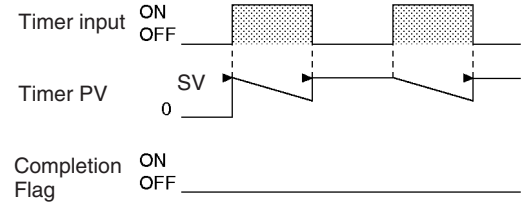
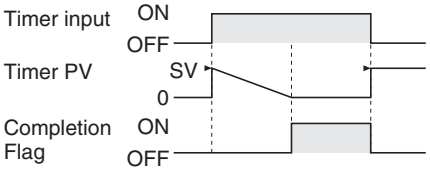
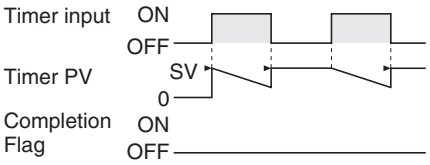
2-2-3 Sequence Control Instructions

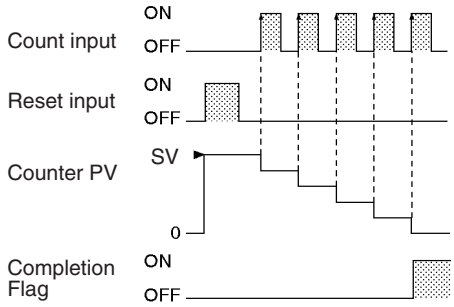
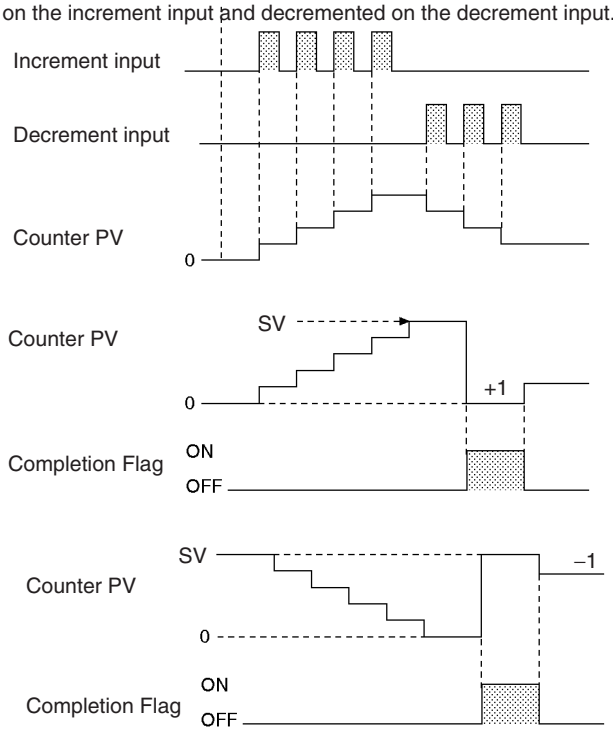
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
END END 001	END(001)	Indicates the end of a program. 	Output Not required	134
NO OPERATION NOP 000		This instruction has no function. (No processing is performed for NOP(000).)	Output Not required	134
INTERLOCK IL 002	IL(002)	Interlocks all outputs between IL(002) and ILC(003) when the execution condition for IL(002) is OFF. IL(002) and ILC(003) are normally used in pairs. 	Output Required	135
INTERLOCK CLEAR ILC 003	ILC(003)	Indicates the end to an interlocked program section. All outputs between IL(002) and ILC(003) are interlocked when the execution condition for IL(002) is OFF. IL(002) and ILC(003) are normally used in pairs.	Output Not required	135
JUMP JMP 004	JMP(004) N **N**: Jump number	When the execution condition for JMP(004) is OFF, program execution jumps directly to the first JME(005) in the program with the same jump number. JMP(004) and JME(005) are used in pairs. 	Output Required	138
JUMP END JME 005	JME(005) N **N**: Jump number	Indicates the end of a jump initiated by JMP(004).	Output Not required	138

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
CONDITIONAL JUMP CJP 510	CJP(510) N **N**: Jump number	The operation of CJP(510) is the basically the opposite of JMP(004). When the execution condition for CJP(510) is ON, program execution jumps directly to the first JME(005) in the program with the same jump number. CJP(510) and JME(005) are used in pairs. 	Output Required	141
CONDITIONAL JUMP CJPN 511	CJPN(511) N **N**: Jump number	The operation of CJPN(511) is almost identical to JMP(004). When the execution condition for CJP(004) is OFF, program execution jumps directly to the first JME(005) in the program with the same jump number. CJPN(511) and JME(005) are used in pairs. 	Output Not required	141
MULTIPLE JUMP JMP0 515	JMP0(515)	When the execution condition for JMP0(515) is OFF, all instructions from JMP0(515) to the next JME0(516) in the program are processed as NOP(000). Use JMP0(515) and JME0(516) in pairs. There is no limit on the number of pairs that can be used in the program. 	Output Required	145
MULTIPLE JUMP END JME0 516	JME0(516)	When the execution condition for JMP0(515) is OFF, all instructions from JMP0(515) to the next JME0(516) in the program are processed as NOP(000). Use JMP0(515) and JME0(516) in pairs. There is no limit on the number of pairs that can be used in the program.	Output Not required	145

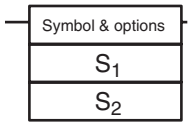
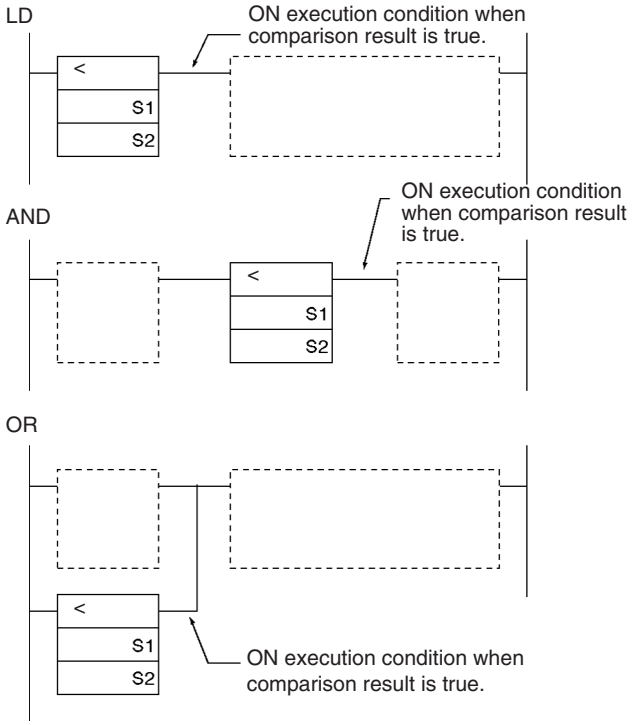
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
FOR-NEXT LOOPS FOR 512	FOR(512) N **N**: Number of loops	The instructions between FOR(512) and NEXT(513) are repeated a specified number of times. FOR(512) and NEXT(513) are used in pairs. 	Output Not required	147
BREAK LOOP BREAK 514	BREAK(514)	Programmed in a FOR-NEXT loop to cancel the execution of the loop for a given execution condition. The remaining instructions in the loop are processed as NOP(000) instructions. 	Output Required	150
FOR-NEXT LOOPS NEXT 513	NEXT(513)	The instructions between FOR(512) and NEXT(513) are repeated a specified number of times. FOR(512) and NEXT(513) are used in pairs.	Output Not required	147

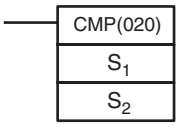
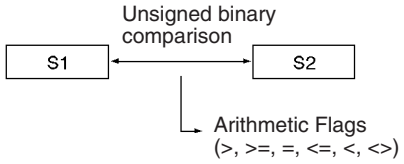
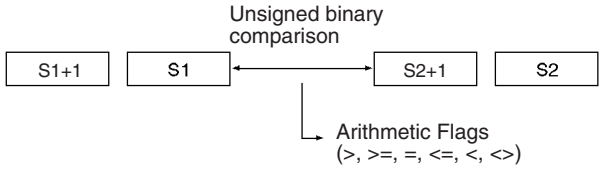
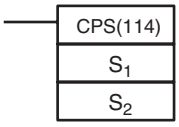
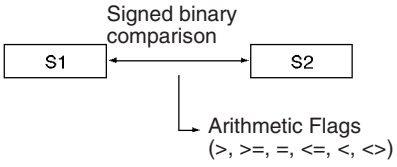
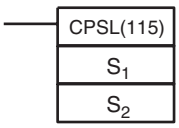
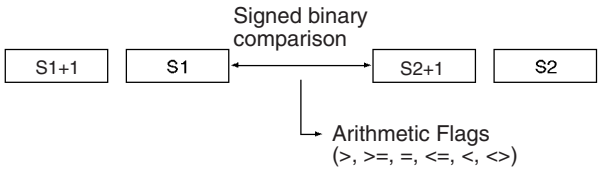
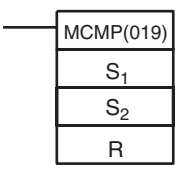
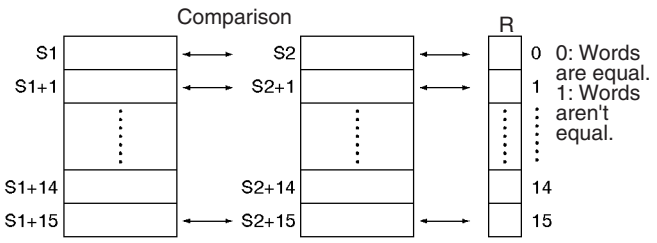
2-2-4 Timer and Counter Instructions

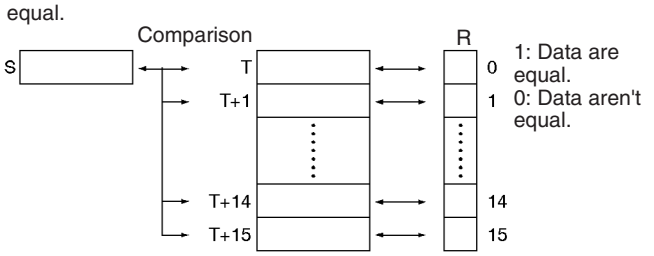
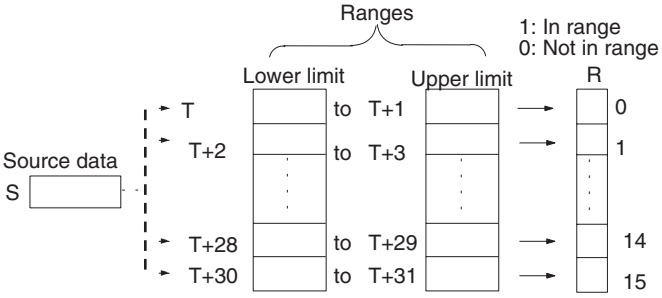
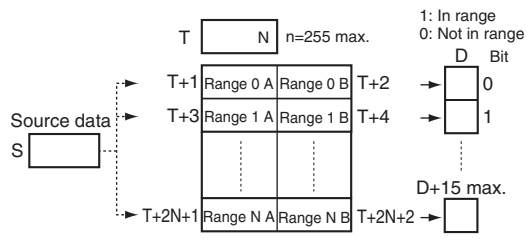
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
TIMER TIM (BCD)	TIM N S **N**: Timer number **S**: Set value	TIM operates a decrementing timer with units of 0.1-s. The setting range for the set value (SV) is 0 to 999.9 s (BCD).  When Timer Input Turns OFF before Completion Flag Turns ON 	Output Required	153
HIGH-SPEED TIMER TIMH 015 (BCD)	TIMH(015) N S **N**: Timer number **S**: Set value	TIMH(015) operates a decrementing timer with units of 10-ms. The setting range for the set value (SV) is 0 to 99.99 s (BCD).  When Timer Input Turns OFF before Completion Flag Turns ON 	Output Required	156
ONE-MS TIMER TMHH 540 (BCD)	TMHH(540) N S **N**: Timer number **S**: Set value	TMHH(540) operates a decrementing timer with units of 1-ms. The setting range for the set value (SV) is 0 to 9.999 s (BCD). The timing charts for TMHH(540) are the same as those given above for TIMH(015).	Output Required	158

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
COUNTER CNT (BCD)	Count input Reset input CNT N S **N**: Counter number **S**: Set value	CNT operates a decrementing counter. The setting range for the set value (SV) is 0 to 9,999 (BCD). 	Output Required	160
REVERSIBLE COUNTER CNTR 012 (BCD)	Increment input Decrement input Reset input CNTR(012) N S **N**: Counter number **S**: Set value	CNTR(012) operates a reversible counter. The counter is incremented on the increment input and decremented on the decrement input. 	Output Required	163

2-2-5 Comparison Instructions

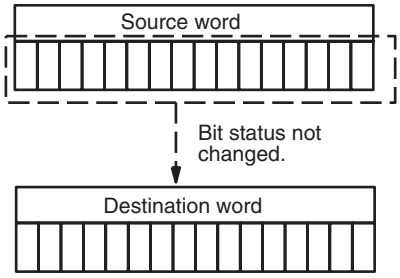
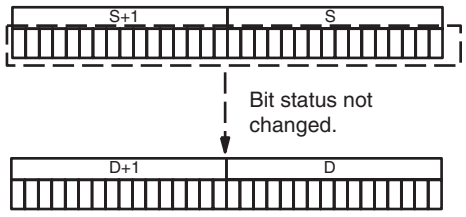
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
Symbol Comparison (Unsigned) LD, AND, OR + =, <>, <, <=, >, >= 300 (=) 305 (<>) 310 (<) 315 (<=) 320 (>) 325(>=)	 **S_1**: Comparison data 1 **S_2**: Comparison data 2	Symbol comparison instructions (unsigned) compare two values (constants and/or the contents of specified words) in 16-bit binary data and create an ON execution condition when the comparison condition is true. There are three types of symbol comparison instructions, LD (LOAD), AND, and OR. 	LD: Start of logic, Not required AND, OR: Continues on rung, Required	167
Symbol Comparison (Double-word, unsigned) LD, AND, OR + =, <>, <, <=, >, >= + L 301 (=) 306 (<>) 311 (<) 316 (<=) 321 (>) 326 (>=)	**S_1**: Comparison data 1 **S_2**: Comparison data 2	Symbol comparison instructions (double-word, unsigned) compare two values (constants and/or the contents of specified double-word data) in unsigned 32-bit binary data and create an ON execution condition when the comparison condition is true. There are three types of symbol comparison instructions, LD (LOAD), AND, and OR.	LD: Start of logic, Not required AND, OR: Continues on rung, Required	167
Symbol Comparison (Signed) LD, AND, OR + =, <>, <, <=, >, >= +S 302 (=) 307 (<>) 312 (<) 317 (<=) 322 (>) 327 (>=)	**S_1**: Comparison data 1 **S_2**: Comparison data 2	Symbol comparison instructions (signed) compare two values (constants and/or the contents of specified words) in signed 16-bit binary (4-digit hexadecimal) and create an ON execution condition when the comparison condition is true. There are three types of symbol comparison instructions, LD (LOAD), AND, and OR.	LD: Start of logic, Not required AND, OR: Continues on rung, Required	167

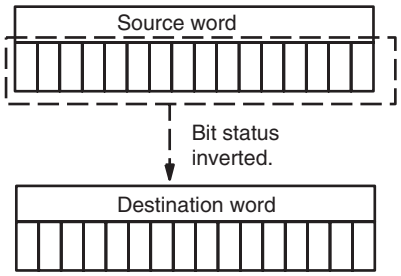
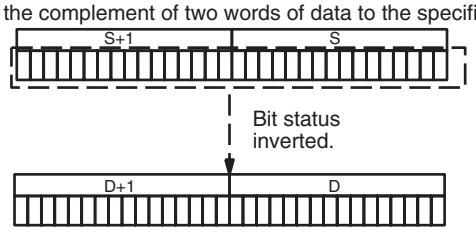
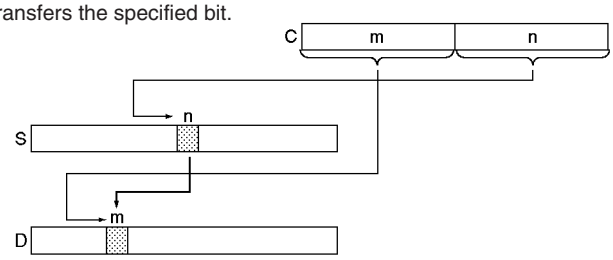
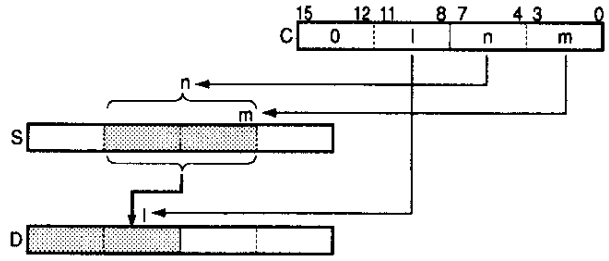
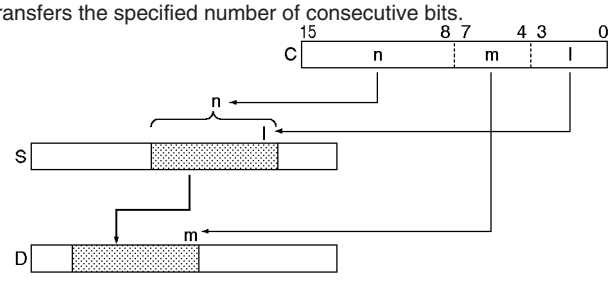
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
Symbol Comparison (Double-word, signed) LD, AND, OR + =, <>, <, <=, >, >= +SL 303 (=) 308 (<>) 313 (<) 318 (<=) 323 (>) 328 (>=)	**S_1**: Comparison data 1 **S_2**: Comparison data 2	Symbol comparison instructions (double-word, signed) compare two values (constants and/or the contents of specified double-word data) in signed 32-bit binary (8-digit hexadecimal) and create an ON execution condition when the comparison condition is true. There are three types of symbol comparison instructions, LD (LOAD), AND, and OR.	LD: Start of logic, Not required AND, OR: Continues on rung, Required	167
UNSIGNED COMPARE CMP 020	 **S_1**: Comparison data 1 **S_2**: Comparison data 2	Compares two unsigned binary values (constants and/or the contents of specified words) and outputs the result to the Arithmetic Flags in the Auxiliary Area. 	Output Required	172
DOUBLE UNSIGNED COMPARE CMPL 060	 **S_1**: Comparison data 1 **S_2**: Comparison data 2	Compares two double unsigned binary values (constants and/or the contents of specified words) and outputs the result to the Arithmetic Flags in the Auxiliary Area. 	Output Required	175
SIGNED BINARY COMPARE CPS 114	 **S_1**: Comparison data 1 **S_2**: Comparison data 2	Compares two signed binary values (constants and/or the contents of specified words) and outputs the result to the Arithmetic Flags in the Auxiliary Area. 	Output Required	177
DOUBLE SIGNED BINARY COMPARE CPSL 115	 **S_1**: Comparison data 1 **S_2**: Comparison data 2	Compares two double signed binary values (constants and/or the contents of specified words) and outputs the result to the Arithmetic Flags in the Auxiliary Area. 	Output Required	180
MULTIPLE COMPARE MCMP @MCMP 019	 **S_1**: 1st word of set 1 **S_2**: 1st word of set 2 **R**: Result word	Compares 16 consecutive words with another 16 consecutive words and turns ON the corresponding bit in the result word where the contents of the words **are not** equal. 	Output Required	182

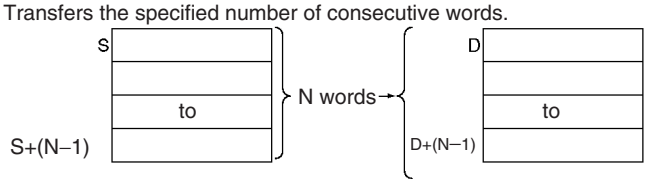
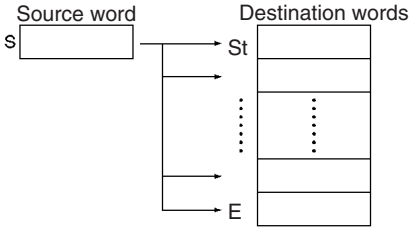
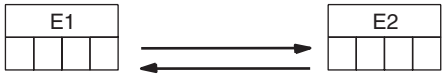
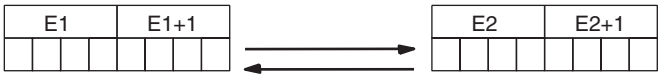
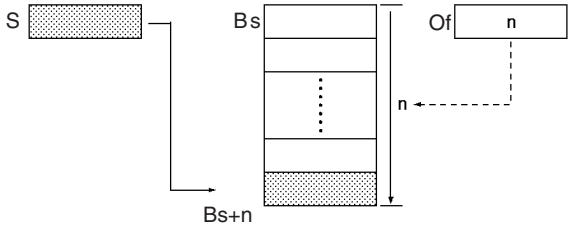
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
TABLE COMPARE TCMP @TCMP 085	TCMP(085) S T R **S**: Source data **T**: 1st word of table **R**: Result word	Compares the source data to the contents of 16 words and turns ON the corresponding bit in the result word when the contents are equal. 	Output Required	185
UNSIGNED BLOCK COMPARE BCMP @BCMP 068	BCMP(068) S T R **S**: Source data **T**: 1st word of table **R**: Result word	Compares the source data to 16 ranges (defined by 16 lower limits and 16 upper limits) and turns ON the corresponding bit in the result word when the source data is within the range. 	Output Required	187
EXPANDED BLOCK COMPARE BCMP2 @BCMP2 502	BCMP2(502) S T R **S**: Source data **T**: 1st word of block **R**: Result word	Compares the source data to up to 256 ranges (defined by upper and lower limits) and turns ON the corresponding bit in the result word when the source data is within a range.  **Note:** A can be less than or equal to B or greater the B.	Output Required	190

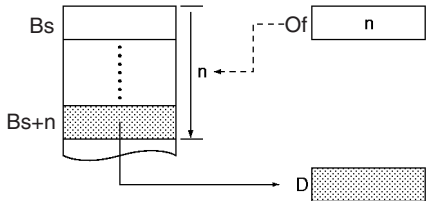
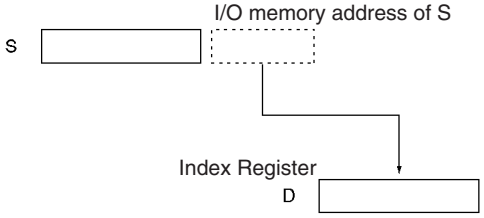
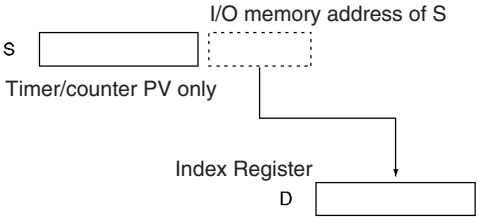
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
AREA RANGE COMPARE ZCP 088	ZCP(088) CD LL UL **CD**: Compare data (1 word) **LL**: Lower limit of range **UL**: Upper limit of range	Compares the 16-bit unsigned binary value in CD (word contents or constant) to the range defined by LL and UL and outputs the results to the Arithmetic Flags in the Auxiliary Area.	Output Required	193
DOUBLE AREA RANGE COMPARE ZCPL 116	ZCPL(116) CD LL UL **CD**: Compare data (2 words) **LL**: Lower limit of range **UL**: Upper limit of range	Compares the 32-bit unsigned binary value in CD and CD+1 (word contents or constant) to the range defined by LL and UL and outputs the results to the Arithmetic Flags in the Auxiliary Area.	Output Required	196

2-2-6 Data Movement Instructions

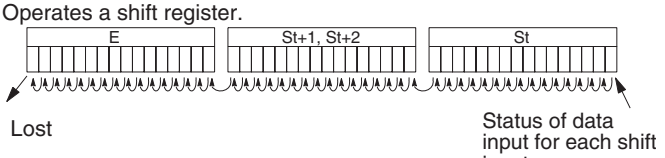
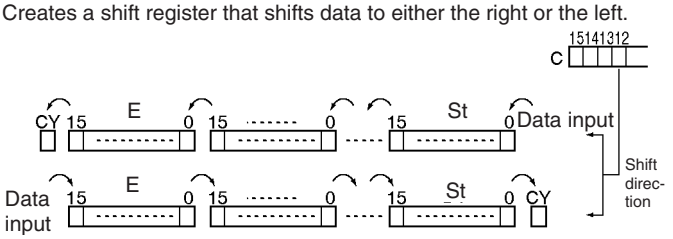
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
MOVE MOV @MOV 021	MOV(021) S D **S**: Source **D**: Destination	Transfers a word of data to the specified word. 	Output Required	199
DOUBLE MOVE MOVL @MOVL 498	MOVL(498) S D **S**: 1st source word **D**: 1st destination word	Transfers two words of data to the specified words. 	Output Required	201

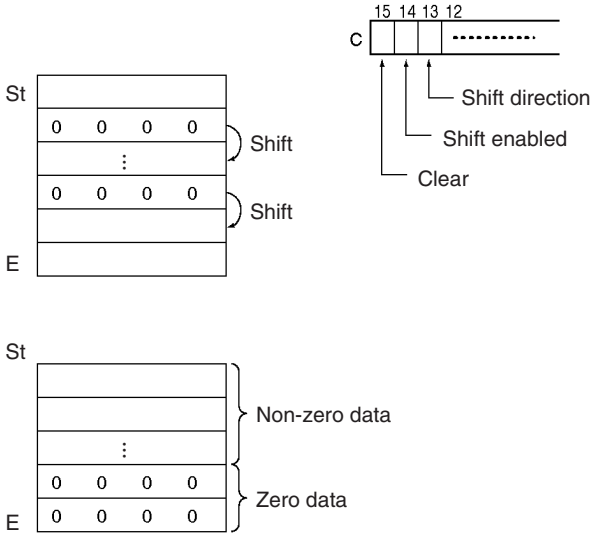
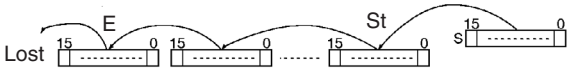
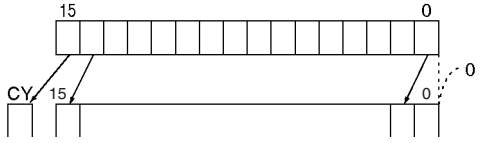
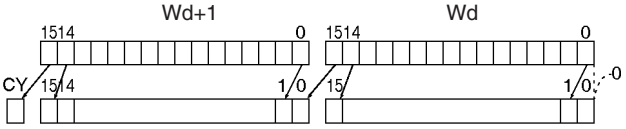
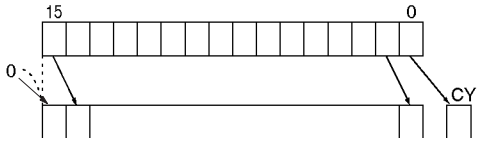
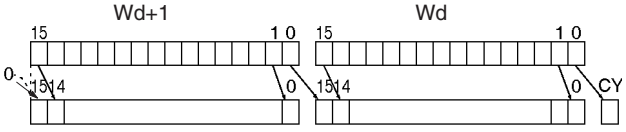
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
MOVE NOT MVN @MVN 022	MVN(022) S D **S**: Source **D**: Destination	Transfers the complement of a word of data to the specified word. 	Output Required	200
DOUBLE MOVE NOT MVNL @MVNL 499	MVNL(499) S D **S**: 1st source word **D**: 1st destination word	Transfers the complement of two words of data to the specified words. 	Output Required	203
MOVE BIT MOVB @MOVB 082	MOVB(082) S C D **S**: Source word or data **C**: Control word **D**: Destination word	Transfers the specified bit. 	Output Required	204
MOVE DIGIT MOVD @MOVD 083	MOVD(083) S C D **S**: Source word or data **C**: Control word **D**: Destination word	Transfers the specified digit or digits. (Each digit is made up of 4 bits.) 	Output Required	206
MULTIPLE BIT TRANSFER XFRB @XFRB 062	XFRB(062) C S D **C**: Control word **S**: 1st source word **D**: 1st destination word	Transfers the specified number of consecutive bits. 	Output Required	208

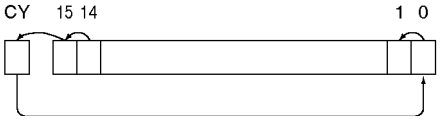
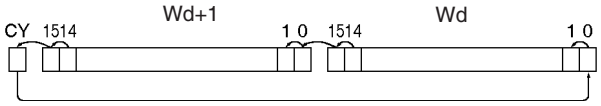
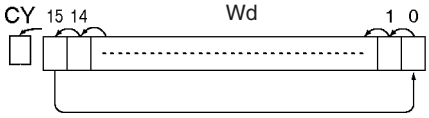
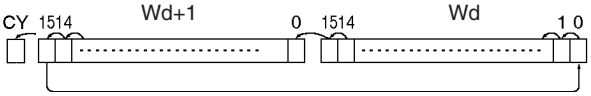
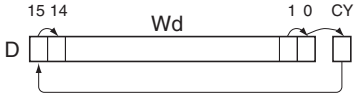
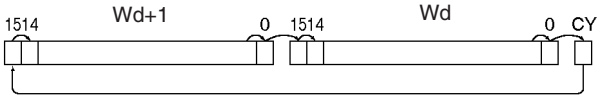
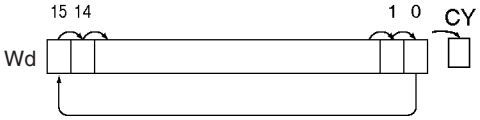
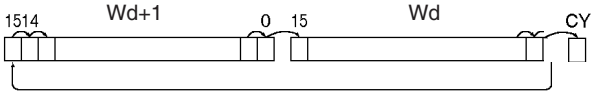
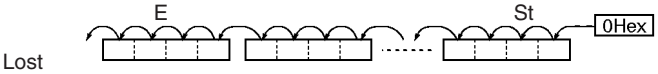
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
BLOCK TRANSFER XFER @XFER 070	XFER(070) N S D **N**: Number of words **S**: 1st source word **D**: 1st destination word	Transfers the specified number of consecutive words. 	Output Required	211
BLOCK SET BSET @BSET 071	BSET(071) S St E **S**: Source word **St**: Starting word **E**: End word	Copies the same word to a range of consecutive words. 	Output Required	213
DATA EXCHANGE XCHG @XCHG 073	XCHG(073) E1 E2 **E1**: 1st exchange word **E2**: Second exchange word	Exchanges the contents of the two specified words. 	Output Required	215
DOUBLE DATA EXCHANGE XCGL @XCGL 562	XCGL(562) E1 E2 **E1**: 1st exchange word **E2**: Second exchange word	Exchanges the contents of a pair of consecutive words with another pair of consecutive words. 	Output Required	216
SINGLE WORD DISTRIBUTE DIST @DIST 080	DIST(080) S Bs Of **S**: Source word **Bs**: Destination base address **Of**: Offset	Transfers the source word to a destination word calculated by adding an offset value to the base address. 	Output Required	217

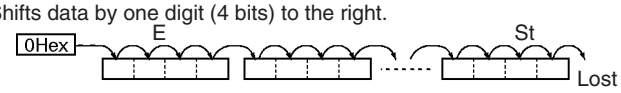
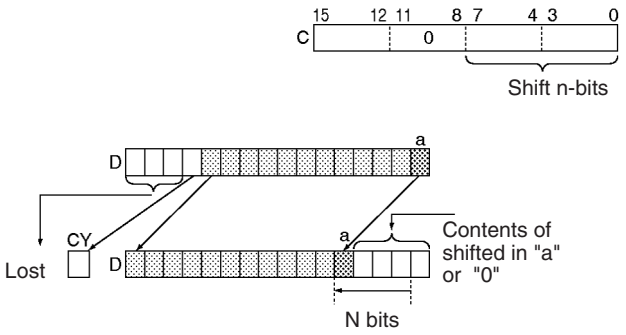
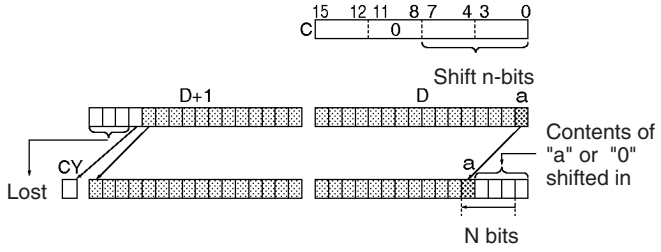
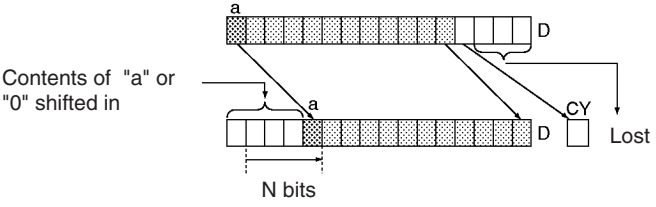
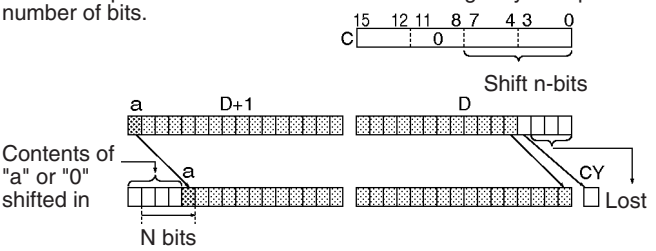
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DATA COLLECT COLL @COLL 081	COLL(081) Bs Of D **Bs**: Source base address **Of**: Offset **D**: Destination word	Transfers the source word (calculated by adding an offset value to the base address) to the destination word. 	Output Required	219
MOVE TO REGISTER MOVR @MOVR 560	MOVR(560) S D **S**: Source (desired word or bit) **D**: Destination (Index Register)	Sets the internal I/O memory address of the specified word, bit, or timer/counter Completion Flag in the specified Index Register. (Use MOVRW(561) to set the internal I/O memory address of a timer/counter PV in an Index Register.) 	Output Required	221
MOVE TIMER/ COUNTER PV TO REGISTER MOVRW @MOVRW 561	MOVRW(561) S D **S**: Source (desired TC number) **D**: Destination (Index Register)	Sets the internal I/O memory address of the specified timer or counter's PV in the specified Index Register. (Use MOVR(560) to set the internal I/O memory address of a word, bit, or timer/counter Completion Flag in an Index Register.) 	Output Required	222

2-2-7 Data Shift Instructions

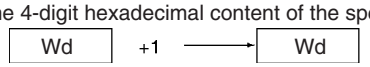
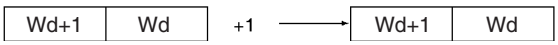
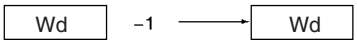
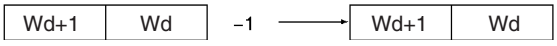
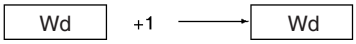
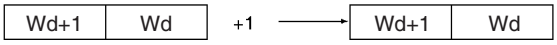
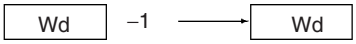
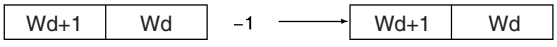
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SHIFT REGISTER SFT 010	Data input Shift input Reset input SFT(010) St E **St**: Starting word **E**: End word	Operates a shift register. 	Output Required	225
REVERSIBLE SHIFT REGISTER SFTR @SFTR 084	SFTR(084) C St E **C**: Control word **St**: Starting word **E**: End word	Creates a shift register that shifts data to either the right or the left. 	Output Required	227

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
ASYNCHRONOUS SHIFT REGISTER ASFT @ASFT 017	ASFT(017) C St E **C**: Control word **St**: Starting word **E**: End word	Shifts all non-zero word data within the specified word range either towards St or toward E, replacing 0000Hex word data. 	Output Required	230
WORD SHIFT WSFT @WSFT 016	WSFT(016) S St E **S**: Source word **St**: Starting word **E**: End word	Shifts data between St and E in word units. 	Output Required	232
ARITHMETIC SHIFT LEFT ASL @ASL 025	ASL(025) Wd **Wd**: Word	Shifts the contents of Wd one bit to the left. 	Output Required	233
DOUBLE SHIFT LEFT ASLL @ASLL 570	ASLL(570) Wd **Wd**: Word	Shifts the contents of Wd and Wd +1 one bit to the left. 	Output Required	235
ARITHMETIC SHIFT RIGHT ASR @ASR 026	ASR(026) Wd **Wd**: Word	Shifts the contents of Wd one bit to the right. 	Output Required	236
DOUBLE SHIFT RIGHT ASRL @ASRL 571	ASRL(571) Wd **Wd**: Word	Shifts the contents of Wd and Wd +1 one bit to the right. 	Output Required	238

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
ROTATE LEFT ROL @ROL 027	ROL(027) Wd **Wd**: Word	Shifts all Wd bits one bit to the left including the Carry Flag (CY). 	Output Required	239
DOUBLE ROTATE LEFT ROLL @ROLL 572	ROLL(572) Wd **Wd**: Word	Shifts all Wd and Wd + 1 bits one bit to the left including the Carry Flag (CY). 	Output Required	241
ROTATE LEFT WITHOUT CARRY RLNC @RLNC 574	RLNC(574) Wd **Wd**: Word	Shifts all Wd bits one bit to the left not including the Carry Flag (CY). 	Output Required	245
DOUBLE ROTATE LEFT WITHOUT CARRY RLNL @RLNL 576	RLNL(576) Wd **Wd**: Word	Shifts all Wd and Wd +1 bits one bit to the left not including the Carry Flag (CY). 	Output Required	247
ROTATE RIGHT ROR @ROR 028	ROR(028) Wd **Wd**: Word	Shifts all Wd bits one bit to the right including the Carry Flag (CY). 	Output Required	242
DOUBLE ROTATE RIGHT RORL @RORL 573	RORL(573) Wd **Wd**: Word	Shifts all Wd and Wd +1 bits one bit to the right including the Carry Flag (CY). 	Output Required	244
ROTATE RIGHT WITHOUT CARRY RRNC @RRNC 575	RRNC(575) Wd **Wd**: Word	Shifts all Wd bits one bit to the right not including the Carry Flag (CY). The contents of the rightmost bit of Wd shifts to the leftmost bit and to the Carry Flag (CY). 	Output Required	248
DOUBLE ROTATE RIGHT WITHOUT CARRY RRNL @RRNL 577	RRNL(577) Wd **Wd**: Word	Shifts all Wd and Wd +1 bits one bit to the right not including the Carry Flag (CY). The contents of the rightmost bit of Wd +1 is shifted to the leftmost bit of Wd, and to the Carry Flag (CY). 	Output Required	250
ONE DIGIT SHIFT LEFT SLD @SLD 074	SLD(074) St E **St**: Starting word **E**: End word	Shifts data by one digit (4 bits) to the left. 	Output Required	251

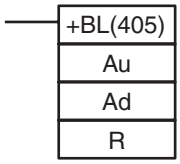
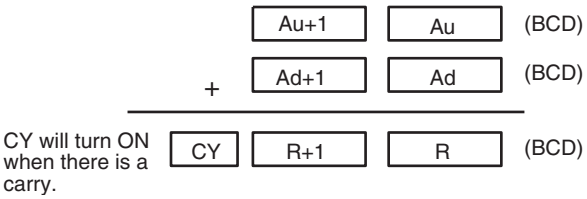
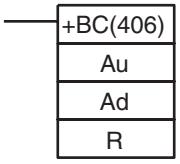
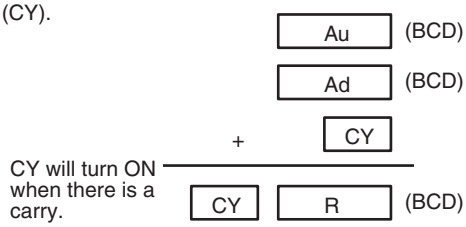
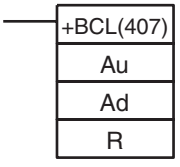
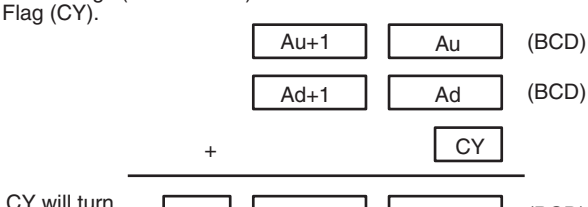
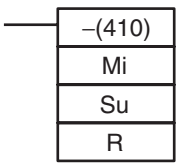
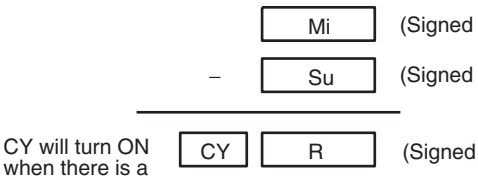
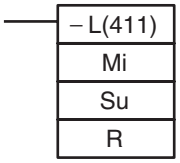
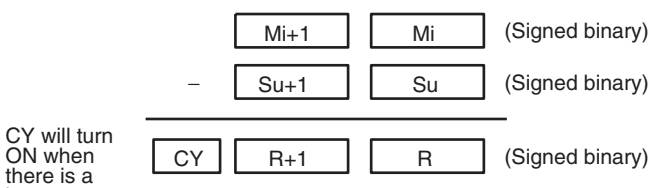
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
ONE DIGIT SHIFT RIGHT SRD @SRD 075	SRD(075) St E **St**: Starting word **E**: End word	Shifts data by one digit (4 bits) to the right. 	Output Required	253
SHIFT N-BITS LEFT NASL @NASL 580	NASL(580) D C **D**: Shift word **C**: Control word	Shifts the specified 16 bits of word data to the left by the specified number of bits. 	Output Required	254
DOUBLE SHIFT N-BITS LEFT NSLL @NSLL 582	NSLL(582) D C **D**: Shift word **C**: Control word	Shifts the specified 32 bits of word data to the left by the specified number of bits. 	Output Required	256
SHIFT N-BITS RIGHT NASR @NASR 581	NASR(581) D C **D**: Shift word **C**: Control word	Shifts the specified 16 bits of word data to the right by the specified number of bits. 	Output Required	259
DOUBLE SHIFT N-BITS RIGHT NSRL @NSRL 583	NSRL(583) D C **D**: Shift word **C**: Control word	Shifts the specified 32 bits of word data to the right by the specified number of bits. 	Output Required	261

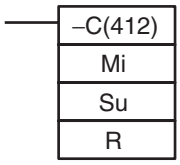
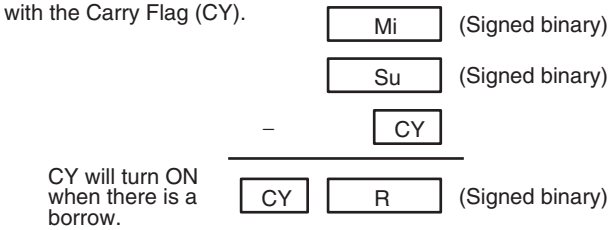
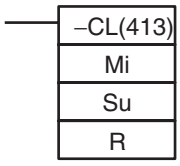
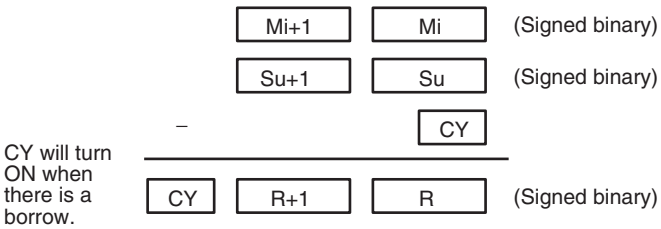
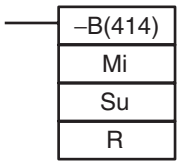
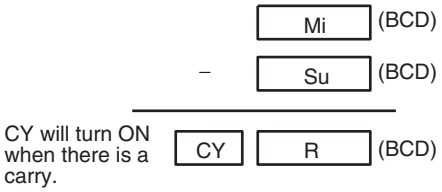
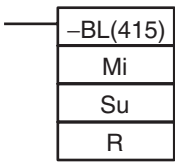
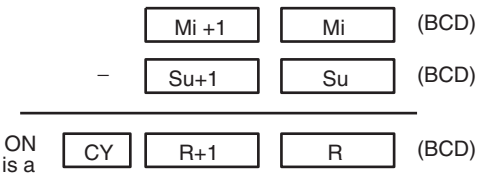
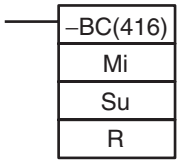
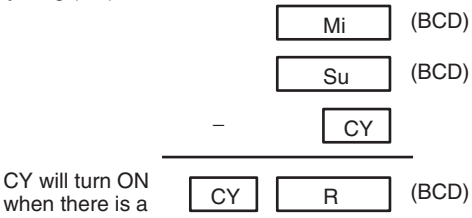
2-2-8 Increment/Decrement Instructions

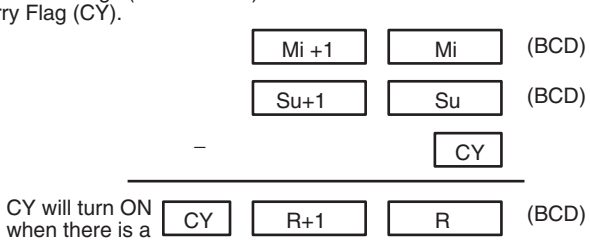
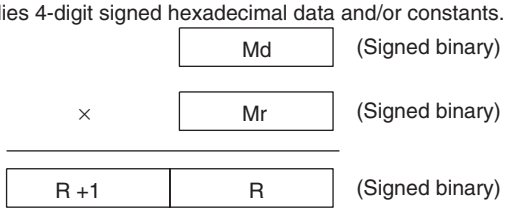
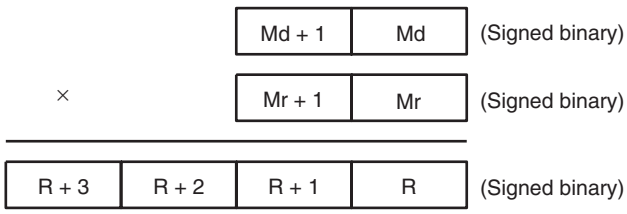
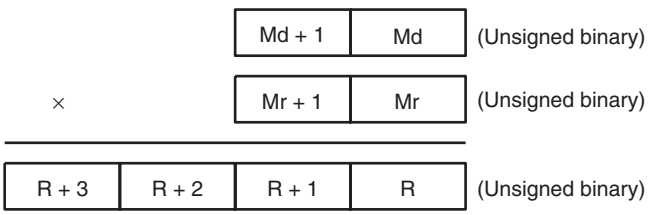
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
INCREMENT BINARY ++ @++ 590	++(590) Wd **Wd**: Word	Increments the 4-digit hexadecimal content of the specified word by 1. 	Output Required	265
DOUBLE INCREMENT BINARY ++L @++L 591	++L(591) Wd **Wd**: Word	Increments the 8-digit hexadecimal content of the specified words by 1. 	Output Required	267
DECREMENT BINARY – – @– – 592	– –(592) Wd **Wd**: Word	Decrements the 4-digit hexadecimal content of the specified word by 1. 	Output Required	269
DOUBLE DECREMENT BINARY – –L @– –L 593	– – L(593) Wd **Wd**: 1st word	Decrements the 8-digit hexadecimal content of the specified words by 1. 	Output Required	271
INCREMENT BCD ++B @++B 594	++B(594) Wd **Wd**: Word	Increments the 4-digit BCD content of the specified word by 1. 	Output Required	273
DOUBLE INCREMENT BCD ++BL @++BL 595	++BL(595) Wd **Wd**: 1st word	Increments the 8-digit BCD content of the specified words by 1. 	Output Required	275
DECREMENT BCD – –B @– –B 596	– –B(596) Wd **Wd**: Word	Decrements the 4-digit BCD content of the specified word by 1. 	Output Required	277
DOUBLE DECREMENT BCD – –BL @– –BL 597	– – BL(597) Wd **Wd**: 1st word	Decrements the 8-digit BCD content of the specified words by 1. 	Output Required	279

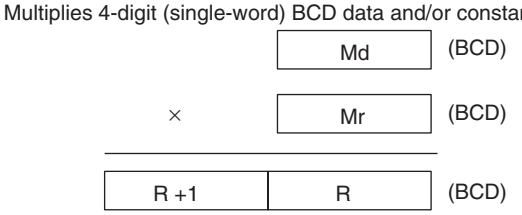
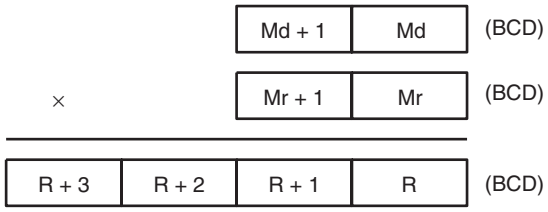
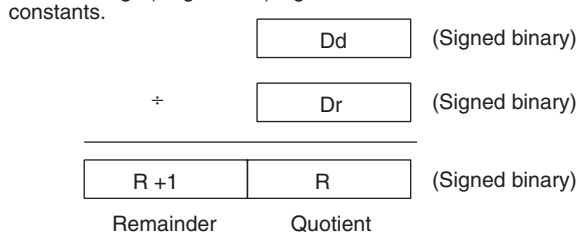
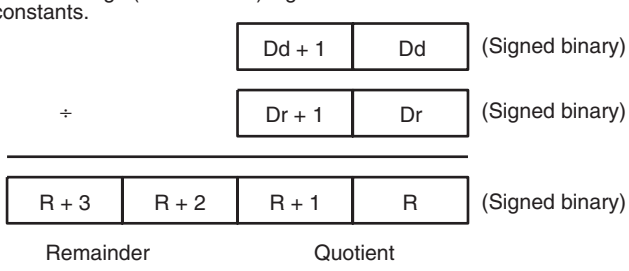
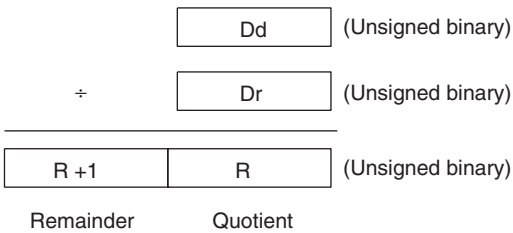
2-2-9 Symbol Math Instructions

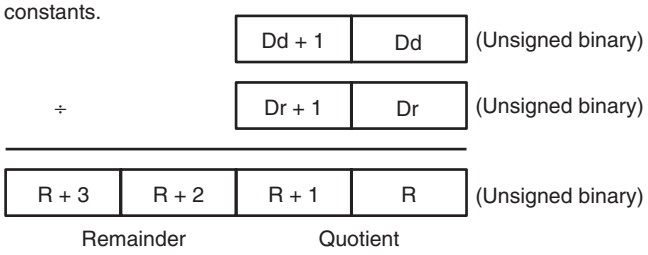
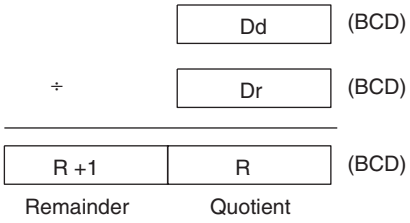
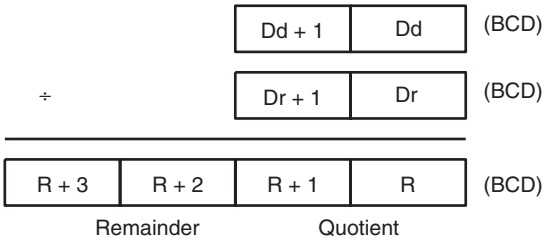
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SIGNED BINARY ADD WITHOUT CARRY + @+ 400	+(400) Au Ad R **Au**: Augend word **Ad**: Addend word **R**: Result word	Adds 4-digit (single-word) hexadecimal data and/or constants. Au (Signed binary) + Ad (Signed binary) CY will turn ON when there is a carry. CY R (Signed binary)	Output Required	282
DOUBLE SIGNED BINARY ADD WITHOUT CARRY +L @+L 401	+L(401) Au Ad R **Au**: 1st augend word **Ad**: 1st addend word **R**: 1st result word	Adds 8-digit (double-word) hexadecimal data and/or constants. Au+1 Au (Signed binary) + Ad+1 Ad (Signed binary) CY will turn ON when there is a carry. CY R+1 R (Signed binary)	Output Required	283
SIGNED BINARY ADD WITH CARRY +C @+C 402	+C(402) Au Ad R **Au**: Augend word **Ad**: Addend word **R**: Result word	Adds 4-digit (single-word) hexadecimal data and/or constants with the Carry Flag (CY). Au (Signed binary) Ad (Signed binary) + CY CY will turn ON when there is a carry. CY R (Signed binary)	Output Required	285
DOUBLE SIGNED BINARY ADD WITH CARRY +CL @+CL 403	+CL(403) Au Ad R **Au**: 1st augend word **Ad**: 1st addend word **R**: 1st result word	Adds 8-digit (double-word) hexadecimal data and/or constants with the Carry Flag (CY). Au+1 Au (Signed binary) Ad+1 Ad (Signed binary) + CY CY will turn ON when there is a carry. CY R+1 R (Signed binary)	Output Required	287
BCD ADD WITHOUT CARRY +B @+B 404	+B(404) Au Ad R **Au**: Augend word **Ad**: Addend word **R**: Result word	Adds 4-digit (single-word) BCD data and/or constants. Au (BCD) + Ad (BCD) CY will turn ON when there is a carry. CY R (BCD)	Output Required	289

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DOUBLE BCD ADD WITHOUT CARRY +BL @+BL 405	 **Au**: 1st augend word **Ad**: 1st addend word **R**: 1st result word	Adds 8-digit (double-word) BCD data and/or constants. 	Output Required	290
BCD ADD WITH CARRY +BC @+BC 406	 **Au**: Augend word **Ad**: Addend word **R**: Result word	Adds 4-digit (single-word) BCD data and/or constants with the Carry Flag (CY). 	Output Required	292
DOUBLE BCD ADD WITH CARRY +BCL @+BCL 407	 **Au**: 1st augend word **Ad**: 1st addend word **R**: 1st result word	Adds 8-digit (double-word) BCD data and/or constants with the Carry Flag (CY). 	Output Required	293
SIGNED BINARY SUBTRACT WITHOUT CARRY – @– 410	 **Mi**: Minuend word **Su**: Subtrahend word **R**: Result word	Subtracts 4-digit (single-word) hexadecimal data and/or constants. 	Output Required	295
DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY –L @–L 411	 **Mi**: Minuend word **Su**: Subtrahend word **R**: Result word	Subtracts 8-digit (double-word) hexadecimal data and/or constants. 	Output Required	296

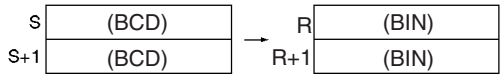
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SIGNED BINARY SUBTRACT WITH CARRY −C @−C 412	 **Mi**: Minuend word **Su**: Subtrahend word **R**: Result word	Subtracts 4-digit (single-word) hexadecimal data and/or constants with the Carry Flag (CY).  CY will turn ON when there is a borrow.	Output Required	300
DOUBLE SIGNED BINARY WITH CARRY −CL @−CL 413	 **Mi**: Minuend word **Su**: Subtrahend word **R**: Result word	Subtracts 8-digit (double-word) hexadecimal data and/or constants with the Carry Flag (CY).  CY will turn ON when there is a borrow.	Output Required	302
BCD SUBTRACT WITHOUT CARRY −B @−B 414	 **Mi**: Minuend word **Su**: Subtrahend word **R**: Result word	Subtracts 4-digit (single-word) BCD data and/or constants.  CY will turn ON when there is a carry.	Output Required	304
DOUBLE BCD SUBTRACT WITHOUT CARRY −BL @−BL 415	 **Mi**: 1st minuend word **Su**: 1st subtrahend word **R**: 1st result word	Subtracts 8-digit (double-word) BCD data and/or constants.  CY will turn ON when there is a borrow.	Output Required	306
BCD SUBTRACT WITH CARRY −BC @−BC 416	 **Mi**: Minuend word **Su**: Subtrahend word **R**: Result word	Subtracts 4-digit (single-word) BCD data and/or constants with the Carry Flag (CY).  CY will turn ON when there is a borrow.	Output Required	309

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DOUBLE BCD SUBTRACT WITH CARRY –BCL @–BCL 417	–BCL(417) Mi Su R **Mi**: 1st minuend word **Su**: 1st subtrahend word **R**: 1st result word	Subtracts 8-digit (double-word) BCD data and/or constants with the Carry Flag (CY). 	Output Required	310
SIGNED BINARY MULTIPLY * @* 420	*(420) Md Mr R **Md**: Multiplicand word **Mr**: Multiplier word **R**: Result word	Multiplies 4-digit signed hexadecimal data and/or constants. 	Output Required	312
DOUBLE SIGNED BINARY MULTIPLY *L @*L 421	*L(421) Md Mr R **Md**: 1st multiplicand word **Mr**: 1st multiplier word **R**: 1st result word	Multiplies 8-digit signed hexadecimal data and/or constants. 	Output Required	314
UNSIGNED BINARY MULTIPLY *U @*U 422	*U(422) Md Mr R **Md**: Multiplicand word **Mr**: Multiplier word **R**: Result word	Multiplies 4-digit unsigned hexadecimal data and/or constants. 	Output Required	315
DOUBLE UNSIGNED BINARY MULTIPLY *UL @*UL 423	*UL(423) Md Mr R **Md**: 1st multiplicand word **Mr**: 1st multiplier word **R**: 1st result word	Multiplies 8-digit unsigned hexadecimal data and/or constants. 	Output Required	317

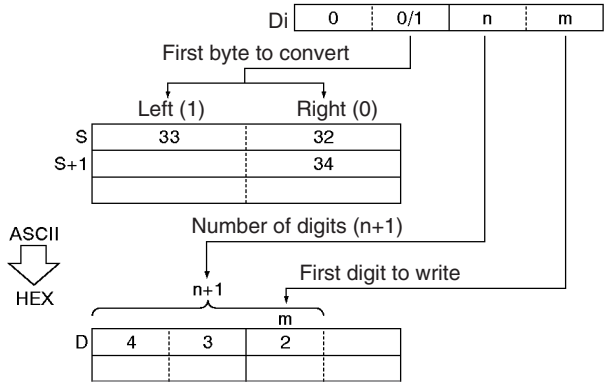
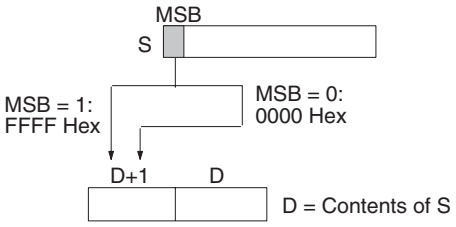
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
BCD MULTIPLY *B @*B 424	*B(424) Md Mr R **Md**: Multiplicand word **Mr**: Multiplier word **R**: Result word	Multiplies 4-digit (single-word) BCD data and/or constants. 	Output Required	318
DOUBLE BCD MULTIPLY *BL @*BL 425	*BL(425) Md Mr R **Md**: 1st multiplicand word **Mr**: 1st multiplier word **R**: 1st result word	Multiplies 8-digit (double-word) BCD data and/or constants. 	Output Required	320
SIGNED BINARY DIVIDE / @/ 430	/(430) Dd Dr R **Dd**: Dividend word **Dr**: Divisor word **R**: Result word	Divides 4-digit (single-word) signed hexadecimal data and/or constants. 	Output Required	321
DOUBLE SIGNED BINARY DIVIDE /L @/L 431	/L(431) Dd Dr R **Dd**: 1st dividend word **Dr**: 1st divisor word **R**: 1st result word	Divides 8-digit (double-word) signed hexadecimal data and/or constants. 	Output Required	323
UNSIGNED BINARY DIVIDE /U @/U 432	/U(432) Dd Dr R **Dd**: Dividend word **Dr**: Divisor word **R**: Result word	Divides 4-digit (single-word) unsigned hexadecimal data and/or constants. 	Output Required	324

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DOUBLE UNSIGNED BINARY DIVIDE /UL @/UL 433	/UL(433) Dd Dr R **Dd**: 1st dividend word **Dr**: 1st divisor word **R**: 1st result word	Divides 8-digit (double-word) unsigned hexadecimal data and/or constants. 	Output Required	326
BCD DIVIDE /B @/B 434	/B(434) Dd Dr R **Dd**: Dividend word **Dr**: Divisor word **R**: Result word	Divides 4-digit (single-word) BCD data and/or constants. 	Output Required	328
DOUBLE BCD DIVIDE /BL @/BL 435	/BL(435) Dd Dr R **Dd**: 1st dividend word **Dr**: 1st divisor word **R**: 1st result word	Divides 8-digit (double-word) BCD data and/or constants. 	Output Required	329

2-2-10 Conversion Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
BCD-TO-BINARY BIN @BIN 023	BIN(023) S R **S**: Source word **R**: Result word	Converts BCD data to binary data. 	Output Required	331
DOUBLE BCD-TO-DOUBLE BINARY BINL @BINL 058	BINL(058) S R **S**: 1st source word **R**: 1st result word	Converts 8-digit BCD data to 8-digit hexadecimal (32-bit binary) data. 	Output Required	333

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
BINARY-TO-BCD BCD @BCD 024	BCD(024) S R **S**: Source word **R**: Result word	Converts a word of binary data to a word of BCD data. S (BIN) → R (BCD)	Output Required	334
DOUBLE BINARY-TO-DOUBLE BCD BCDL @BCDL 059	BCDL(059) S R **S**: 1st source word **R**: 1st result word	Converts 8-digit hexadecimal (32-bit binary) data to 8-digit BCD data. S (BIN), S+1 (BIN) → R (BCD), R+1 (BCD)	Output Required	336
2'S COMPLEMENT NEG @NEG 160	NEG(160) S R **S**: Source word **R**: Result word	Calculates the 2's complement of a word of hexadecimal data. 2's complement (Complement + 1) $\overline{(S)}$ → (R)	Output Required	338
DOUBLE 2'S COMPLEMENT NEGL @NEGL 161	NEGL(161) S R **S**: 1st source word **R**: 1st result word	Calculates the 2's complement of two words of hexadecimal data. 2's complement (Complement + 1) $\overline{(S+1, S)}$ → (R+1, R)	Output Required	339
ASCII CONVERT ASC @ASC 086	ASC(086) S Di D **S**: Source word **Di**: Digit designator **D**: 1st destination word	Converts 4-bit hexadecimal digits in the source word into their 8-bit ASCII equivalents. Di: 15 12 11 8 7 4 3 0 — 0 | I/0 | n | m First digit to convert m S: 1 2 3 HEX → ASCII Number of digits (n+1) Left (1) Right (0) D: 33 / 31 32	Output Required	341

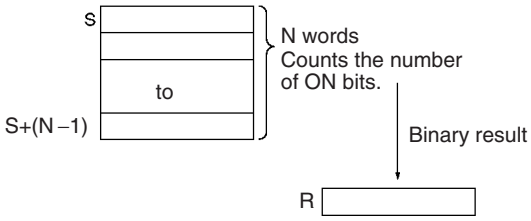
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
ASCII TO HEX HEX @HEX 162	HEX(162) S Di D **S**: 1st source word **Di**: Digit designator **D**: Destination word	Converts up to 4 bytes of ASCII data in the source word to their hexadecimal equivalents and writes these digits in the specified destination word. 	Output Required	345
16-BIT TO 32-BIT SIGNED BINARY SIGN @SIGN 600	SIGN(600) S R **S**: Source word **R**: 1st result word	Expands a 16-bit signed binary value to its 32-bit equivalent. 	Output Required	349

2-2-11 Logic Instructions

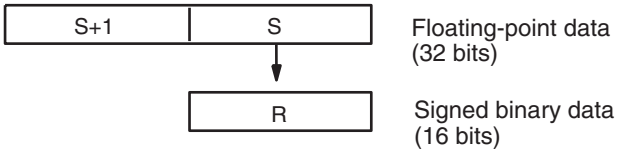
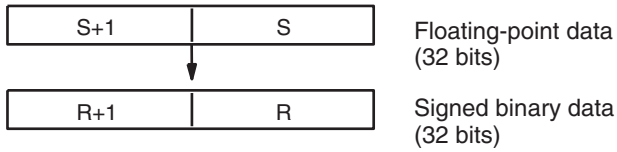
<table>
<tr><th>Instruction
Mnemonic
Code</th><th>Symbol/Operand</th><th>Function</th><th>Location
Execution condition</th><th>Page</th></tr>
<tr><td>LOGICAL AND
ANDW
@ANDW
034</td><td>ANDW(034)
I_1
I_2
R
I_1: Input 1
I_2: Input 2
R: Result word</td><td>Takes the logical AND of corresponding bits in single words of word data and/or constants.
$I_1 . I_2 \rightarrow R$
<table><tr><th>I_1</th><th>I_2</th><th>R</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table></td><td>Output
Required</td><td>351</td></tr>
<tr><td>DOUBLE
LOGICAL AND
ANDL
@ANDL
610</td><td>ANDL(610)
I_1
I_2
R
I_1: Input 1
I_2: Input 2
R: Result word</td><td>Takes the logical AND of corresponding bits in double words of word data and/or constants.
$(I_1, I_1+1) . (I_2, I_2+1) \rightarrow (R, R+1)$
<table><tr><th>I_1, I_1+1</th><th>I_2, I_2+1</th><th>R, R+1</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table></td><td>Output
Required</td><td>353</td></tr>
<tr><td>LOGICAL OR
ORW
@ORW
035</td><td>ORW(035)
I_1
I_2
R
I_1: Input 1
I_2: Input 2
R: Result word</td><td>Takes the logical OR of corresponding bits in single words of word data and/or constants.
$I_1 + I_2 \rightarrow R$
<table><tr><th>I_1</th><th>I_2</th><th>R</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table></td><td>Output
Required</td><td>354</td></tr>
<tr><td>DOUBLE
LOGICAL OR
ORWL
@ORWL
611</td><td>ORWL(611)
I_1
I_2
R
I_1: Input 1
I_2: Input 2
R: Result word</td><td>Takes the logical OR of corresponding bits in double words of word data and/or constants.
$(I_1, I_1+1) + (I_2, I_2+1) \rightarrow (R, R+1)$
<table><tr><th>I_1, I_1+1</th><th>I_2, I_2+1</th><th>R, R+1</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table></td><td>Output
Required</td><td>356</td></tr>
<tr><td>EXCLUSIVE OR
XORW
@XORW
036</td><td>XORW(036)
I_1
I_2
R
I_1: Input 1
I_2: Input 2
R: Result word</td><td>Takes the logical exclusive OR of corresponding bits in single words of word data and/or constants.
$I_1 . \overline{I_2} + \overline{I_1} . I_2 \rightarrow R$
<table><tr><th>I_1</th><th>I_2</th><th>R</th></tr><tr><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table></td><td>Output
Required</td><td>358</td></tr>
</table>

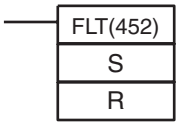
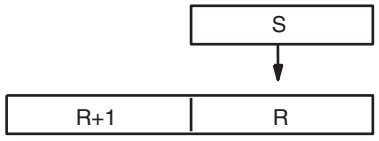
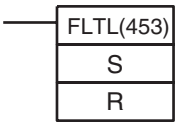
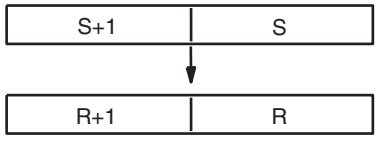
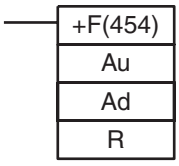
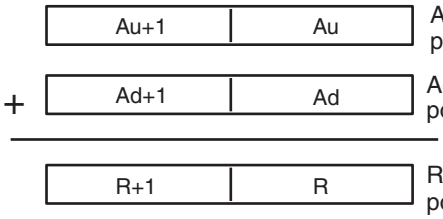
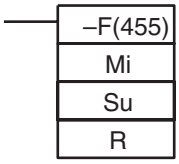
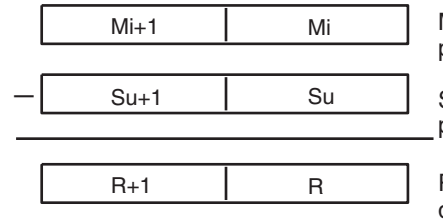
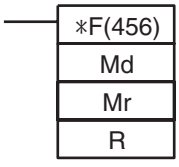
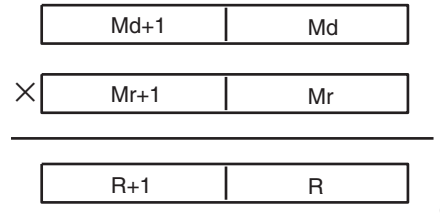
<table>
<tr><th>Instruction
Mnemonic
Code</th><th>Symbol/Operand</th><th>Function</th><th>Location
Execution condition</th><th>Page</th></tr>
<tr><td>DOUBLE EXCLUSIVE OR
XORL
@XORL
612</td><td>XORL(612)
I_1
I_2
R
I_1: Input 1
I_2: Input 2
R: Result word</td><td>Takes the logical exclusive OR of corresponding bits in double words of word data and/or constants.
$(I_1,I_1+1).\ (\overline{I_2,I_2+1}) + (\overline{I_1,I_1+1}).\ (I_2,I_2+1) \rightarrow (R,R+1)$
<table><tr><th>I_1,I_1+1</th><th>I_2,I_2+1</th><th>R, R+1</th></tr><tr><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table></td><td>Output
Required</td><td>360</td></tr>
<tr><td>EXCLUSIVE NOR
XNRW
@XNRW
037</td><td>XNRW(037)
I_1
I_2
R
I_1: Input 1
I_2: Input 2
R: Result word</td><td>Takes the logical exclusive NOR of corresponding single words of word data and/or constants.
$I_1.I_2 + \overline{I_1}.\overline{I_2} \rightarrow R$
<table><tr><th>I_1</th><th>I_2</th><th>R</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td></tr></table></td><td>Output
Required</td><td>362</td></tr>
<tr><td>DOUBLE EXCLUSIVE NOR
XNRL
@XNRL
613</td><td>XNRL(613)
I_1
I_2
R
I_1: Input 1
I_2: Input 2
R: 1st result word</td><td>Takes the logical exclusive NOR of corresponding bits in double words of word data and/or constants.
$(I_1,I_1+1).\ (I_2,I_2+1) + (\overline{I_1,I_1+1}).\ (\overline{I_2,I_2+1}) \rightarrow (R,R+1)$
<table><tr><th>I_1,I_1+1</th><th>I_2,I_2+1</th><th>R, R+1</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td></tr></table></td><td>Output
Required</td><td>363</td></tr>
<tr><td>COMPLEMENT
COM
@COM
029</td><td>COM(029)
Wd
Wd: Word</td><td>Turns OFF all ON bits and turns ON all OFF bits in Wd.
$\overline{Wd} \rightarrow Wd$: 1 → 0 and 0 → 1</td><td>Output
Required</td><td>365</td></tr>
<tr><td>DOUBLE COMPLEMENT
COML
@COML
614</td><td>COML(614)
Wd
Wd: Word</td><td>Turns OFF all ON bits and turns ON all OFF bits in Wd and Wd+1
$\overline{(Wd+1, Wd)} \rightarrow (Wd+1, Wd)$</td><td>Output
Required</td><td>366</td></tr>
</table>

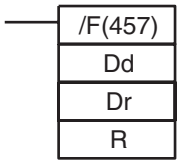
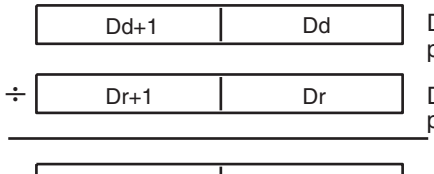
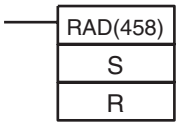
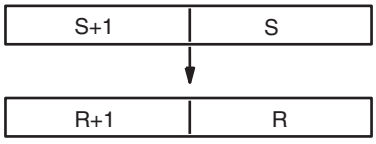
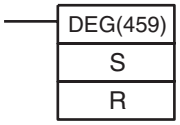
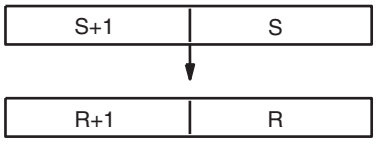
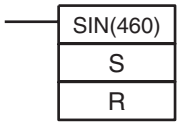
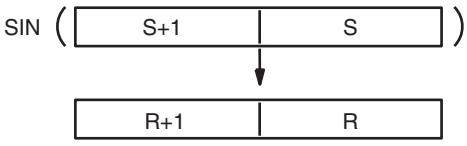
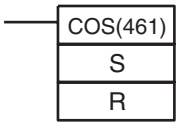
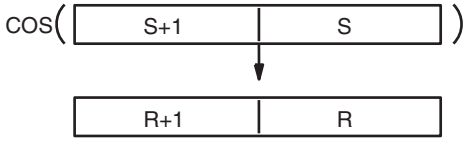
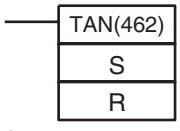
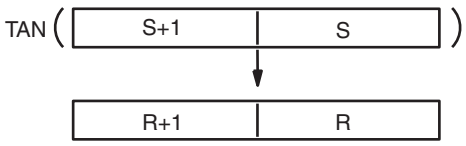
2-2-12 Special Math Instructions

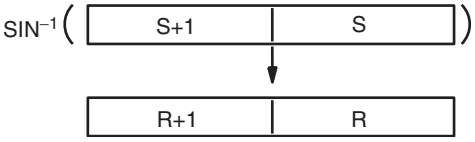
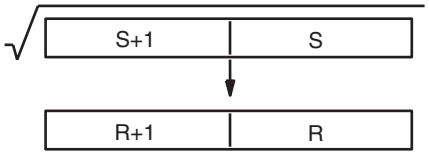
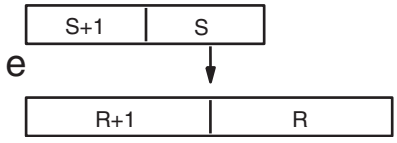
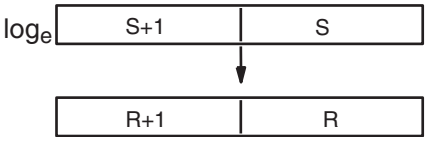
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
ARITHMETIC PROCESS APR @APR 069	APR(069) C S R **C**: Control word **S**: Source data **R**: Result word	Calculates the sine, cosine, or a linear extrapolation of the source data. The linear extrapolation function allows any relationship between X and Y to be approximated with line segments.	Output Required	368
BIT COUNTER BCNT @BCNT 067	BCNT(067) N S R **N**: Number of words **S**: 1st source word **R**: Result word	Counts the total number of ON bits in the specified word(s). 	Output Required	375
VIRTUAL AXIS AXIS 981	AXIS(981) M C T **M**: Mode designation **C**: Processing cycle **T**: 1st settings table word	Produces a virtual pulse output for trapezoidal acceleration/deceleration.	Output Required	376

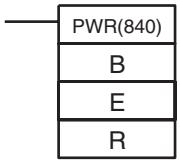
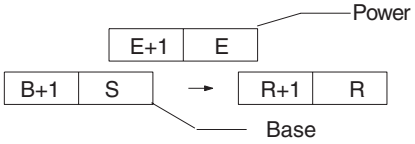
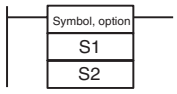
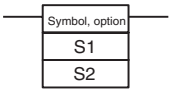
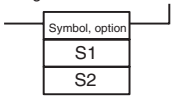
2-2-13 Floating-point Math Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
FLOATING TO 16-BIT FIX @FIX 450	FIX(450) S R **S**: 1st source word **R**: Result word	Converts a 32-bit floating-point value to 16-bit signed binary data and places the result in the specified result word. 	Output Required	386
FLOATING TO 32-BIT FIXL @FIXL 451	FIXL(451) S R **S**: 1st source word **R**: 1st result word	Converts a 32-bit floating-point value to 32-bit signed binary data and places the result in the specified result words. 	Output Required	388

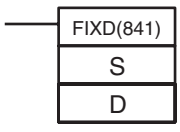
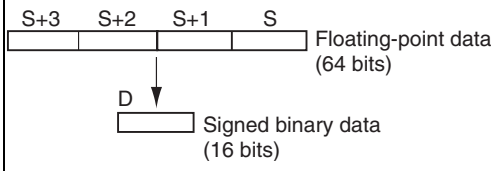
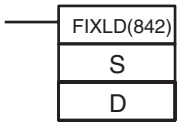
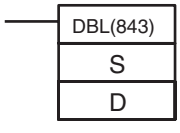
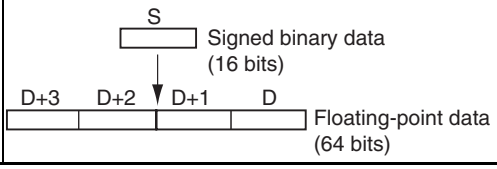
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
16-BIT TO FLOATING FLT @FLT 452	 **S**: Source word **R**: 1st result word	Converts a 16-bit signed binary value to 32-bit floating-point data and places the result in the specified result words.  Signed binary data (16 bits) Floating-point data (32 bits)	Output Required	389
32-BIT TO FLOATING FLTL @FLTL 453	 **S**: 1st source word **R**: 1st result word	Converts a 32-bit signed binary value to 32-bit floating-point data and places the result in the specified result words.  Signed binary data (32 bits) Floating-point data (32 bits)	Output Required	390
FLOATING-POINT ADD +F @+F 454	 **Au**: 1st augend word **AD**: 1st addend word **R**: 1st result word	Adds two 32-bit floating-point numbers and places the result in the specified result words.  Augend (floating-point data, 32 bits) Addend (floating-point data, 32 bits) Result (floating-point data, 32 bits)	Output Required	392
FLOATING-POINT SUB-TRACT −F @−F 455	 **Mi**: 1st Minuend word **Su**: 1st Subtrahend word **R**: 1st result word	Subtracts one 32-bit floating-point number from another and places the result in the specified result words.  Minuend (floating-point data, 32 bits) Subtrahend (floating-point data, 32 bits) Result (floating-point data, 32 bits)	Output Required	394
FLOATING-POINT MULTIPLY *F @*F 456	 **Md**: 1st Multiplicand word **Mr**: 1st Multiplier word **R**: 1st result word	Multiplies two 32-bit floating-point numbers and places the result in the specified result words.  Multiplicand (floating-point data, 32 bits) Multiplier (floating-point data, 32 bits) Result (floating-point data, 32 bits)	Output Required	396

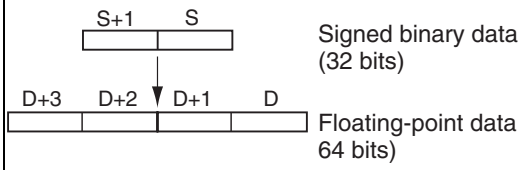
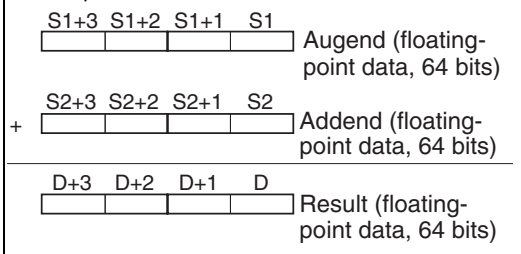
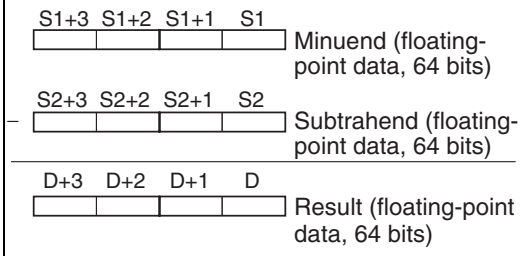
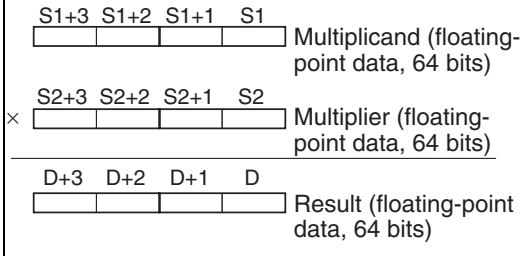
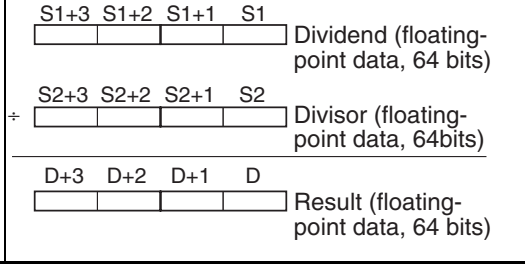
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
FLOATING-POINT DIVIDE /F @/F 457	 **Dd**: 1st Dividend word **Dr**: 1st Divisor word **R**: 1st result word	Divides one 32-bit floating-point number by another and places the result in the specified result words.  Dividend (floating-point data, 32 bits) Divisor (floating-point data, 32 bits) Result (floating-point data, 32 bits)	Output Required	397
DEGREES TO RADIANS RAD @RAD 458	 **S**: 1st source word **R**: 1st result word	Converts a 32-bit floating-point number from degrees to radians and places the result in the specified result words.  Source (degrees, 32-bit floating-point data) Result (radians, 32-bit floating-point data)	Output Required	400
RADIANS TO DEGREES DEG @DEG 459	 **S**: 1st source word **R**: 1st result word	Converts a 32-bit floating-point number from radians to degrees and places the result in the specified result words.  Source (radians, 32-bit floating-point data) Result (degrees, 32-bit floating-point data)	Output Required	401
SINE SIN @SIN 460	 **S**: 1st source word **R**: 1st result word	Calculates the sine of a 32-bit floating-point number (in radians) and places the result in the specified result words. 	Output Required	403
COSINE COS @COS 461	 **S**: 1st source word **R**: 1st result word	Calculates the cosine of a 32-bit floating-point number (in radians) and places the result in the specified result words. 	Output Required	404
TANGENT TAN @TAN 462	 **S**: 1st source word **R**: 1st result word	Calculates the tangent of a 32-bit floating-point number (in radians) and places the result in the specified result words. 	Output Required	406

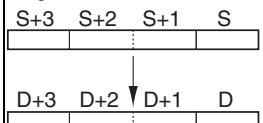
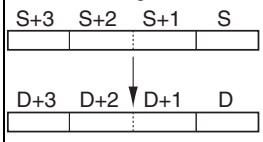
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
ARC SINE ASIN @ASIN 463	ASIN(463) S R **S**: 1st source word **R**: 1st result word	Calculates the arc sine of a 32-bit floating-point number and places the result in the specified result words. (The arc sine function is the inverse of the sine function; it returns the angle that produces a given sine value between −1 and 1.) 	Output Required	408
ARC COSINE ACOS @ACOS 464	ACOS(464) S R **S**: 1st source word **R**: 1st result word	Calculates the arc cosine of a 32-bit floating-point number and places the result in the specified result words. (The arc cosine function is the inverse of the cosine function; it returns the angle that produces a given cosine value between −1 and 1.) 	Output Required	410
ARC TANGENT ATAN @ATAN 465	ATAN(465) S R **S**: 1st source word **R**: 1st result word	Calculates the arc tangent of a 32-bit floating-point number and places the result in the specified result words. (The arc tangent function is the inverse of the tangent function; it returns the angle that produces a given tangent value.) 	Output Required	412
SQUARE ROOT SQRT @SQRT 466	SQRT(466) S R **S**: 1st source word **R**: 1st result word	Calculates the square root of a 32-bit floating-point number and places the result in the specified result words. 	Output Required	413
EXPONENT EXP @EXP 467	EXP(467) S R **S**: 1st source word **R**: 1st result word	Calculates the natural (base e) exponential of a 32-bit floating-point number and places the result in the specified result words. 	Output Required	415
LOGARITHM LOG @LOG 468	LOG(468) S R **S**: 1st source word **R**: 1st result word	Calculates the natural (base e) logarithm of a 32-bit floating-point number and places the result in the specified result words. 	Output Required	417

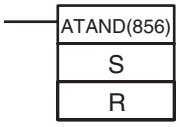
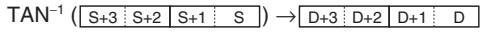
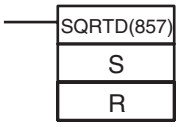
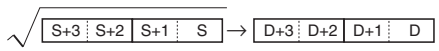
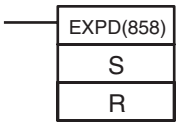
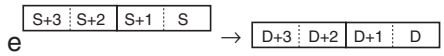
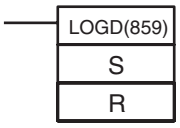
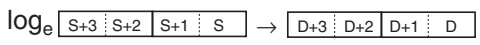
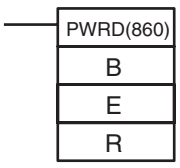
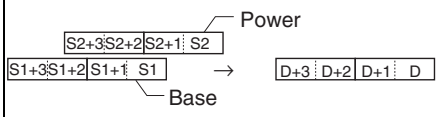
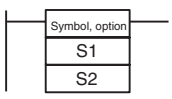
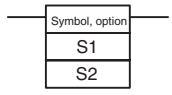
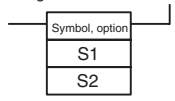
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
EXPONENTIAL POWER PWR @PWR 840	 **B**: 1st base word **E**: 1st exponent word **R**: 1st result word	Raises a 32-bit floating-point number to the power of another 32-bit floating-point number. 	Output Required	419
FLOATING SYMBOL COMPARISON LD, AND. or OR + =F (329), <>F (330), <F (331), <=F (332), >F (333), or >=F (334)	Using LD:  Using AND:  Using OR:  **S1**: Comparison data 1 **S2**: Comparison data 2	Compares the specified single-precision data (32 bits) or constants and creates an ON execution condition if the comparison result is true. Three kinds of symbols can be used with the floating-point symbol comparison instructions: LD (Load), AND, and OR.	LD: Start of logic, Not required AND or OR: Continues on rung, Required	421

2-2-14 Double-precision Floating-point Instructions

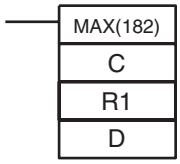
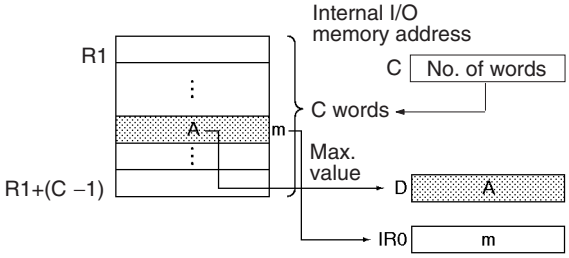
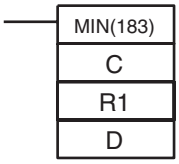
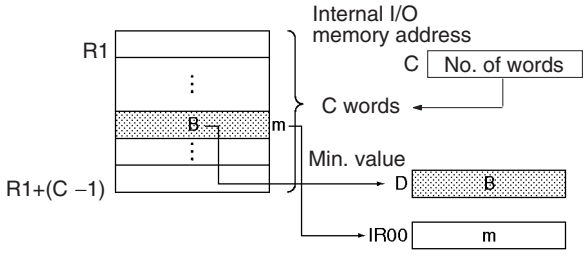
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DOUBLE FLOATING TO 16-BIT BINARY FIXD @FIXD 841	 **S:** 1st source word **D**: Destination word	Converts the specified double-precision floating-point data (64 bits) to 16-bit signed binary data and outputs the result to the destination word. 	Output Required	430
DOUBLE FLOATING TO 32-BIT BINARY FIXLD @FIXLD 842	 **S:** 1st source word **D**: 1st destination word	Converts the specified double-precision floating-point data (64 bits) to 32-bit signed binary data and outputs the result to the destination words. 	Output Required	432
16-BIT BINARY TO DOUBLE FLOATING DBL @DBL 843	 **S:** Source word **D**: 1st destination word	Converts the specified 16-bit signed binary data to double-precision floating-point data (64 bits) and outputs the result to the destination words. 	Output Required	433

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
32-BIT BINARY TO DOUBLE FLOATING DBLL @DBLL 844	DBLL(844) S D **S:** 1st source word **D**: 1st destination word	Converts the specified 32-bit signed binary data to double-precision floating-point data (64 bits) and outputs the result to the destination words. 	Output Required	434
DOUBLE FLOATING-POINT ADD +D @+D 845	+D(845) Au Ad R **Au:** 1st augend word **Ad**: 1st addend word **R**: 1st result word	Adds the specified double-precision floating-point values (64 bits each) and outputs the result to the result words. 	Output Required	436
DOUBLE FLOATING-POINT SUBTRACT −D @−D 846	−D(846) Mi Su R **Mi:** 1st minuend word **Su**: 1st subtrahend word **R**: 1st result word	Subtracts the specified double-precision floating-point values (64 bits each) and outputs the result to the result words. 	Output Required	437
DOUBLE FLOATING-POINT MULTIPLY *D @*D 847	*D(847) Md Mr R **Md:** 1st multiplicand word **Mr**: 1st multiplier word **R**: 1st result word	Multiplies the specified double-precision floating-point values (64 bits each) and outputs the result to the result words. 	Output Required	439
DOUBLE FLOATING-POINT DIVIDE /D @/D 848	/D(848) Dd Dr R **Dd:** 1st Dividend word **Dr**: 1st divisor word **R**: 1st result word	Divides the specified double-precision floating-point values (64 bits each) and outputs the result to the result words. 	Output Required	441

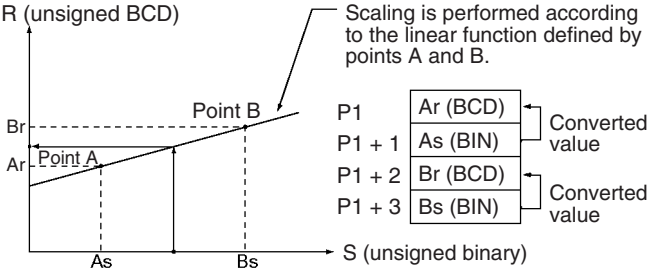
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DOUBLE DEGREES TO RADIANS RADD @RADD 849	RADD(849) S R **S:** 1st source word **R**: 1st result word	Converts the specified double-precision floating-point data (64 bits) from degrees to radians and outputs the result to the result words.  	Output Required	443
DOUBLE RADIANS TO DEGREES DEGD @DEGD 850	DEGD(850) S R **S:** 1st source word **R**: 1st result word	Converts the specified double-precision floating-point data (64 bits) from radians to degrees and outputs the result to the result words.  	Output Required	444
DOUBLE SINE SIND @SIND 851	SIND(851) S R **S:** 1st source word **R**: 1st result word	Calculates the sine of the angle (radians) in the specified double-precision floating-point data (64 bits) and outputs the result to the result words. SIN (S+3 S+2 S+1 S) → D+3 D+2 D+1 D	Output Required	446
DOUBLE COSINE COSD @COSD 852	COSD(852) S R **S:** 1st source word **R**: 1st result word	Calculates the cosine of the angle (radians) in the specified double-precision floating-point data (64 bits) and outputs the result to the result words. COS (S+3 S+2 S+1 S) → D+3 D+2 D+1 D	Output Required	447
DOUBLE TANGENT TAND @TAND 853	TAND(853) S R **S:** 1st source word **R**: 1st result word	Calculates the tangent of the angle (radians) in the specified double-precision floating-point data (64 bits) and outputs the result to the result words. TAN (S+3 S+2 S+1 S) → D+3 D+2 D+1 D	Output Required	449
DOUBLE ARC SINE ASIND @ASIND 854	ASIND(854) S R **S:** 1st source word **R**: 1st result word	Calculates the angle (in radians) from the sine value in the specified double-precision floating-point data (64 bits) and outputs the result to the result words. (The arc sine function is the inverse of the sine function; it returns the angle that produces a given sine value between -1 and 1.) SIN^{-1} (S+3 S+2 S+1 S) → D+3 D+2 D+1 D	Output Required	450
DOUBLE ARC COSINE ACOSD @ACOSD 855	ACOSD(855) S R **S:** 1st source word **R**: 1st result word	Calculates the angle (in radians) from the cosine value in the specified double-precision floating-point data (64 bits) and outputs the result to the result words. (The arc cosine function is the inverse of the cosine function; it returns the angle that produces a given cosine value between -1 and 1.) COS^{-1} (S+3 S+2 S+1 S) → D+3 D+2 D+1 D	Output Required	452

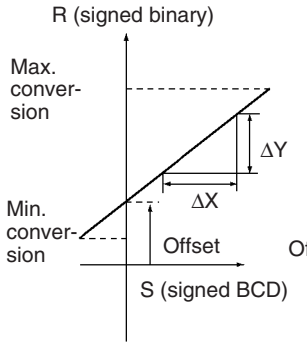
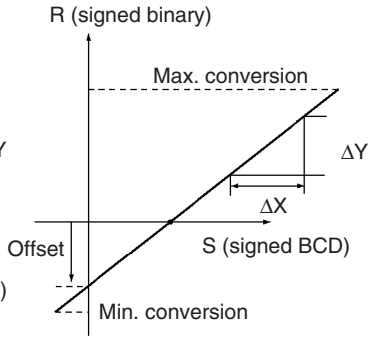
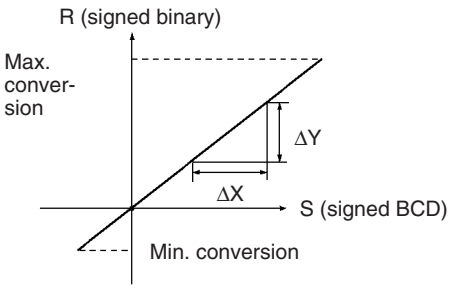
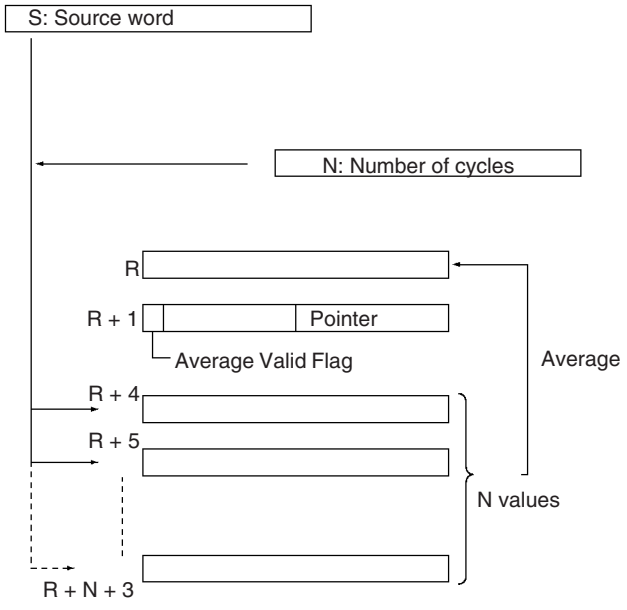
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DOUBLE ARC TANGENT ATAND @ATAND 856	 **S:** 1st source word **R**: 1st result word	Calculates the angle (in radians) from the tangent value in the specified double-precision floating-point data (64 bits) and outputs the result to the result words. (The arc tangent function is the inverse of the tangent function; it returns the angle that produces a given tangent value.) 	Output Required	454
DOUBLE SQUARE ROOT SQRTD @SQRTD 857	 **S:** 1st source word **R**: 1st result word	Calculates the square root of the specified double-precision floating-point data (64 bits) and outputs the result to the result words. 	Output Required	456
DOUBLE EXPONENT EXPD @EXPD 858	 **S:** 1st source word **R**: 1st result word	Calculates the natural (base e) exponential of the specified double-precision floating-point data (64 bits) and outputs the result to the result words. 	Output Required	457
DOUBLE LOGARITHM LOGD @LOGD 859	 **S:** 1st source word **R**: 1st result word	Calculates the natural (base e) logarithm of the specified double-precision floating-point data (64 bits) and outputs the result to the result words. 	Output Required	459
DOUBLE EXPONENTIAL POWER PWRD @PWRD 860	 **B:** 1st base word **E**: 1st exponent word **R**: 1st result word	Raises a double-precision floating-point number (64 bits) to the power of another double-precision floating-point number and outputs the result to the result words. 	Output Required	461
DOUBLE SYMBOL COMPARISON LD, AND. or OR + =D (335), <>D (336), <D (337), <=D (338), >D (339), or >=D (340)	Using LD:  Using AND:  Using OR:  **S1**: Comparison data 1 **S2**: Comparison data 2	Compares the specified double-precision data (64 bits) and creates an ON execution condition if the comparison result is true. Three kinds of symbols can be used with the floating-point symbol comparison instructions: LD (Load), AND, and OR.	LD: Not required AND or OR: Required	462

2-2-15 Table Data Processing Instructions

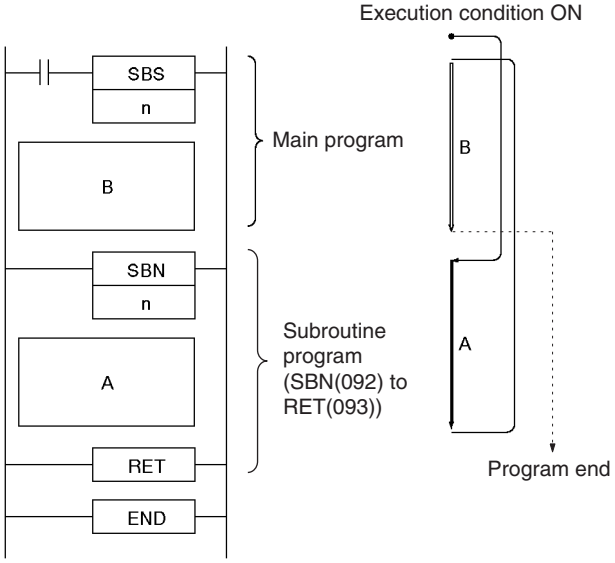
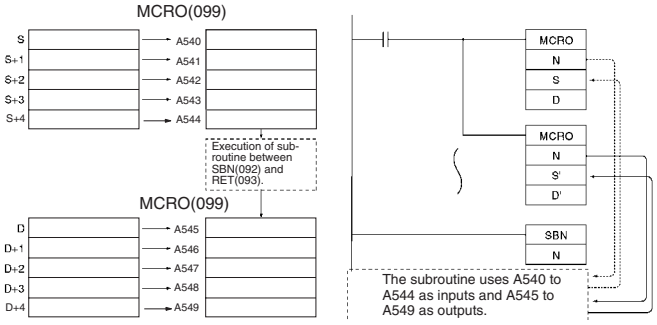
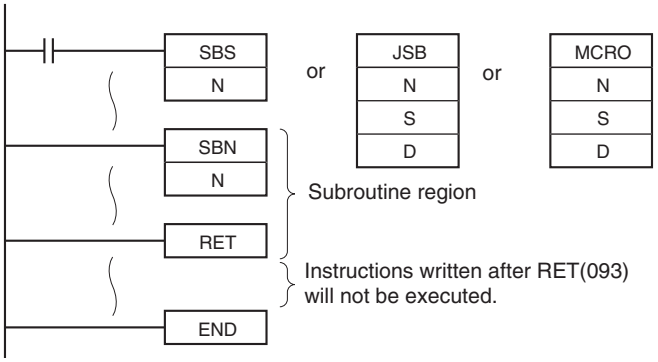
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
FIND MAXIMUM MAX @MAX 182	 **C**: 1st control word **R1**: 1st word in range **D**: Destination word	Finds the maximum value in the range. 	Output Required	467
FIND MINIMUM MIN @MIN 183	 **C**: 1st control word **R1**: 1st word in range **D**: Destination word	Finds the minimum value in the range. 	Output Required	471

2-2-16 Data Control Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SCALING SCL @SCL 194	SCL(194) S P1 R **S**: Source word **P1**: 1st parameter word **R**: Result word	Converts unsigned binary data into unsigned BCD data according to the specified linear function. 	Output Required	475
SCALING 2 SCL2 @SCL2 486	SCL2(486) S P1 R **S**: Source word **P1**: 1st parameter word **R**: Result word	Converts signed binary data into signed BCD data according to the specified linear function. An offset can be input in defining the linear function. **Positive Offset** R (signed BCD) Offset ΔY ΔX S (signed binary) **Negative Offset** R (signed BCD) ΔY ΔX Offset S (signed binary) P1 Offset (Signed binary) P1 + 1 ΔX (Signed binary) P1 + 2 ΔY (Signed BCD) **Offset of 0000** R (signed BCD) Offset = 0000 hex ΔY ΔX S (signed binary)	Output Required	479

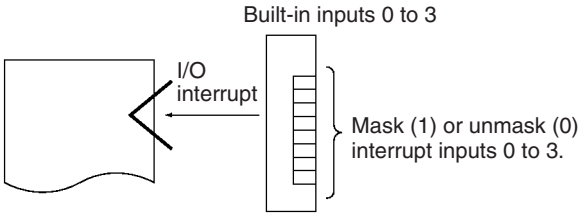
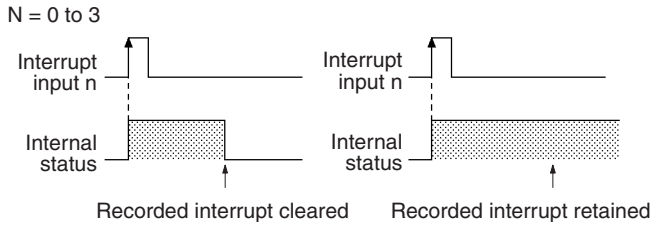
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SCALING 3 SCL3 @SCL3 487	SCL3(487) S P1 R **S**: Source word **P1**: 1st parameter word **R**: Result word	Converts signed BCD data into signed binary data according to the specified linear function. An offset can be input in defining the linear function. **Positive Offset**  **Negative Offset**  **Offset of 0000** 	Output Required	483
AVERAGE AVG 195	AVG(195) S N R **S**: Source word **N**: Number of cycles **R**: Result word	Calculates the average value of an input word for the specified number of cycles. 	Output Required	486

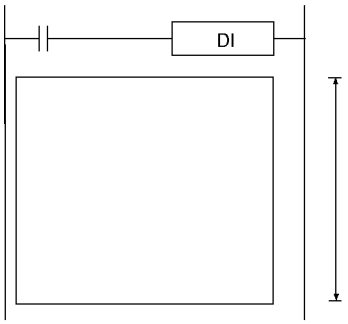
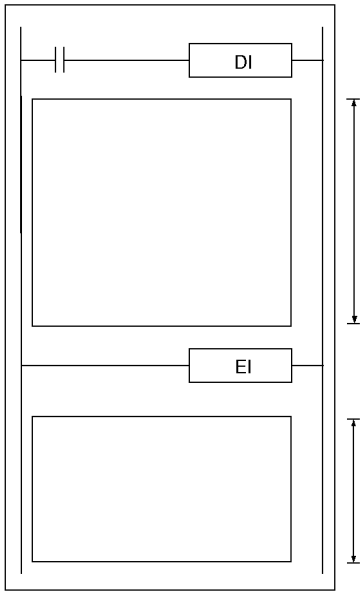
2-2-17 Subroutine Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SUBROUTINE CALL SBS @SBS 091	SBS(091) N **N**: Subroutine number	Calls the subroutine with the specified subroutine number and executes that program. 	Output Required	491
MACRO MCRO @MCRO 099	MCRO(099) N S D **N**: Subroutine number **S**: 1st input parameter word **D**: 1st output parameter word	Calls the subroutine with the specified subroutine number and executes that program using the input parameters in S to S+4 and the output parameters in D to D+4. 	Output Required	496
SUBROUTINE ENTRY SBN 092	SBN(092) N **N**: Subroutine number	Indicates the beginning of the subroutine program with the specified subroutine number. 	Output Not required	500

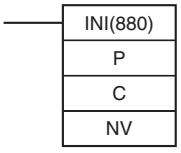
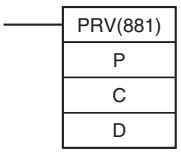
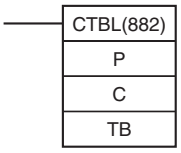
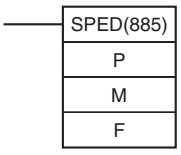
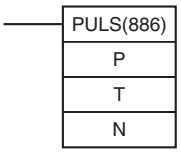
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SUBROUTINE RETURN RET 093	RET(093)	Indicates the end of a subroutine program.	Output Not required	503
JUMP TO SUB-ROUTINE JSB 982	JSB(982) N S D **N**: Subroutine number **S**: 1st input parameter word **D**: 1st output parameter word	Calls the subroutine with the specified subroutine number regardless of the status of the input condition and executes that program. Data beginning with the word specified in S is passed to the subroutine. After execution of the subroutine, the result data is passed to the area beginning with the word specified in D. The ON/OFF status of the input condition is set in an Auxiliary Area bit so that it can be referenced from within the subroutine program.	Output Required	503

2-2-18 Interrupt Control Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SET INTERRUPT MASK MSKS @MSKS 690	MSKS(690) N S **N**: Interrupt identifier **S**: Interrupt data	Sets up interrupt processing for I/O interrupts. Both I/O interrupt tasks are masked (disabled) when the FQM1 is first turned ON. MSKS(690) can be used to unmask or mask I/O interrupts. 	Output Required	508
READ INTERRUPT MASK MSKR @MSKR 692	MSKR(692) N D **N**: Interrupt identifier **D**: Destination word	Reads the current interrupt processing settings that were set with MSKS(690).	Output Required	510
CLEAR INTERRUPT CLI @CLI 691	CLI(691) N S **N**: Interrupt identifier **S**: Interrupt data	Clears or retains recorded interrupt inputs for I/O interrupts. 	Output Required	512

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
DISABLE INTERRUPTS DI @DI 693	DI(693)	Disables execution of all interrupt tasks except the power OFF interrupt.  	Output Required	513
ENABLE INTERRUPTS EI 694	EI(694)	Enables execution of all interrupt tasks that were disabled with DI(693).  	Output Not required	514
INTERVAL TIMER STIM @STIM 980	STIM(980) C1 C2 C3 C1: Control data #1 C2: Control data #2 C3: Control data #3	Controls interval timers and controls the pulse output ports. Interval timer control can be used to start one-shot or scheduled interrupt tasks.	Output Required	516

2-2-19 High-speed Counter and Pulse Output Instructions

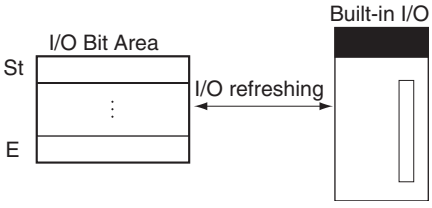
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
MODE CONTROL INI @INI 880	 **P:** Port specifier **C:** Control data **NV:** 1st word with new PV	INI(880) is used to start and stop target value comparison, to change the present value (PV) of a high-speed counter, to change the circular maximum count for a high-speed counter or pulse output counter, to change the PV of a pulse output, or to stop pulse output. INI(880) is also used, for example, to change the circular maximum count or present value for a sampling counter.	Output Required	521
HIGH-SPEED COUNTER PV READ PRV @PRV 881	 **P:** Port specifier **C:** Control data **D:** 1st destination word	PRV(881) is used to read the present value (PV) of a high-speed counter, pulse output, or high-speed counter latch. PRV(881) is used to read analog I/O values for the FQM1-MMA22.	Output Required	527
COMPARISON TABLE LOAD CTBL @CTBL 882	 **P:** Port specifier **C:** Control data **TB:** 1st comparison table word	CTBL(882) is used to perform target value or range comparisons for the present value (PV) of a high-speed counter. It is also used to start high-speed analog sampling.	Output Required	530
SPEED OUTPUT SPED @SPED 885	 **P:** Port specifier **M:** Output mode **F:** 1st pulse frequency word	SPED(885) is used to specify the frequency and perform pulse output without acceleration or deceleration. SPED(885) is also used to produce analog outputs for the FQM1-MMA22.	Output Required	537
SET PULSES PULS @PULS 886	 **P:** Port specifier **T:** Pulse type **N:** Number of pulses	PULS(886) is used to set the number of pulses for pulse output. PULS(886) also executes pulse outputs for absolute positions.	Output Required	543

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
PULSE OUTPUT PLS2 @PLS2 887	PLS2(887) P M S **P:** Port specifier **M:** Output mode **S:** 1st word of settings table	PLS2(887) is used to set the pulse frequency and acceleration/deceleration rates, and to perform pulse output with acceleration/deceleration (with different acceleration/deceleration rates). Only positioning is possible.	Output Required	550
ACCELERATION CONTROL ACC @ACC 888	ACC(888) P M S **P:** Port specifier **M:** Output mode **S:** 1st word of settings table	ACC(888) is used to set the pulse frequency and acceleration/deceleration rates, and to perform pulse output with acceleration/deceleration (with the same acceleration/deceleration rate). Both positioning and speed control are possible. ACC(888) is also used to produce slopped analog outputs for the FQM1-MMA22.	Output Required	555

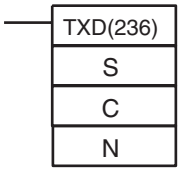
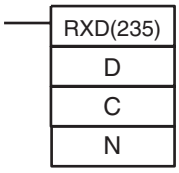
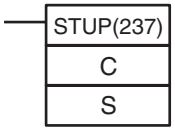
2-2-20 Step Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
STEP DEFINE STEP 008	STEP(008) B **B**: Bit	STEP(008) functions in following 2 ways, depending on its position and whether or not a control bit has been specified. (1)Starts a specific step. (2)Ends the step programming area (i.e., step execution). The step programming area extends from the first STEP(008) with a step number and the first STEP(008) without a step number.	Output Required	563
STEP START SNXT 009	SNXT(009) B **B**: Bit	SNXT(009) is used in the following three ways: (1)To start step programming execution. (2)To proceed to the next step control bit. (3)To end step programming execution.	Output Required	563

2-2-21 I/O Refresh Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
I/O REFRESH IORF @IORF 097	IORF(097) St E **St**: Starting word **E**: End word	Refreshes the specified I/O words of the Module's built-in I/O. 	Output Required	580

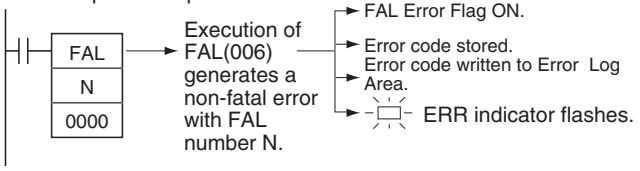
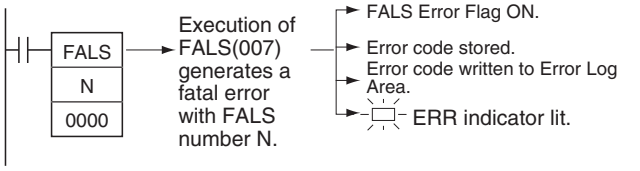
2-2-22 Serial Communications Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
TRANSMIT TXD @TXD 236	 **S**: 1st source word **C**: Control word **N**: Number of bytes 0000 to 0100 hex (0 to 256 decimal)	Outputs the specified number of bytes of data starting from the specified word from the RS-232C or RS-422A port (no-protocol mode) built into the Coordinator Module without conversion and using the start and end codes specified in the System Setup.	Output Required	582
RECEIVE RXD @RXD 235	 **D**: 1st destination word **C**: Control word **N**: Number of bytes to store 0000 to 0100 hex (0 to 256 decimal)	Receives the specified number of bytes of data from the RS-232C or RS-422A port (no-protocol mode) built into the Coordinator Module without conversion and using the start and end codes specified in the System Setup and store the data starting from the specified word.	Output Required	587
CHANGE SERIAL PORT SETUP STUP 237	 **C**: Control word (port) **S**: First source word	Changes the communications parameters of the built-in serial port on the Coordinator Module during operation.	Output Required	592

2-2-23 Debugging Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
TRACE MEMORY SAMPLING TRSM 045		When TRSM(045) is executed, the status of a preselected bit or word is sampled and stored in Trace Memory. TRSM(045) can be used anywhere in the program, any number of times.	Output Not required	596

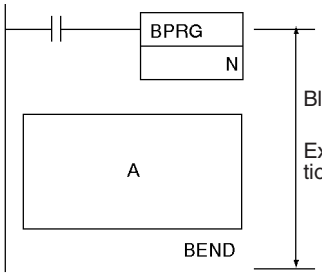
2-2-24 Failure Diagnosis Instructions

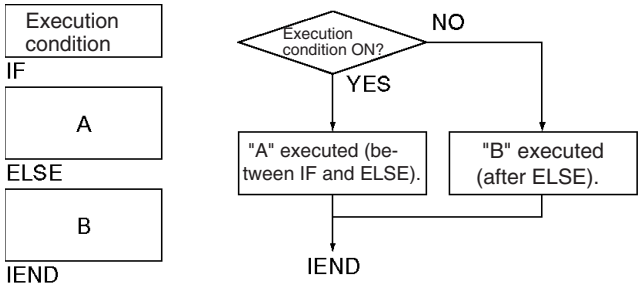
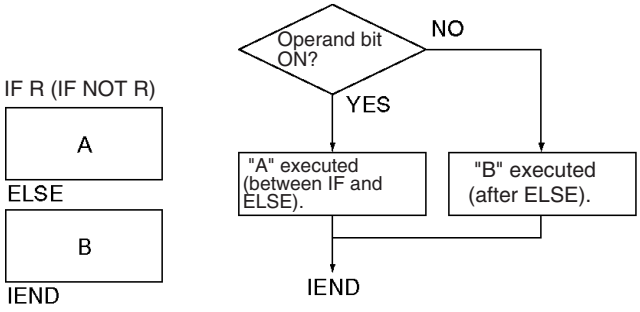
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
FAILURE ALARM FAL @FAL 006	FAL(006) N C **N**: FAL number **C**: Error code to generate (#0000 to #FFFF)	Generates or clears user-defined non-fatal errors. Non-fatal errors do not stop FQM1 operation. 	Output Required	600
SEVERE FAILURE ALARM FALS 007	FALS(007) N S **N:** FALS number **S:** Error code to generate (#0000 to #FFFF)	Generates user-defined fatal errors. Fatal errors stop FQM1 operation. 	Output Required	603

2-2-25 Other Instructions

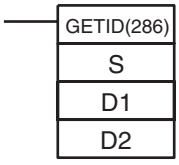
Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
SET CARRY STC @STC 040	STC(040)	Sets the Carry Flag (CY).	Output Required	606
CLEAR CARRY CLC @CLC 041	CLC(041)	Turns OFF the Carry Flag (CY).	Output Required	606

2-2-26 Block Programming Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
BLOCK PROGRAM BEGIN BPRG 096	BPRG(096) N **N**: Block program number	Define a block programming area. For every BPRG(096) there must be a corresponding BEND(801). 	Output Required	611
BLOCK PROGRAM END BEND 801	---	Define a block programming area. For every BPRG(096) there must be a corresponding BEND(801).	Block program Required	611

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
CONDITIONAL BLOCK BRANCHING IF 802	IF (802)	If the execution condition is ON, the instructions between IF(802) and ELSE(803) will be executed and if the execution condition is OFF, the instructions between ELSE(803) and IEND(804) will be executed. 	Block program Required	613
CONDITIONAL BLOCK BRANCHING IF 802	IF (802) B **B**: Bit operand	If the operand bit is ON, the instructions between IF(802) and ELSE(803) will be executed. If the operand bit is OFF, the instructions between ELSE(803) and IEND(804) will be executed. 	Block program Required	613
CONDITIONAL BLOCK BRANCHING (NOT) IF NOT 802	IF (802) NOT B **B**: Bit operand	If the operand bit is OFF, the instructions between IF(802) and ELSE(803) will be executed. If the operand bit is ON, the instructions between ELSE(803) and IEND(804) will be executed.	Block program Required	613
CONDITIONAL BLOCK BRANCHING (ELSE) ELSE 803	---	If the ELSE(803) instruction is omitted and the operand bit is ON, the instructions between IF(802) and IEND(804) will be executed	Block program Required	613
CONDITIONAL BLOCK BRANCHING END IEND 804	---	If the operand bit is OFF, only the instructions after IEND(804) will be executed.	Block program Required	613

2-2-27 Special Function Block Instructions

Instruction Mnemonic Code	Symbol/Operand	Function	Location Execution condition	Page
GET VARIABLE ID GETID @GETID 286	 **S**: Variable or address **D1**: ID code **D2**: Destination word	Outputs the FINS command variable type (data area) code and word address for the specified variable or address. This instruction is generally used to get the assigned address of a variable in a function block.	Output Required	618

2-3 Alphabetical List of Instructions by Mnemonic

A

Mnemonic	Instruction	Function code	Upward Differentiation	Downward Differentiation	Page
ACC	ACCELERATION CONTROL	888	@ACC	---	555
ACOS	ARC COSINE	464	@ACOS	---	410
ACOSD	DOUBLE ARC COSINE	855	@ACOSD	---	452
AND	AND	---	@AND	%AND	99
AND <	AND LESS THAN	310	---	---	167
AND <>	AND NOT EQUAL	305	---	---	167
AND <>D	AND DOUBLE FLOATING NOT EQUAL	336	---	---	462
AND <>F	AND FLOATING NOT EQUAL	330	---	---	421
AND <>L	AND DOUBLE NOT EQUAL	306	---	---	167
AND <>S	AND SIGNED NOT EQUAL	307	---	---	167
AND <>SL	AND DOUBLE SIGNED NOT EQUAL	308	---	---	167
AND <D	AND DOUBLE FLOATING LESS THAN	337	---	---	462
AND <F	AND FLOATING LESS THAN	331	---	---	421
AND <L	AND DOUBLE LESS THAN	311	---	---	167
AND <S	AND SIGNED LESS THAN	312	---	---	167
AND <SL	AND DOUBLE SIGNED LESS THAN	313	---	---	167
AND =	AND EQUAL	300	---	---	167
AND =D	AND DOUBLE FLOATING EQUAL	335	---	---	462
AND =F	AND FLOATING EQUAL	329	---	---	421
AND =L	AND DOUBLE EQUAL	301	---	---	167
AND =S	AND SIGNED EQUAL	302	---	---	167
AND =SL	AND DOUBLE SIGNED EQUAL	303	---	---	167
AND >	AND GREATER THAN	320	---	---	167
AND >D	AND DOUBLE FLOATING GREATER THAN	339	---	---	462
AND >F	AND FLOATING GREATER THAN	333	---	---	421
AND >L	AND DOUBLE GREATER THAN	321	---	---	167
AND >S	AND SIGNED GREATER THAN	322	---	---	167
AND >SL	AND DOUBLE SIGNED GREATER THAN	323	---	---	167
AND LD	AND LOAD	---	---	---	105
AND NOT	AND NOT	---	---	---	101
AND TST	AND BIT TEST	350	---	---	113
AND TSTN	AND BIT TEST	351	---	---	113
AND <=	AND LESS THAN OR EQUAL	315	---	---	167
AND <=D	AND DOUBLE FLOATING LESS THAN OR EQUAL	338	---	---	462
AND <=F	AND FLOATING LESS THAN OR EQUAL	332	---	---	421
AND <=L	AND DOUBLE LESS THAN OR EQUAL	316	---	---	167
AND <=S	AND SIGNED LESS THAN OR EQUAL	317	---	---	167
AND <=SL	AND DOUBLE SIGNED LESS THAN OR EQUAL	318	---	---	167
AND >=	AND GREATER THAN OR EQUAL	325	---	---	167
AND >=D	AND DOUBLE FLOATING GREATER THAN OR EQUAL	340	---	---	462
AND >=F	AND FLOATING GREATER THAN OR EQUAL	334	---	---	421
AND >=L	AND DOUBLE GREATER THAN OR EQUAL	326	---	---	167
AND >=S	AND SIGNED GREATER THAN OR EQUAL	327	---	---	167
AND >=SL	AND DOUBLE SIGNED GREATER THAN OR EQUAL	328	---	---	167

Mnemonic	Instruction	Function code	Upward Differentiation	Downward Differentiation	Page
ANDL	DOUBLE LOGICAL AND	610	@ANDL	---	353
ANDW	LOGICAL AND	034	@ANDW	---	351
APR	ARITHMETIC PROCESS	069	@APR	---	368
ASC	ASCII CONVERT	086	@ASC	---	341
ASFT	ASYNCHRONOUS SHIFT REGISTER	017	@ASFT	---	230
ASIN	ARC SINE	463	@ASIN	---	408
ASIND	DOUBLE ARC SINE	854	@ASIND	---	450
ASL	ARITHMETIC SHIFT LEFT	025	@ASL	---	233
ASLL	DOUBLE SHIFT LEFT	570	@ASLL	---	235
ASR	ARITHMETIC SHIFT RIGHT	026	@ASR	---	236
ASRL	DOUBLE SHIFT RIGHT	571	@ASRL	---	238
ATAN	ARC TANGENT	465	@ATAN	---	412
ATAND	DOUBLE ARC TANGENT	856	@ATAND	---	454
AVG	AVERAGE	195	---	---	486
AXIS	VIRTUAL AXIS	981	---	---	376

B

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
BCD	BINARY-TO-BCD	024	@BCD	---	334
BCDL	DOUBLE BINARY-TO-DOUBLE BCD	059	@BCDL	---	336
BCMP	UNSIGNED BLOCK COMPARE	068	@BCMP	---	187
BCMP2	EXPANDED BLOCK COMPARE	502	@BCMP2	---	190
BCNT	BIT COUNTER	067	@BCNT	---	375
BEND	BLOCK PROGRAM END	801	---	---	611
BIN	BCD-TO-BINARY	023	@BIN	---	331
BINL	DOUBLE BCD-TO-DOUBLE BINARY	058	@BINL	---	333
BPRG	BLOCK PROGRAM BEGIN	096	---	---	611
BREAK	BREAK LOOP	514	---	---	150
BSET	BLOCK SET	071	@BSET	---	213

C

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
CJP	CONDITIONAL JUMP	510	---	---	141
CJPN	CONDITIONAL JUMP	511	---	---	141
CLC	CLEAR CARRY	041	@CLC	---	606
CLI	CLEAR INTERRUPT	691	@CLI	---	512
CMP	COMPARE	020	---	---	172
CMPL	DOUBLE COMPARE	060	---	---	175
CNT	COUNTER	---	---	---	160
CNTR	REVERSIBLE COUNTER	012	---	---	163
COLL	DATA COLLECT	081	@COLL	---	219
COM	COMPLEMENT	029	@COM	---	365
COML	DOUBLE COMPLEMENT	614	@COML	---	366
COS	COSINE	461	@COS	---	404
COSD	DOUBLE COSINE	852	@COSD	---	447
CPS	SIGNED BINARY COMPARE	114	---	---	177
CPSL	DOUBLE SIGNED BINARY COMPARE	115	---	---	180
CTBL	COMPARISON TABLE LOAD	882	@CTBL	---	530

D

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
DBL	16-BIT BINARY TO DOUBLE FLOATING	843	@DBL	---	433
DBLL	32-BIT BINARY TO DOUBLE FLOATING	844	@DBLL	---	434
DEG	RADIANS-TO DEGREES	459	@DEG	---	401
DEGD	DOUBLE RADIANS TO DEGREES	850	@RADD	---	444
DI	DISABLE INTERRUPTS	693	@DI	---	513
DIFD	DIFFERENTIATE DOWN	014	---	---	122
DIFU	DIFFERENTIATE UP	013	---	---	122
DIST	SINGLE WORD DISTRIBUTE	080	@DIST	---	217
DOWN	CONDITION OFF	522	---	---	112

E

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
EI	ENABLE INTERRUPTS	694	---	---	514
ELSE	ELSE	803	---	---	613
END	END	001	---	---	134
EXP	EXPONENT	467	@EXP	---	415
EXPD	DOUBLE EXPONENT	858	@EXPD	---	457

F

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
FAL	FAILURE ALARM	006	@FAL	---	600
FALS	SEVERE FAILURE ALARM	007	---	---	603
FIX	FLOATING TO 16-BIT	450	@FIX	---	386
FIXD	DOUBLE FLOATING TO 16-BIT BINARY	841	@FIXD	---	430
FIXL	FLOATING TO 32-BIT	451	@FIXL	---	388
FIXLD	DOUBLE FLOATING TO 32-BIT BINARY	842	@FIXLD	---	432
FLT	16-BIT TO FLOATING	452	@FLT	---	389
FLTL	32-BIT TO FLOATING	453	@FLTL	---	390
FOR	FOR-NEXT LOOPS	512	---	---	147

G

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
GETID	GET VARIABLE ID	286	@GETID	---	618

H

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
HEX	ASCII TO HEX	162	@HEX	---	345

I

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
IEND	IF END	804	---	---	613
IF NOT (operand)	IF NOT	802	---	---	613
IF (input condition)	IF	802	---	---	613
IF (operand)	IF	802	---	---	613
IL	INTERLOCK	002	---	---	135
ILC	INTERLOCK CLEAR	003	---	---	135

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
INI	MODE CONTROL	880	@INI	---	521
IORF	I/O REFRESH	097	@IORF	---	580

J

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
JME	JUMP END	005	---	---	138
JME0	MULTIPLE JUMP END	516	---	---	145
JMP	JUMP	004	---	---	138
JMP0	MULTIPLE JUMP	515	---	---	145
JSB	JUMP TO SUBROUTINE	982	---	---	503

K

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
KEEP	KEEP	011	---	---	119

L

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
LD	LOAD	---	@LD	%LD	95
LD <	LOAD LESS THAN	310	---	---	167
LD <D	LOAD DOUBLE FLOATING LESS THAN	337	---	---	462
LD <F	LOAD FLOATING LESS THAN	331	---	---	421
LD <>	LOAD NOT EQUAL	305	---	---	167
LD <>D	LOAD DOUBLE FLOATING NOT EQUAL	336	---	---	462
LD <>F	LOAD FLOATING NOT EQUAL	330	---	---	421
LD <>L	LOAD DOUBLE NOT EQUAL	306	---	---	167
LD <>S	LOAD SIGNED NOT EQUAL	307	---	---	167
LD <>SL	LOAD DOUBLE SIGNED NOT EQUAL	308	---	---	167
LD <L	LOAD DOUBLE LESS THAN	311	---	---	167
LD <S	LOAD SIGNED LESS THAN	312	---	---	167
LD <SL	LOAD DOUBLE SIGNED LESS THAN	313	---	---	167
LD =	LOAD EQUAL	300	---	---	167
LD =D	LOAD DOUBLE FLOATING EQUAL	335	---	---	462
LD =F	LOAD FLOATING EQUAL	329	---	---	421
LD =L	LOAD DOUBLE EQUAL	301	---	---	167
LD =S	LOAD SIGNED EQUAL	302	---	---	167
LD =SL	LOAD DOUBLE SIGNED EQUAL	303	---	---	167
LD >	LOAD GREATER THAN	320	---	---	167
LD >D	LOAD DOUBLE FLOATING GREATER THAN	339	---	---	462
LD >F	LOAD FLOATING GREATER THAN	333	---	---	421
LD >L	LOAD DOUBLE GREATER THAN	321	---	---	167
LD >S	LOAD SIGNED GREATER THAN	322	---	---	167
LD >SL	LOAD DOUBLE SIGNED GREATER THAN	323	---	---	167
LD NOT	LOAD NOT	---	---	---	97
LD TST	LOAD BIT TEST	350	---	---	113
LD TSTN	LOAD BIT TEST	351	---	---	113
LD <=	LOAD LESS THAN OR EQUAL	315	---	---	167
LD <=D	LOAD DOUBLE FLOATING LESS THAN OR EQUAL	338	---	---	462
LD <=F	LOAD FLOATING LESS THAN OR EQUAL	332	---	---	421
LD <=L	LOAD DOUBLE LESS THAN OR EQUAL	316	---	---	167

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
LD <=S	LOAD SIGNED LESS THAN OR EQUAL	317	---	---	167
LD <=SL	LOAD DOUBLE SIGNED LESS THAN OR EQUAL	318	---	---	167
LD >=	LOAD GREATER THAN OR EQUAL	325	---	---	167
LD >=D	LOAD DOUBLE FLOATING GREATER THAN OR EQUAL	340	---	---	462
LD >=F	LOAD FLOATING GREATER THAN OR EQUAL	334	---	---	421
LD >=L	LOAD DOUBLE GREATER THAN OR EQUAL	326	---	---	167
LD >=S	LOAD SIGNED GREATER THAN OR EQUAL	327	---	---	167
LD >=SL	LOAD DOUBLE SIGNED GREATER THAN OR EQUAL	328	---	---	167
LOG	LOGARITHM	468	@LOG	---	417
LOGD	DOUBLE LOGARITHM	859	@LOGD	---	459

M

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
MAX	FIND MAXIMUM	182	@MAX	---	467
MCMP	MULTIPLE COMPARE	019	@MCMP	---	182
MCRO	MACRO	099	@MCRO	---	496
MIN	FIND MINIMUM	183	@MIN	---	471
MOV	MOVE	021	@MOV	---	199
MOVB	MOVE BIT	082	@MOVB	---	204
MOVD	MOVE DIGIT	083	@MOVD	---	206
MOVL	DOUBLE MOVE	498	@MOVL	---	201
MOVR	MOVE TO REGISTER	560	@MOVR	---	221
MOVRW	MOVE TIMER/COUNTER PV TO REGISTER	561	---	---	222
MSKR	READ INTERRUPT MASK	692	@MSKR	---	510
MSKS	SET INTERRUPT MASK	690	@MSKS	---	508
MVN	MOVE NOT	022	@MVN	---	200
MVNL	DOUBLE MOVE NOT	499	@MVNL	---	203

N

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
NASL	SHIFT N-BITS LEFT	580	@NASL	---	254
NASR	SHIFT N-BITS RIGHT	581	@NASR	---	259
NEG	2'S COMPLEMENT	160	@NEG	---	338
NEGL	DOUBLE 2'S COMPLEMENT	161	@NEGL	---	339
NEXT	FOR-NEXT LOOPS	513	---	---	147
NOP	NO OPERATION	000	---	---	134
NOT	NOT	520	---	---	111
NSLL	DOUBLE SHIFT N-BITS LEFT	582	@NSLL	---	256
NSRL	DOUBLE SHIFT N-BITS RIGHT	583	@NSRL	---	261

O

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
OR	OR	---	@OR	%OR	102
OR <	OR LESS THAN	310	---	---	167

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
OR <>	OR NOT EQUAL	305	---	---	167
OR <>D	OR DOUBLE FLOATING NOT EQUAL	336	---	---	462
OR <>F	OR FLOATING NOT EQUAL	330	---	---	421
OR <>L	OR DOUBLE NOT EQUAL	306	---	---	167
OR <>S	OR SIGNED NOT EQUAL	307	---	---	167
OR <>SL	OR DOUBLE SIGNED NOT EQUAL	308	---	---	167
OR <D	OR DOUBLE FLOATING LESS THAN	337	---	---	462
OR <F	OR FLOATING LESS THAN	331	---	---	421
OR <L	OR DOUBLE LESS THAN	311	---	---	167
OR <S	OR SIGNED LESS THAN	312	---	---	167
OR <SL	OR DOUBLE SIGNED LESS THAN	313	---	---	167
OR =	OR EQUAL	300	---	---	167
OR =D	OR DOUBLE FLOATING EQUAL	335	---	---	462
OR =F	OR FLOATING EQUAL	329	---	---	421
OR =L	OR DOUBLE EQUAL	301	---	---	167
OR =S	OR SIGNED EQUAL	302	---	---	167
OR =SL	OR DOUBLE SIGNED EQUAL	303	---	---	167
OR >	OR GREATER THAN	320	---	---	167
OR >D	OR DOUBLE FLOATING GREATER THAN	339	---	---	462
OR >F	OR FLOATING GREATER THAN	333	---	---	421
OR >L	OR DOUBLE GREATER THAN	321	---	---	167
OR >S	OR SIGNED GREATER THAN	322	---	---	167
OR >SL	OR DOUBLE SIGNED GREATER THAN	323	---	---	167
OR LD	OR LOAD	---	---	---	107
OR NOT	OR NOT	---	---	---	104
OR TST	OR BIT TEST	350	---	---	113
OR TSTN	OR BIT TEST	351	---	---	113
OR <=	OR LESS THAN OR EQUAL	315	---	---	167
OR <=D	OR DOUBLE FLOATING LESS THAN OR EQUAL	338	---	---	462
OR <=F	OR FLOATING LESS THAN OR EQUAL	332	---	---	421
OR <=L	OR DOUBLE LESS THAN OR EQUAL	316	---	---	167
OR <=S	OR SIGNED LESS THAN OR EQUAL	317	---	---	167
OR <=SL	OR DOUBLE SIGNED LESS THAN OR EQUAL	318	---	---	167
OR >=	OR GREATER THAN OR EQUAL	325	---	---	167
OR >=D	OR DOUBLE FLOATING GREATER THAN OR EQUAL	340	---	---	462
OR >=F	OR FLOATING GREATER THAN OR EQUAL	334	---	---	421
OR >=L	OR DOUBLE GREATER THAN OR EQUAL	326	---	---	167
OR >=S	OR SIGNED GREATER THAN OR EQUAL	327	---	---	167
OR >=SL	OR DOUBLE SIGNED GREATER THAN OR EQUAL	328	---	---	167
ORW	LOGICAL OR	035	@ORW	---	354
ORWL	DOUBLE LOGICAL OR	611	@ORWL	---	356
OUT	OUTPUT	---	---	---	117
OUT NOT	OUTPUT NOT	---	---	---	118
OUTB	SINGLE BIT OUTPUT	534	@OUTB	---	132

P

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
PRV	HIGH-SPEED COUNTER PV READ	881	@PRV	---	527
PULS	SET PULSES	886	@PULS	---	543
PLS2	PULSE OUTPUT	887	@PLS2	---	550
PWR	EXPONENTIAL POWER	840	@PWR	---	419
PWRD	DOUBLE EXPONENTIAL POWER	860	@PWRD	---	461

R

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
RAD	DEGREES TO RADIANS	458	@RAD	---	400
RADD	DOUBLE DEGREES TO RADIANS	849	@RADD	---	443
RET	SUBROUTINE RETURN	093	---	---	503
RLNC	ROTATE LEFT WITHOUT CARRY	574	@RLNC	---	245
RLNL	DOUBLE ROTATE LEFT WITHOUT CARRY	576	@RLNL	---	247
ROL	ROTATE LEFT	027	@ROL	---	239
ROLL	DOUBLE ROTATE LEFT	572	@ROLL	---	241
ROR	ROTATE RIGHT	028	@ROR	---	242
RORL	DOUBLE ROTATE RIGHT	573	@RORL	---	244
RRNC	ROTATE RIGHT WITHOUT CARRY	575	@RRNC	---	248
RRNL	DOUBLE ROTATE RIGHT WITHOUT CARRY	577	@RRNL	---	250
RSET	RESET	---	@RSET	%RSET	125
RSTA	MULTIPLE BIT RESET	531	@RSTA	---	126
RSTB	SINGLE BIT RESET	533	@RSTB	---	129
RXD	RECEIVE	235	@RXD	--	587

S

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
SBN	SUBROUTINE ENTRY	092	---	---	500
SBS	SUBROUTINE CALL	091	@SBS	---	491
SCL	SCALING	194	@SCL	---	475
SCL2	SCALING 2	486	@SCL2	---	479
SCL3	SCALING 3	487	@SCL3	---	483
SET	SET	---	@SET	%SET	125
SETA	MULTIPLE BIT SET	530	@SETA	---	126
SETB	SINGLE BIT SET	532	@SETB	---	129
SFT	SHIFT REGISTER	010	---	---	225
SFTR	REVERSIBLE SHIFT REGISTER	084	@SFTR	---	227
SIN	SINE	460	@SIN	---	403
SIND	DOUBLE SINE	851	@SIND	---	446
SLD	ONE DIGIT SHIFT LEFT	074	@SLD	---	251
SNXT	STEP START	009	---	---	563
SPED	SPEED OUTPUT	885	@SPED	---	537
SQRT	SQUARE ROOT	466	@SQRT	---	413
SQRTD	DOUBLE SQUARE ROOT	857	@SQRTD	---	456
SRD	ONE DIGIT SHIFT RIGHT	075	@SRD	---	253
STC	SET CARRY	040	@STC	---	606
STIM	INTERVAL TIMER	980	@STIM	---	516
STEP	STEP DEFINE	008	---	---	563
STUP	CHANGE SERIAL PORT SETUP	237	@STUP	---	592

T

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
TAN	TANGENT	462	@TAN	---	406
TAND	DOUBLE TANGENT	853	@TAND	---	449
TCMP	TABLE COMPARE	085	@TCMP	---	185
TIM	TIMER	---	---	---	153
TIMH	HIGH-SPEED TIMER	015	---	---	156
TMHH	ONE-MS TIMER	540	---	---	158
TRSM	TRACE MEMORY SAMPLING	045	---	---	596
TXD	TRANSMIT	236	@TXD	---	582

U

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
UP	CONDITION ON	521	---	---	112

W

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
WSFT	WORD SHIFT	016	@WSFT	---	232

X

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
XCGL	DOUBLE DATA EXCHANGE	562	@XCGL	---	216
XCHG	DATA EXCHANGE	073	@XCHG	---	215
XFER	BLOCK TRANSFER	070	@XFER	---	211
XFRB	MULTIPLE BIT TRANSFER	062	@XFRB	---	208
XNRL	DOUBLE EXCLUSIVE NOR	613	@XNRL	---	363
XNRW	EXCLUSIVE NOR	037	@XNRW	---	362
XORL	DOUBLE EXCLUSIVE OR	612	@XORL	---	360
XORW	EXCLUSIVE OR	036	@XORW	---	358

Z

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
ZCP	AREA RANGE COMPARE	088	---	---	193
ZCPL	DOUBLE AREA RANGE COMPARE	116	---	---	196

Symbols

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
+	SIGNED BINARY ADD WITHOUT CARRY	400	@+	---	282
++	INCREMENT BINARY	590	@++	---	265
++B	INCREMENT BCD	594	@++B	---	273
++BL	DOUBLE INCREMENT BCD	595	@++BL	---	275
++L	DOUBLE INCREMENT BINARY	591	@++L	---	267
+B	BCD ADD WITHOUT CARRY	404	@+B	---	289
+BC	BCD ADD WITH CARRY	406	@+BC	---	292
+BCL	DOUBLE BCD ADD WITH CARRY	407	@+BCL	---	293
+BL	DOUBLE BCD ADD WITHOUT CARRY	405	@+BL	---	290
+C	SIGNED BINARY ADD WITH CARRY	402	@+C	---	285
+CL	DOUBLE SIGNED BINARY ADD WITH CARRY	403	@+CL	---	287
+D	DOUBLE FLOATING-POINT ADD	845	@+D	---	436

Mnemonic	Instruction	FUN code	Upward Differentiation	Downward Differentiation	Page
+F	FLOATING-POINT ADD	454	@+F	---	392
+L	DOUBLE SIGNED BINARY ADD WITHOUT CARRY	401	@+L	---	283
–	SIGNED BINARY SUBTRACT WITHOUT CARRY	410	@–	---	295
– –	DECREMENT BINARY	592	@– –	---	269
– –B	DECREMENT BCD	596	@– –B	---	277
– –BL	DOUBLE DECREMENT BCD	597	@– –BL	---	279
– –L	DOUBLE DECREMENT BINARY	593	@– –L	---	271
–B	BCD SUBTRACT WITHOUT CARRY	414	@–B	---	304
–BC	BCD SUBTRACT WITH CARRY	416	@–BC	---	309
–BCL	DOUBLE BCD SUBTRACT WITH CARRY	417	@–BCL	---	310
–BL	DOUBLE BCD SUBTRACT WITHOUT CARRY	415	@–BL	---	306
–C	SIGNED BINARY SUBTRACT WITH CARRY	412	@–C	---	300
–CL	DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	413	@–CL	---	302
–D	DOUBLE FLOATING-POINT SUBTRACT	846	@–D	---	437
–F	FLOATING-POINT SUBTRACT	455	@–F	---	394
*	SIGNED BINARY MULTIPLY	420	@*	---	312
*B	BCD MULTIPLY	424	@*B	---	318
*BL	DOUBLE BCD MULTIPLY	425	@*BL	---	320
*D	DOUBLE FLOATING-POINT MULTIPLY	847	@*D	---	439
*F	FLOATING-POINT MULTIPLY	456	@*F	---	396
*L	DOUBLE SIGNED BINARY MULTIPLY	421	@*L	---	314
*U	UNSIGNED BINARY MULTIPLY	422	@*U	---	315
*UL	DOUBLE UNSIGNED BINARY MULTIPLY	423	@*UL	---	317
–L	DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	411	@–L	---	296
/	SIGNED BINARY DIVIDE	430	@/	---	321
/B	BCD DIVIDE	434	@/B	---	328
/BL	DOUBLE BCD DIVIDE	435	@/BL	---	329
/D	DOUBLE FLOATING-POINT DIVIDE	848	@/D	---	441
/F	FLOATING-POINT DIVIDE	457	@/F	---	397
/L	DOUBLE SIGNED BINARY DIVIDE	431	@/L	---	323
/U	UNSIGNED BINARY DIVIDE	432	@/U	---	324
/UL	DOUBLE UNSIGNED BINARY DIVIDE	433	@/UL	---	326

2-4 List of Instructions by Function Code

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Page
---	LD	LOAD	@LD	%LD	95
---	LD NOT	LOAD NOT	---	---	97
---	AND	AND	@AND	%AND	99
---	AND NOT	AND NOT	---	---	101
---	OR	OR	@OR	%OR	102
---	OR NOT	OR NOT	---	---	104
---	AND LD	AND LOAD	---	---	105
---	OR LD	OR LOAD	---	---	107
---	OUT	OUTPUT	---	---	117
---	OUT NOT	OUTPUT NOT	---	---	118
---	SET	SET	@SET	%SET	125
---	RSET	RESET	@RSET	%RSET	125
---	TIM	TIMER	---	---	153
---	CNT	COUNTER	---	---	160
000	NOP	NO OPERATION	---	---	134
001	END	END	---	---	134
002	IL	INTERLOCK	---	---	135
003	ILC	INTERLOCK CLEAR	---	---	135
004	JMP	JUMP	---	---	138
005	JME	JUMP END	---	---	138
006	FAL	FAILURE ALARM	@FAL	---	600
007	FALS	SEVERE FAILURE ALARM	---	---	603
008	STEP	STEP DEFINE	---	---	563
009	SNXT	STEP START	---	---	563
010	SFT	SHIFT REGISTER	---	---	225
011	KEEP	KEEP	---	---	119
012	CNTR	REVERSIBLE COUNTER	---	---	163
013	DIFU	DIFFERENTIATE UP	---	---	122
014	DIFD	DIFFERENTIATE DOWN	---	---	122
015	TIMH	HIGH-SPEED TIMER	---	---	156
016	WSFT	WORD SHIFT	@WSFT	---	232
017	ASFT	ASYNCHRONOUS SHIFT REGISTER	@ASFT	---	230
019	MCMP	MULTIPLE COMPARE	@MCMP	---	182
020	CMP	UNSIGNED COMPARE	---	---	172
021	MOV	MOVE	@MOV	---	199
022	MVN	MOVE NOT	@MVN	---	200
023	BIN	BCD-TO-BINARY	@BIN	---	331
024	BCD	BINARY-TO-BCD	@BCD	---	334
025	ASL	ARITHMETIC SHIFT LEFT	@ASL	---	233
026	ASR	ARITHMETIC SHIFT RIGHT	@ASR	---	236
027	ROL	ROTATE LEFT	@ROL	---	239
028	ROR	ROTATE RIGHT	@ROR	---	242
029	COM	COMPLEMENT	@COM	---	365
034	ANDW	LOGICAL AND	@ANDW	---	351
035	ORW	LOGICAL OR	@ORW	---	354
036	XORW	EXCLUSIVE OR	@XORW	---	358
037	XNRW	EXCLUSIVE NOR	@XNRW	---	362
040	STC	SET CARRY	@STC	---	606
041	CLC	CLEAR CARRY	@CLC	---	606
045	TRSM	TRACE MEMORY SAMPLING	---	---	596

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Page
058	BINL	DOUBLE BCD-TO-DOUBLE BINARY	@BINL	---	333
059	BCDL	DOUBLE BINARY-TO-DOUBLE BCD	@BCDL	---	336
060	CMPL	DOUBLE UNSIGNED COMPARE	---	---	175
062	XFRB	MULTIPLE BIT TRANSFER	@XFRB	---	208
067	BCNT	BIT COUNTER	@BCNT	---	375
068	BCMP	UNSIGNED BLOCK COMPARE	@BCMP	---	187
069	APR	ARITHMETIC PROCESS	@APR	---	368
070	XFER	BLOCK TRANSFER	@XFER	---	211
071	BSET	BLOCK SET	@BSET	---	213
073	XCHG	DATA EXCHANGE	@XCHG	---	215
074	SLD	ONE DIGIT SHIFT LEFT	@SLD	---	251
075	SRD	ONE DIGIT SHIFT RIGHT	@SRD	---	253
080	DIST	SINGLE WORD DISTRIBUTE	@DIST	---	217
081	COLL	DATA COLLECT	@COLL	---	219
082	MOVB	MOVE BIT	@MOVB	---	204
083	MOVD	MOVE DIGIT	@MOVD	---	206
084	SFTR	REVERSIBLE SHIFT REGISTER	@SFTR	---	227
085	TCMP	TABLE COMPARE	@TCMP	---	185
086	ASC	ASCII CONVERT	@ASC	---	341
088	ZCP	AREA RANGE COMPARE	---	---	193
091	SBS	SUBROUTINE CALL	@SBS	---	491
092	SBN	SUBROUTINE ENTRY	---	---	500
093	RET	SUBROUTINE RETURN	---	---	503
096	BPRG	BLOCK PROGRAM BEGIN	---	---	611
097	IORF	I/O REFRESH	@IORF	---	580
099	MCRO	MACRO	@MCRO	---	496
114	CPS	SIGNED BINARY COMPARE	---	---	177
115	CPSL	DOUBLE SIGNED BINARY COMPARE	---	---	180
116	ZCPL	DOUBLE AREA RANGE COMPARE	---	---	196
160	NEG	2'S COMPLEMENT	@NEG	---	338
161	NEGL	DOUBLE 2'S COMPLEMENT	@NEGL	---	339
162	HEX	ASCII TO HEX	@HEX	---	345
182	MAX	FIND MAXIMUM	@MAX	---	467
183	MIN	FIND MINIMUM	@MIN	---	471
194	SCL	SCALING	@SCL	---	475
195	AVG	AVERAGE	---	---	486
235	RXD	RECEIVE	@RXD	---	587
236	TXD	TRANSMIT	@TXD	---	582
237	STUP	CHANGE SERIAL PORT SETUP	@STUP	---	592
286	GETID	GET VARIABLE ID	@GETID	---	618
300	AND =	AND EQUAL	---	---	167
300	LD =	LOAD EQUAL	---	---	167
300	OR =	OR EQUAL	---	---	167
301	AND =L	AND DOUBLE EQUAL	---	---	167
301	LD =L	LOAD DOUBLE EQUAL	---	---	167
301	OR =L	OR DOUBLE EQUAL	---	---	167
302	AND =S	AND SIGNED EQUAL	---	---	167
302	LD =S	LOAD SIGNED EQUAL	---	---	167
302	OR =S	OR SIGNED EQUAL	---	---	167
303	AND =SL	AND DOUBLE SIGNED EQUAL	---	---	167
303	LD =SL	LOAD DOUBLE SIGNED EQUAL	---	---	167
303	OR =SL	OR DOUBLE SIGNED EQUAL	---	---	167

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Page
305	AND <>	AND NOT EQUAL	---	---	167
305	LD <>	LOAD NOT EQUAL	---	---	167
305	OR <>	OR NOT EQUAL	---	---	167
306	AND <>L	AND DOUBLE NOT EQUAL	---	---	167
306	LD <>L	LOAD DOUBLE NOT EQUAL	---	---	167
306	OR <>L	OR DOUBLE NOT EQUAL	---	---	167
307	AND <>S	AND SIGNED NOT EQUAL	---	---	167
307	LD <>S	LOAD SIGNED NOT EQUAL	---	---	167
307	OR <>S	OR SIGNED NOT EQUAL	---	---	167
308	AND <>SL	AND DOUBLE SIGNED NOT EQUAL	---	---	167
308	LD <>SL	LOAD DOUBLE SIGNED NOT EQUAL	---	---	167
308	OR <>SL	OR DOUBLE SIGNED NOT EQUAL	---	---	167
310	AND <	AND LESS THAN	---	---	167
310	LD <	LOAD LESS THAN	---	---	167
310	OR <	OR LESS THAN	---	---	167
311	AND <L	AND DOUBLE LESS THAN	---	---	167
311	LD <L	LOAD DOUBLE LESS THAN	---	---	167
311	OR <L	OR DOUBLE LESS THAN	---	---	167
312	AND <S	AND SIGNED LESS THAN	---	---	167
312	LD <S	LOAD SIGNED LESS THAN	---	---	167
312	OR <S	OR SIGNED LESS THAN	---	---	167
313	AND <SL	AND DOUBLE SIGNED LESS THAN	---	---	167
313	LD <SL	LOAD DOUBLE SIGNED LESS THAN	---	---	167
313	OR <SL	OR DOUBLE SIGNED LESS THAN	---	---	167
315	AND <=	AND LESS THAN OR EQUAL	---	---	167
315	LD <=	LOAD LESS THAN OR EQUAL	---	---	167
315	OR <=	OR LESS THAN OR EQUAL	---	---	167
316	AND <=L	AND DOUBLE LESS THAN OR EQUAL	---	---	167
316	LD <=L	LOAD DOUBLE LESS THAN OR EQUAL	---	---	167
316	OR <=L	OR DOUBLE LESS THAN OR EQUAL	---	---	167
317	AND <=S	AND SIGNED LESS THAN OR EQUAL	---	---	167
317	LD <=S	LOAD SIGNED LESS THAN OR EQUAL	---	---	167
317	OR <=S	OR SIGNED LESS THAN OR EQUAL	---	---	167
318	AND <=SL	AND DOUBLE SIGNED LESS THAN OR EQUAL	---	---	167
318	LD <=SL	LOAD DOUBLE SIGNED LESS THAN OR EQUAL	---	---	167
318	OR <=SL	OR DOUBLE SIGNED LESS THAN OR EQUAL	---	---	167
320	AND >	AND GREATER THAN	---	---	167
320	LD >	LOAD GREATER THAN	---	---	167
320	OR >	OR GREATER THAN	---	---	167
321	AND >L	AND DOUBLE GREATER THAN	---	---	167
321	LD >L	LOAD DOUBLE GREATER THAN	---	---	167
321	OR >L	OR DOUBLE GREATER THAN	---	---	167
322	AND >S	AND SIGNED GREATER THAN	---	---	167
322	LD >S	LOAD SIGNED GREATER THAN	---	---	167
322	OR >S	OR SIGNED GREATER THAN	---	---	167
323	AND >SL	AND DOUBLE SIGNED GREATER THAN	---	---	167
323	LD >SL	LOAD DOUBLE SIGNED GREATER THAN	---	---	167
323	OR >SL	OR DOUBLE SIGNED GREATER THAN	---	---	167
325	AND >=	AND GREATER THAN OR EQUAL	---	---	167
325	LD >=	LOAD GREATER THAN OR EQUAL	---	---	167

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Page
325	OR >=	OR GREATER THAN OR EQUAL	---	---	167
326	AND >=L	AND DOUBLE GREATER THAN OR EQUAL	---	---	167
326	LD >=L	LOAD DOUBLE GREATER THAN OR EQUAL	---	---	167
326	OR >=L	OR DOUBLE GREATER THAN OR EQUAL	---	---	167
327	AND >=S	AND SIGNED GREATER THAN OR EQUAL	---	---	167
327	LD >=S	LOAD SIGNED GREATER THAN OR EQUAL	---	---	167
327	OR >=S	OR SIGNED GREATER THAN OR EQUAL	---	---	167
328	AND >=SL	AND DOUBLE SIGNED GREATER THAN OR EQUAL	---	---	167
328	LD >=SL	LOAD DOUBLE SIGNED GREATER THAN OR EQUAL	---	---	167
328	OR >=SL	OR DOUBLE SIGNED GREATER THAN OR EQUAL	---	---	167
329	AND =F	AND FLOATING EQUAL	---	---	421
329	LD =F	LOAD FLOATING EQUAL	---	---	421
329	OR =F	OR FLOATING EQUAL	---	---	421
330	AND <>F	AND FLOATING NOT EQUAL	---	---	421
330	LD <>F	LOAD FLOATING NOT EQUAL	---	---	421
330	OR <>F	OR FLOATING NOT EQUAL	---	---	421
331	AND <F	AND FLOATING LESS THAN	---	---	421
331	LD <F	LOAD FLOATING LESS THAN	---	---	421
331	OR <F	OR FLOATING LESS THAN	---	---	421
332	AND <=F	AND FLOATING LESS THAN OR EQUAL	---	---	421
332	LD <=F	LOAD FLOATING LESS THAN OR EQUAL	---	---	421
332	OR <=F	OR FLOATING LESS THAN OR EQUAL	---	---	421
333	AND >F	AND FLOATING GREATER THAN	---	---	421
333	LD >F	LOAD FLOATING GREATER THAN	---	---	421
333	OR >F	OR FLOATING GREATER THAN	---	---	421
334	AND >=F	AND FLOATING GREATER THAN OR EQUAL	---	---	421
334	LD >=F	LOAD FLOATING GREATER THAN OR EQUAL	---	---	421
334	OR >=F	OR FLOATING GREATER THAN OR EQUAL	---	---	421
335	AND =D	AND DOUBLE FLOATING EQUAL	---	---	462
335	LD =D	LOAD DOUBLE FLOATING EQUAL	---	---	462
335	OR =D	OR DOUBLE FLOATING EQUAL	---	---	462
336	AND <>D	AND DOUBLE FLOATING NOT EQUAL	---	---	462
336	LD <>D	LOAD DOUBLE FLOATING NOT EQUAL	---	---	462
336	OR <>D	OR DOUBLE FLOATING NOT EQUAL	---	---	462
337	AND <D	AND DOUBLE FLOATING LESS THAN	---	---	462
337	LD <D	LOAD DOUBLE FLOATING LESS THAN	---	---	462
337	OR <D	OR DOUBLE FLOATING LESS THAN	---	---	462
338	AND <=D	AND DOUBLE FLOATING LESS THAN OR EQUAL	---	---	462
338	LD <=D	LOAD DOUBLE FLOATING LESS THAN OR EQUAL	---	---	462
338	OR <=D	OR DOUBLE FLOATING LESS THAN OR EQUAL	---	---	462
339	AND >D	AND DOUBLE FLOATING GREATER THAN	---	---	462
339	LD >D	LOAD DOUBLE FLOATING GREATER THAN	---	---	462

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Page
339	OR >D	OR DOUBLE FLOATING GREATER THAN	---	---	462
340	AND >=D	AND DOUBLE FLOATING GREATER THAN OR EQUAL	---	---	462
340	LD >=D	LOAD DOUBLE FLOATING GREATER THAN OR EQUAL	---	---	462
340	OR >=D	OR DOUBLE FLOATING GREATER THAN OR EQUAL	---	---	462
350	AND TST	AND BIT TEST	---	---	113
350	LD TST	LOAD BIT TEST	---	---	113
350	OR TST	OR BIT TEST	---	---	113
351	AND TSTN	AND BIT TEST NOT	---	---	113
351	LD TSTN	LOAD BIT TEST NOT	---	---	113
351	OR TSTN	OR BIT TEST NOT	---	---	113
400	+	SIGNED BINARY ADD WITHOUT CARRY	@+	---	282
401	+L	DOUBLE SIGNED BINARY ADD WITHOUT CARRY	@+L	---	283
402	+C	SIGNED BINARY ADD WITH CARRY	@+C	---	285
403	+CL	DOUBLE SIGNED BINARY ADD WITH CARRY	@+CL	---	287
404	+B	BCD ADD WITHOUT CARRY	@+B	---	289
405	+BL	DOUBLE BCD ADD WITHOUT CARRY	@+BL	---	290
406	+BC	BCD ADD WITH CARRY	@+BC	---	292
407	+BCL	DOUBLE BCD ADD WITH CARRY	@+BCL	---	293
410	−	SIGNED BINARY SUBTRACT WITHOUT CARRY	@−	---	295
411	−L	DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	@−L	---	296
412	−C	SIGNED BINARY SUBTRACT WITH CARRY	@−C	---	300
413	−CL	DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	@−CL	---	302
414	−B	BCD SUBTRACT WITHOUT CARRY	@−B	---	304
415	−BL	DOUBLE BCD SUBTRACT WITHOUT CARRY	@−BL	---	306
416	−BC	BCD SUBTRACT WITH CARRY	@−BC	---	309
417	−BCL	DOUBLE BCD SUBTRACT WITH CARRY	@−BCL	---	310
420	*	SIGNED BINARY MULTIPLY	@*	---	312
421	*L	DOUBLE SIGNED BINARY MULTIPLY	@*L	---	314
422	*U	UNSIGNED BINARY MULTIPLY	@*U	---	315
423	*UL	DOUBLE UNSIGNED BINARY MULTIPLY	@*UL	---	317
424	*B	BCD MULTIPLY	@*B	---	318
425	*BL	DOUBLE BCD MULTIPLY	@*BL	---	320
430	/	SIGNED BINARY DIVIDE	@/	---	321
431	/L	DOUBLE SIGNED BINARY DIVIDE	@/L	---	323
432	/U	UNSIGNED BINARY DIVIDE	@/U	---	324
433	/UL	DOUBLE UNSIGNED BINARY DIVIDE	@/UL	---	326
434	/B	BCD DIVIDE	@/B	---	328
435	/BL	DOUBLE BCD DIVIDE	@/BL	---	329
450	FIX	FLOATING TO 16-BIT	@FIX	---	386
451	FIXL	FLOATING TO 32-BIT	@FIXL	---	388
452	FLT	16-BIT TO FLOATING	@FLT	---	389
453	FLTL	32-BIT TO FLOATING	@FLTL	---	390
454	+F	FLOATING-POINT ADD	@+F	---	392
455	−F	FLOATING-POINT SUBTRACT	@−F	---	394
456	*F	FLOATING-POINT MULTIPLY	@*F	---	396

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Page
457	/F	FLOATING-POINT DIVIDE	@/F	---	397
458	RAD	DEGREES TO RADIANS	@RAD	---	400
459	DEG	RADIANS-TO DEGREES	@DEG	---	401
460	SIN	SINE	@SIN	---	403
461	COS	COSINE	@COS	---	404
462	TAN	TANGENT	@TAN	---	406
463	ASIN	ARC SINE	@ASIN	---	408
464	ACOS	ARC COSINE	@ACOS	---	410
465	ATAN	ARC TANGENT	@ATAN	---	412
466	SQRT	SQUARE ROOT	@SQRT	---	413
467	EXP	EXPONENT	@EXP	---	415
468	LOG	LOGARITHM	@LOG	---	417
486	SCL2	SCALING 2	@SCL2	---	479
487	SCL3	SCALING 3	@SCL3	---	483
498	MOVL	DOUBLE MOVE	@MOVL	---	201
499	MVNL	DOUBLE MOVE NOT	@MVNL	---	203
502	BCMP2	EXPANDED BLOCK COMPARE	@BCMP2	---	190
510	CJP	CONDITIONAL JUMP	---	---	141
511	CJPN	CONDITIONAL JUMP	---	---	141
512	FOR	FOR-NEXT LOOPS	---	---	147
513	NEXT	FOR-NEXT LOOPS	---	---	147
514	BREAK	BREAK LOOP	---	---	150
515	JMP0	MULTIPLE JUMP	---	---	145
516	JME0	MULTIPLE JUMP END	---	---	145
520	NOT	NOT	---	---	111
521	UP	CONDITION ON	---	---	112
522	DOWN	CONDITION OFF	---	---	112
530	SETA	MULTIPLE BIT SET	@SETA	---	126
531	RSTA	MULTIPLE BIT RESET	@RSTA	---	126
532	SETB	SINGLE BIT SET	@SETB	---	129
533	RSTB	SINGLE BIT RESET	@RSTB	---	129
534	OUTB	SINGLE BIT OUTPUT	@OUTB	---	132
540	TMHH	ONE-MS TIMER	---	---	158
560	MOVR	MOVE TO REGISTER	@MOVR	---	221
561	MOVRW	MOVE TIMER/COUNTER PV TO REGISTER	@MOVRW	---	222
562	XCGL	DOUBLE DATA EXCHANGE	@XCGL	---	216
570	ASLL	DOUBLE SHIFT LEFT	@ASLL	---	235
571	ASRL	DOUBLE SHIFT RIGHT	@ASRL	---	238
572	ROLL	DOUBLE ROTATE LEFT	@ROLL	---	241
573	RORL	DOUBLE ROTATE RIGHT	@RORL	---	244
574	RLNC	ROTATE LEFT WITHOUT CARRY	@RLNC	---	245
575	RRNC	ROTATE RIGHT WITHOUT CARRY	@RRNC	---	248
576	RLNL	DOUBLE ROTATE LEFT WITHOUT CARRY	@RLNL	---	247
577	RRNL	DOUBLE ROTATE RIGHT WITHOUT CARRY	@RRNL	---	250
580	NASL	SHIFT N-BITS LEFT	@NASL	---	254
581	NASR	SHIFT N-BITS RIGHT	@NASR	---	259
582	NSLL	DOUBLE SHIFT N-BITS LEFT	@NSLL	---	256
583	NSRL	DOUBLE SHIFT N-BITS RIGHT	@NSRL	---	261
590	++	INCREMENT BINARY	@++	---	265

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Page
591	++L	DOUBLE INCREMENT BINARY	@++L	---	267
592	−−	DECREMENT BINARY	@−−	---	269
593	−−L	DOUBLE DECREMENT BINARY	@−−L	---	271
594	++B	INCREMENT BCD	@++B	---	273
595	++BL	DOUBLE INCREMENT BCD	@++BL	---	275
596	−−B	DECREMENT BCD	@−−B	---	277
597	−−BL	DOUBLE DECREMENT BCD	@−−BL	---	279
610	ANDL	DOUBLE LOGICAL AND	@ANDL	---	353
611	ORWL	DOUBLE LOGICAL OR	@ORWL	---	356
612	XORL	DOUBLE EXCLUSIVE OR	@XORL	---	360
613	XNRL	DOUBLE EXCLUSIVE NOR	@XNRL	---	363
614	COML	DOUBLE COMPLEMENT	@COML	---	366
690	MSKS	SET INTERRUPT MASK	@MSKS	---	508
691	CLI	CLEAR INTERRUPT	@CLI	---	512
692	MSKR	READ INTERRUPT MASK	@MSKR	---	510
693	DI	DISABLE INTERRUPTS	@DI	---	513
694	EI	ENABLE INTERRUPTS	---	---	514
801	BEND	BLOCK PROGRAM END	---	---	611
802	IF	CONDITIONAL BRANCHING BLOCK	---	---	613
802	IF	CONDITIONAL BRANCHING BLOCK	---	---	613
802	IF NOT	CONDITIONAL BRANCHING BLOCK NOT	---	---	613
803	ELSE	ELSE	---	---	613
804	IEND	IF END	---	---	613
840	PWR	EXPONENTIAL POWER	@PWR	---	419
841	FIXD	DOUBLE FLOATING TO 16-BIT BINARY	@FIXD	---	430
842	FIXLD	DOUBLE FLOATING TO 32-BIT BINARY	@FIXLD	---	432
843	DBL	16-BIT BINARY TO DOUBLE FLOATING	@DBL	---	433
844	DBLL	32-BIT BINARY TO DOUBLE FLOATING	@DBLL	---	434
845	+D	DOUBLE FLOATING-POINT ADD	@+D	---	436
846	−D	DOUBLE FLOATING-POINT SUBTRACT	@−D	---	437
847	∗D	DOUBLE FLOATING-POINT MULTIPLY	@∗D	---	439
848	/D	DOUBLE FLOATING-POINT DIVIDE	@/D	---	441
849	RADD	DOUBLE DEGREES TO RADIANS	@RADD	---	443
850	DEGD	DOUBLE RADIANS TO DEGREES	@RADD	---	444
851	SIND	DOUBLE SINE	@SIND	---	446
852	COSD	DOUBLE COSINE	@COSD	---	447
853	TAND	DOUBLE TANGENT	@TAND	---	449
854	ASIND	DOUBLE ARC SINE	@ASIND	---	450
855	ACOSD	DOUBLE ARC COSINE	@ACOSD	---	452
856	ATAND	DOUBLE ARC TANGENT	@ATAND	---	454
857	SQRTD	DOUBLE SQUARE ROOT	@SQRTD	---	456
858	EXPD	DOUBLE EXPONENT	@EXPD	---	457
859	LOGD	DOUBLE LOGARITHM	@LOGD	---	459
860	PWRD	DOUBLE EXPONENTIAL POWER	@PWRD	---	461
880	INI	MODE CONTROL	@INI	---	521
881	PRV	HIGH-SPEED COUNTER PV READ	@PRV	---	527
882	CTBL	COMPARISON TABLE LOAD	@CTBL	---	530
885	SPED	SPEED OUTPUT	@SPED	---	537
886	PULS	SET PULSES	@PULS	---	543
887	PLS2	PULSE OUTPUT	@PLS2	---	550
888	ACC	ACCELERATION CONTROL	@ACC	---	555
980	STIM	INTERVAL TIMER	@STIM	---	516

Function code	Mnemonic	Instruction	Upward Differentiation	Downward Differentiation	Page
981	AXIS	VIRTUAL AXIS	---	---	376
982	JSB	JUMP TO SUBROUTINE	---	---	503

SECTION 3
Instructions

This section describes each of the instructions that can be used in programming the FQM1. Instructions are described in order of function, as classified in *Section 2 Summary of Instructions.*

3-1 Notation and Layout of Instruction Descriptions 92
3-2 Sequence Input Instructions 95
3-2-1 LOAD: LD 95
3-2-2 LOAD NOT: LD NOT 97
3-2-3 AND: AND 99
3-2-4 AND NOT: AND NOT 101
3-2-5 OR: OR 102
3-2-6 OR NOT: OR NOT 104
3-2-7 AND LOAD: AND LD 105
3-2-8 OR LOAD: OR LD 107
3-2-9 Differentiated Instructions 109
3-2-10 Operation Timing for I/O Instructions 110
3-2-11 TR Bits 110
3-2-12 NOT: NOT(520) 111
3-2-13 CONDITION ON/OFF: UP(521) and DOWN(522) 112
3-2-14 BIT TEST: TST(350) and TSTN(351) 113
3-3 Sequence Output Instructions 117
3-3-1 OUTPUT: OUT 117
3-3-2 OUTPUT NOT: OUT NOT 118
3-3-3 KEEP: KEEP(011) 119
3-3-4 DIFFERENTIATE UP/DOWN: DIFU(013) and DIFD(014) 122
3-3-5 SET and RESET: SET and RSET 125
3-3-6 MULTIPLE BIT SET/RESET: SETA(530)/RSTA(531) 126
3-3-7 SINGLE BIT SET/RESET: SETB(532)/RSTB(533) 129
3-3-8 SINGLE BIT OUTPUT: OUTB(534) 132
3-4 Sequence Control Instructions 134
3-4-1 END: END(001) 134
3-4-2 NO OPERATION: NOP(000) 134
3-4-3 INTERLOCK and INTERLOCK CLEAR: IL(002) and ILC(003) 135
3-4-4 JUMP and JUMP END: JMP(004) and JME(005) 138
3-4-5 CONDITIONAL JUMP: CJP(510)/CJPN(511) 141
3-4-6 MULTIPLE JUMP and JUMP END: JMP0(515) and JME0(516) 145
3-4-7 FOR-NEXT LOOPS: FOR(512)/NEXT(513) 147
3-4-8 BREAK LOOP: BREAK(514) 150
3-5 Timer and Counter Instructions 152
3-5-1 TIMER: TIM 153
3-5-2 HIGH-SPEED TIMER: TIMH(015) 156
3-5-3 ONE-MS TIMER: TMHH(540) 158
3-5-4 COUNTER: CNT 160
3-5-5 REVERSIBLE COUNTER: CNTR(012) 163
3-6 Comparison Instructions 167
3-6-1 Input Comparison Instructions (300 to 328) 167
3-6-2 COMPARE: CMP(020) 172
3-6-3 DOUBLE COMPARE: CMPL(060) 175
3-6-4 SIGNED BINARY COMPARE: CPS(114) 177
3-6-5 DOUBLE SIGNED BINARY COMPARE: CPSL(115) 180
3-6-6 MULTIPLE COMPARE: MCMP(019) 182
3-6-7 TABLE COMPARE: TCMP(085) 185

3-6-8 BLOCK COMPARE: BCMP(068) . . . 187
3-6-9 EXPANDED BLOCK COMPARE: BCMP2(502) . . . 190
3-6-10 AREA RANGE COMPARE: ZCP(088) . . . 193
3-6-11 DOUBLE AREA RANGE COMPARE: ZCPL(116) . . . 196
3-7 Data Movement Instructions . . . 199
3-7-1 MOVE: MOV(021) . . . 199
3-7-2 MOVE NOT: MVN(022) . . . 200
3-7-3 DOUBLE MOVE: MOVL(498) . . . 201
3-7-4 DOUBLE MOVE NOT: MVNL(499) . . . 203
3-7-5 MOVE BIT: MOVB(082) . . . 204
3-7-6 MOVE DIGIT: MOVD(083) . . . 206
3-7-7 MULTIPLE BIT TRANSFER: XFRB(062) . . . 208
3-7-8 BLOCK TRANSFER: XFER(070) . . . 211
3-7-9 BLOCK SET: BSET(071) . . . 213
3-7-10 DATA EXCHANGE: XCHG(073) . . . 215
3-7-11 DOUBLE DATA EXCHANGE: XCGL(562) . . . 216
3-7-12 SINGLE WORD DISTRIBUTE: DIST(080) . . . 217
3-7-13 DATA COLLECT: COLL(081) . . . 219
3-7-14 MOVE TO REGISTER: MOVR(560) . . . 221
3-7-15 MOVE TIMER/COUNTER PV TO REGISTER: MOVRW(561) . . . 222
3-8 Data Shift Instructions . . . 225
3-8-1 SHIFT REGISTER: SFT(010) . . . 225
3-8-2 REVERSIBLE SHIFT REGISTER: SFTR(084) . . . 227
3-8-3 ASYNCHRONOUS SHIFT REGISTER: ASFT(017) . . . 230
3-8-4 WORD SHIFT: WSFT(016) . . . 232
3-8-5 ARITHMETIC SHIFT LEFT: ASL(025) . . . 233
3-8-6 DOUBLE SHIFT LEFT: ASLL(570) . . . 235
3-8-7 ARITHMETIC SHIFT RIGHT: ASR(026) . . . 236
3-8-8 DOUBLE SHIFT RIGHT: ASRL(571) . . . 238
3-8-9 ROTATE LEFT: ROL(027) . . . 239
3-8-10 DOUBLE ROTATE LEFT: ROLL(572) . . . 241
3-8-11 ROTATE RIGHT: ROR(028) . . . 242
3-8-12 DOUBLE ROTATE RIGHT: RORL(573) . . . 244
3-8-13 ROTATE LEFT WITHOUT CARRY: RLNC(574) . . . 245
3-8-14 DOUBLE ROTATE LEFT WITHOUT CARRY: RLNL(576) . . . 247
3-8-15 ROTATE RIGHT WITHOUT CARRY: RRNC(575) . . . 248
3-8-16 DOUBLE ROTATE RIGHT WITHOUT CARRY: RRNL(577) . . . 250
3-8-17 ONE DIGIT SHIFT LEFT: SLD(074) . . . 251
3-8-18 ONE DIGIT SHIFT RIGHT: SRD(075) . . . 253
3-8-19 SHIFT N-BITS LEFT: NASL(580) . . . 254
3-8-20 DOUBLE SHIFT N-BITS LEFT: NSLL(582) . . . 256
3-8-21 SHIFT N-BITS RIGHT: NASR(581) . . . 259
3-8-22 DOUBLE SHIFT N-BITS RIGHT: NSRL(583) . . . 261
3-9 Increment/Decrement Instructions . . . 265
3-9-1 INCREMENT BINARY: ++(590) . . . 265
3-9-2 DOUBLE INCREMENT BINARY: ++L(591) . . . 267
3-9-3 DECREMENT BINARY: – –(592) . . . 269
3-9-4 DOUBLE DECREMENT BINARY: – –L(593) . . . 271
3-9-5 INCREMENT BCD: ++B(594) . . . 273
3-9-6 DOUBLE INCREMENT BCD: ++BL(595) . . . 275
3-9-7 DECREMENT BCD: – –B(596) . . . 277
3-9-8 DOUBLE DECREMENT BCD: – –BL(597) . . . 279
3-10 Symbol Math Instructions . . . 281
3-10-1 SIGNED BINARY ADD WITHOUT CARRY: +(400) . . . 282
3-10-2 DOUBLE SIGNED BINARY ADD WITHOUT CARRY: +L(401) . . . 283
3-10-3 SIGNED BINARY ADD WITH CARRY: +C(402) . . . 285
3-10-4 DOUBLE SIGNED BINARY ADD WITH CARRY: +CL(403) . . . 287

3-10-5 BCD ADD WITHOUT CARRY: +B(404) . . . 289
3-10-6 DOUBLE BCD ADD WITHOUT CARRY: +BL(405) . . . 290
3-10-7 BCD ADD WITH CARRY: +BC(406) . . . 292
3-10-8 DOUBLE BCD ADD WITH CARRY: +BCL(407) . . . 293
3-10-9 SIGNED BINARY SUBTRACT WITHOUT CARRY: –(410) . . . 295
3-10-10 DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY: –L(411) . . . 296
3-10-11 SIGNED BINARY SUBTRACT WITH CARRY: –C(412) . . . 300
3-10-12 DOUBLE SIGNED BINARY SUBTRACT WITH CARRY: –CL(413) . . . 302
3-10-13 BCD SUBTRACT WITHOUT CARRY: –B(414) . . . 304
3-10-14 DOUBLE BCD SUBTRACT WITHOUT CARRY: –BL(415) . . . 306
3-10-15 BCD SUBTRACT WITH CARRY: –BC(416) . . . 309
3-10-16 DOUBLE BCD SUBTRACT WITH CARRY: –BCL(417) . . . 310
3-10-17 SIGNED BINARY MULTIPLY: *(420) . . . 312
3-10-18 DOUBLE SIGNED BINARY MULTIPLY: *L(421) . . . 314
3-10-19 UNSIGNED BINARY MULTIPLY: *U(422) . . . 315
3-10-20 DOUBLE UNSIGNED BINARY MULTIPLY: *UL(423) . . . 317
3-10-21 BCD MULTIPLY: *B(424) . . . 318
3-10-22 DOUBLE BCD MULTIPLY: *BL(425) . . . 320
3-10-23 SIGNED BINARY DIVIDE: /(430) . . . 321
3-10-24 DOUBLE SIGNED BINARY DIVIDE: /L(431) . . . 323
3-10-25 UNSIGNED BINARY DIVIDE: /U(432) . . . 324
3-10-26 DOUBLE UNSIGNED BINARY DIVIDE: /UL(433) . . . 326
3-10-27 BCD DIVIDE: /B(434) . . . 328
3-10-28 DOUBLE BCD DIVIDE: /BL(435) . . . 329
3-11 Conversion Instructions . . . 331
3-11-1 BCD-TO-BINARY: BIN(023) . . . 331
3-11-2 DOUBLE BCD-TO-DOUBLE BINARY: BINL(058) . . . 333
3-11-3 BINARY-TO-BCD: BCD(024) . . . 334
3-11-4 DOUBLE BINARY-TO-DOUBLE BCD: BCDL(059) . . . 336
3-11-5 2'S COMPLEMENT: NEG(160) . . . 338
3-11-6 DOUBLE 2'S COMPLEMENT: NEGL(161) . . . 339
3-11-7 ASCII CONVERT: ASC(086) . . . 341
3-11-8 ASCII TO HEX: HEX(162) . . . 345
3-11-9 16-BIT TO 32-BIT SIGNED BINARY: SIGN(600) . . . 349
3-12 Logic Instructions . . . 351
3-12-1 LOGICAL AND: ANDW(034) . . . 351
3-12-2 DOUBLE LOGICAL AND: ANDL(610) . . . 353
3-12-3 LOGICAL OR: ORW(035) . . . 354
3-12-4 DOUBLE LOGICAL OR: ORWL(611) . . . 356
3-12-5 EXCLUSIVE OR: XORW(036) . . . 358
3-12-6 DOUBLE EXCLUSIVE OR: XORL(612) . . . 360
3-12-7 EXCLUSIVE NOR: XNRW(037) . . . 362
3-12-8 DOUBLE EXCLUSIVE NOR: XNRL(613) . . . 363
3-12-9 COMPLEMENT: COM(029) . . . 365
3-12-10 DOUBLE COMPLEMENT: COML(614) . . . 366
3-13 Special Math Instructions . . . 368
3-13-1 ARITHMETIC PROCESS: APR(069) . . . 368
3-13-2 BIT COUNTER: BCNT(067) . . . 375
3-13-3 VIRTUAL AXIS: AXIS(981) . . . 376
3-14 Floating-point Math Instructions . . . 380
3-14-1 FLOATING TO 16-BIT: FIX(450) . . . 386
3-14-2 FLOATING TO 32-BIT: FIXL(451) . . . 388
3-14-3 16-BIT TO FLOATING: FLT(452) . . . 389
3-14-4 32-BIT TO FLOATING: FLTL(453) . . . 390
3-14-5 FLOATING-POINT ADD: +F(454) . . . 392
3-14-6 FLOATING-POINT SUBTRACT: –F(455) . . . 394
3-14-7 FLOATING-POINT MULTIPLY: *F(456) . . . 396

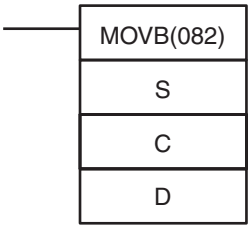
3-14-8 FLOATING-POINT DIVIDE: /F(457) . . . 397
3-14-9 DEGREES TO RADIANS: RAD(458) . . . 400
3-14-10 RADIANS TO DEGREES: DEG(459) . . . 401
3-14-11 SINE: SIN(460) . . . 403
3-14-12 COSINE: COS(461) . . . 404
3-14-13 TANGENT: TAN(462) . . . 406
3-14-14 ARC SINE: ASIN(463) . . . 408
3-14-15 ARC COSINE: ACOS(464) . . . 410
3-14-16 ARC TANGENT: ATAN(465) . . . 412
3-14-17 SQUARE ROOT: SQRT(466) . . . 413
3-14-18 EXPONENT: EXP(467) . . . 415
3-14-19 LOGARITHM: LOG(468) . . . 417
3-14-20 EXPONENTIAL POWER: PWR(840) . . . 419
3-14-21 Single-precision Floating-point Comparison Instructions . . . 421
3-15 Double-precision Floating-point Instructions . . . 425
3-15-1 DOUBLE FLOATING TO 16-BIT: FIXD(841) . . . 430
3-15-2 DOUBLE FLOATING TO 32-BIT: FIXLD(842) . . . 432
3-15-3 16-BIT TO DOUBLE FLOATING: DBL(843) . . . 433
3-15-4 32-BIT TO DOUBLE FLOATING: DBLL(844) . . . 434
3-15-5 DOUBLE FLOATING-POINT ADD: +D(845) . . . 436
3-15-6 DOUBLE FLOATING-POINT SUBTRACT: –D(846) . . . 437
3-15-7 DOUBLE FLOATING-POINT MULTIPLY: *D(847) . . . 439
3-15-8 DOUBLE FLOATING-POINT DIVIDE: /D(848) . . . 441
3-15-9 DOUBLE DEGREES TO RADIANS: RADD(849) . . . 443
3-15-10 DOUBLE RADIANS TO DEGREES: DEGD(850) . . . 444
3-15-11 DOUBLE SINE: SIND(851) . . . 446
3-15-12 DOUBLE COSINE: COSD(852) . . . 447
3-15-13 DOUBLE TANGENT: TAND(853) . . . 449
3-15-14 DOUBLE ARC SINE: ASIND(854) . . . 450
3-15-15 DOUBLE ARC COSINE: ACOSD(855) . . . 452
3-15-16 DOUBLE ARC TANGENT: ATAND(856) . . . 454
3-15-17 DOUBLE SQUARE ROOT: SQRTD(857) . . . 456
3-15-18 DOUBLE EXPONENT: EXPD(858) . . . 457
3-15-19 DOUBLE LOGARITHM: LOGD(859) . . . 459
3-15-20 DOUBLE EXPONENTIAL POWER: PWRD(860) . . . 461
3-15-21 Double-precision Floating-point Input Instructions . . . 462
3-16 Table Data Processing Instructions . . . 467
3-16-1 FIND MAXIMUM: MAX(182) . . . 467
3-16-2 FIND MINIMUM: MIN(183) . . . 471
3-17 Data Control Instructions . . . 475
3-17-1 SCALING: SCL(194) . . . 475
3-17-2 SCALING 2: SCL2(486) . . . 479
3-17-3 SCALING 3: SCL3(487) . . . 483
3-17-4 AVERAGE: AVG(195) . . . 486
3-18 Subroutines . . . 491
3-18-1 SUBROUTINE CALL: SBS(091) . . . 491
3-18-2 MACRO: MCRO(099) . . . 496
3-18-3 SUBROUTINE ENTRY: SBN(092) . . . 500
3-18-4 SUBROUTINE RETURN: RET(093) . . . 503
3-18-5 JUMP SUBROUTINE: JSB(982) . . . 503
3-19 Interrupt Control Instructions . . . 508
3-19-1 SET INTERRUPT MASK: MSKS(690) . . . 508
3-19-2 READ INTERRUPT MASK: MSKR(692) . . . 510
3-19-3 CLEAR INTERRUPT: CLI(691) . . . 512
3-19-4 DISABLE INTERRUPTS: DI(693) . . . 513
3-19-5 ENABLE INTERRUPTS: EI(694) . . . 514
3-19-6 INTERVAL TIMER: STIM(980) . . . 516

3-20 High-speed Counter/Pulse Output Instructions 521
3-20-1 MODE CONTROL: INI(880) 521
3-20-2 HIGH-SPEED COUNTER PV READ: PRV(881) 527
3-20-3 REGISTER COMPARISON TABLE: CTBL(882) 530
3-20-4 SPEED OUTPUT: SPED(885) 537
3-20-5 SET PULSES: PULS(886) 543
3-20-6 PULSE OUTPUT: PLS2(887) 550
3-20-7 ACCELERATION CONTROL: ACC(888) 555
3-21 Step Instructions 562
3-21-1 STEP DEFINE and STEP START: STEP(008)/SNXT(009) 563
3-22 I/O Refresh Instruction 580
3-22-1 I/O REFRESH: IORF(097) 580
3-23 Serial Communications Instructions 582
3-23-1 Serial Communications 582
3-23-2 TRANSMIT: TXD(236) 582
3-23-3 RECEIVE: RXD(235) 587
3-23-4 CHANGE SERIAL PORT SETUP: STUP(237) 592
3-24 Debugging Instructions 596
3-24-1 Trace Memory Sampling: TRSM(045) 596
3-25 Failure Diagnosis Instructions 600
3-25-1 FAILURE ALARM: FAL(006) 600
3-25-2 SEVERE FAILURE ALARM: FALS(007) 603
3-26 Other Instructions 606
3-26-1 SET CARRY: STC(040) 606
3-26-2 CLEAR CARRY: CLC(041) 606
3-27 Block Programming Instructions 608
3-27-1 Introduction 608
3-27-2 BLOCK PROGRAM BEGIN/END: BPRG(096)/BEND(801) 611
3-27-3 Branching: IF (NOT)(802), ELSE(803), and IEND(804) 613
3-28 Function Block Instructions 618
3-28-1 GET VARIABLE ID: GETID(286) 618

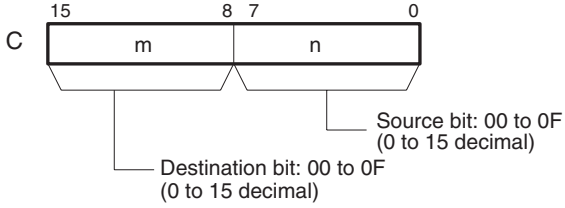
3-1 Notation and Layout of Instruction Descriptions

Instructions are described in groups by function. Refer to *2-3 Alphabetical List of Instructions by Mnemonic* for a list of instructions by mnemonic that lists the page number in this section for each instruction.

The description of each instruction is organized as described in the following table.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Name and Mnemonic</td><td>The heading of each section consists of the name of the instruction followed by the mnemonic with the function code in parentheses. Example: MOVE BIT: MOVB(082)</td></tr>
<tr><td>Purpose</td><td>The basic purpose of the instruction is described after the section heading.</td></tr>
<tr><td>Ladder Symbol and Operand Names</td><td>The ladder symbol used to represent the instruction on the CX-Programmer is shown, as in the example for the MOVE BIT instruction given below. The name of each operand is also provided with the ladder symbol.
 

S: Source word or data
C: Control word
D: Destination word</td></tr>
<tr><td>Variations</td><td>The variations that can be used to control execution of the instruction under special conditions are given using the mnemonic form. Any variation that is not supported by an instruction is given as “Not supported.”
• Executed Each Cycle for ON Condition: The instruction is executed as long as it receives an ON execution condition.
• Executed Once for Upward Differentiation: The instruction is executed during the next cycle only after the execution condition changes from OFF to ON.
• Executed Once for Downward Differentiation: The instruction is executed during the next cycle only after the execution condition changes from ON to OFF.
• Always Executed: The instruction does not require an execution condition and is executed each cycle.
• Creates ON Condition....: The instruction is executed each cycle to create an execution condition for the next instruction.
<table><tr><td>Variations</td><td>Executed Each Cycle for ON Condition</td><td>MOVB(082)</td></tr><tr><td></td><td>Executed Once for Upward Differentiation</td><td>@MOVB(082)</td></tr><tr><td></td><td>Executed Once for Downward Differentiation</td><td>Not supported</td></tr></table></td></tr>
<tr><td>Applicable Program Areas</td><td>The program areas in which the instruction can be used are specified. “OK” indicates the areas in which the instruction can be used.
<table><tr><th>Block program areas</th><th>Step program areas</th><th>Subroutines</th><th>Interrupt tasks</th></tr><tr><td>OK</td><td>OK</td><td>OK</td><td>OK</td></tr></table></td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operands</td><td>Where necessary, the meaning of words and bits used in specific operands, such as control words, is given.
 
</td></tr>
<tr><td>Operand Specifications</td><td>The memory areas addresses that can be used each operand are listed in a table like the following one. The letters used in the column headings on the left are the same as those used in the ladder symbol. “---” is used to indicate when an area cannot be specific for an operand.
<table>
<tr><th>Area</th><th>S</th><th>C</th><th>D</th></tr>
<tr><td>CIO Area</td><td colspan="3">CIO 0000 to CIO 6413</td></tr>
<tr><td>Work Area</td><td colspan="3">W000 to W255</td></tr>
<tr><td>Auxiliary Bit Area</td><td colspan="2">A000 to A447
A448 to A899</td><td>A448 to A899</td></tr>
<tr><td>Timer Area</td><td colspan="3">T0000 to T0255</td></tr>
<tr><td>Counter Area</td><td colspan="3">C0000 to C0255</td></tr>
<tr><td>DM Area</td><td colspan="3">D00000 to D32767</td></tr>
</table></td></tr>
<tr><td>Description</td><td>The function of the instruction and the operands used in the instruction are described.</td></tr>
<tr><td>Flags</td><td>The flags table indicates the status of the condition flags immediately after execution of the instruction. Any flags that are not listed are not affected by the instruction. “OFF” indicates that a flag is turned OFF immediately after execution of the instruction regardless of the results of executing the instruction.
<table>
<tr><th>Name</th><th>Label</th><th>Operation</th></tr>
<tr><td>Error Flag</td><td>ER</td><td>ON if control data is within ranges.
OFF in all other cases.</td></tr>
<tr><td>Equals Flag</td><td>=</td><td>OFF</td></tr>
<tr><td>Negative Flag</td><td>N</td><td>OFF</td></tr>
</table></td></tr>
<tr><td>Precautions</td><td>Special precautions required in using the instruction are provided. Be sure to read and follow these precautions.</td></tr>
<tr><td>Example</td><td>An example of using the instruction with specific operands is provided to further explain the function of the instruction.</td></tr>
</table>

Constants

Constants input for operands are given as listed below.

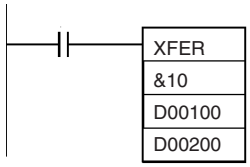
Operand Descriptions and Operand Specifications

- Operands Specifying Bit Strings (Normally Input as Hexadecimal):
  Only the hexadecimal form is given for operands specifying bit strings, e.g., only “#0000 to #FFFF” is specified as the S operand for the MOV(021) instruction. On the CX-Programmer, however, bit strings can be input in decimal form by using the & prefix.
- Operands Specifying Numeric Values (Normally Input as Decimal, Including Jump Numbers):
  Both the decimal and hexadecimal forms are given for operands specifying numeric values, e.g., “#0000 to #FFFF” and “&0 to &65535” are given for the N operand for the XFER(070) instruction.

- Operands Indicating Control Numbers (Except for Jump Numbers):
  The decimal form is given for control numbers, e.g., “0 to 255” is given for the N operand for the SBS(091) instruction.

Examples

In the examples, constants are given using the CX-Programmer notation, e.g., operands specifying numeric values are given in decimal with an & prefix, as shown in the following example.



The input methods for constants from the CX-Programmer are given in the following table.

Operand	CX-Programmer
Operands specifying bit strings (normally input as hexadecimal)	Input as decimal with an & prefix or input as hexadecimal with an # prefix. (See note.)
Operands specifying numeric values (normally input as decimal)	
Operands specifying control numbers (except for jump numbers)	Input as decimal with an # prefix. (See note.)

Note When operands are input on the CX-Programmer, the input ranges will be displayed along with the appropriate prefixes.

Condition Flags

With the CX-Programmer, the condition flags are registered in advance as global symbols with “P_” in front of the symbol name.

Flag	Label used in this manual	CX-Programmer label
Error Flag	ER	P_ER
Access Error Flag	AER	P_AER
Carry Flag	CY	P_CY
Greater Than Flag	>	P_GT
Equals Flag	=	P_EQ
Less Than Flag	<	P_LT
Negative Flag	N	P_N
Overflow Flag	OF	P_OF
Underflow Flag	UF	P_UF
Greater Than or Equals Flag	>=	P_GE
Not Equal Flag	<>	P_NE
Less Than or Equals Flag	<=	P_LE
Always ON Flag	ON	P_On
Always OFF Flag	OFF	P_Off

3-2 Sequence Input Instructions

This section describes the sequence input instructions.

Instruction	Mnemonic	Function code	Page
LOAD	LD	---	95
LOAD NOT	LD NOT	---	97
AND	AND	---	99
AND NOT	AND NOT	---	101
OR	OR	---	102
OR NOT	OR NOT	---	104
AND LOAD	AND LD	---	105
OR LOAD	OR LD	---	107
NOT	NOT	520	111
UP	UP	521	112
DOWN	DOWN	522	112
LOAD BIT TEST	LD TST	350	113
LOAD BIT TEST NOT	LD TSTN	351	113
AND BIT TEST	AND TST	350	113
AND BIT TEST NOT	AND TSTN	351	113
OR BIT TEST	OR TST	350	113
OR BIT TEST NOT	OR TSTN	351	113

3-2-1 LOAD: LD

Purpose

Indicates a logical start and creates an ON/OFF execution condition based on the ON/OFF status of the specified operand bit.

Ladder Symbol

Variations

Variations		
	Restarts Logic and Creates ON Each Cycle Operand Bit is ON	LD
	Restarts Logic and Creates ON Once for Upward Differentiation	@LD
	Restarts Logic and Creates ON Once for Downward Differentiation	%LD

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	LD operand bit
CIO Area	CIO 0000.00 to CIO 6143.15
Work Area	W000.00 to W255.15
Auxiliary Bit Area	A000.00 to A959.15
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255

Area	LD operand bit
Task Flag	TK0000
Condition Flags	ER, CY, N, OF, UF, >, =, <, >=, <>, <=, ON, OFF, AER
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
TR Area	TR0 to TR15
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	---
Data Registers	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Note TR bits are used only when inputting programs in mnemonic form. It is not necessary to input them when programming in ladder diagram form.

Description

LD is used for the first normally open bit from the bus bar or for the first normally open bit of a logic block. The specified bit in I/O memory is read. LD is used in the following circumstances as an instruction for indicating a logical start.

- When directly connecting to the bus bar.
- When logic blocks are connected by AND LD or OR LD, i.e., at the beginning of a logic block.

The AND LOAD and OR LOAD instructions are used to connect in series or in parallel logic blocks beginning with LD or LD NOT.

At least one LOAD or LOAD NOT instruction is required for the execution condition when output-related instructions cannot be connected directly to the bus bar. If there is no LOAD or LOAD NOT instruction, a programming error will occur with the program check by the CX-Programmer.

When logic blocks are connected by AND LOAD or OR LOAD instructions, the total number of AND LOAD/OR LOAD instructions must match the total number of LOAD/LOAD NOT instructions minus1. If they do not match, a programming error will occur. For details, refer to *3-2-7 AND LOAD: AND LD* and *3-2-8 OR LOAD: OR LD*.

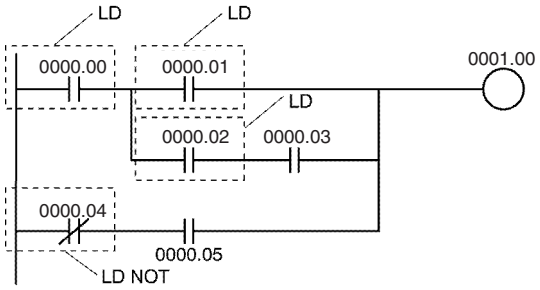
Flags

There are no flags affected by this instruction.

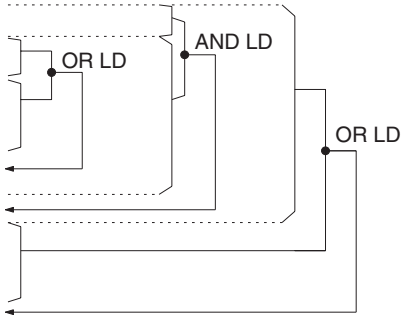
Precautions

Differentiate up (@) or differentiate down (%) can be specified for LD. If differentiate up (@) is specified, the execution condition is turned ON for one cycle only after the status of the operand bit goes from OFF to ON. If differentiate down (%) is specified, the execution condition is turned ON for one cycle only after the status of the operand bit goes from ON to OFF.

Example



Instruction	Operand
LD	0000.00
LD	0000.01
LD	0000.02
AND	0000.03
OR LD	---
AND LD	---
LD NOT	0000.04
AND	0000.05
OR LD	---
OUT	0001.00

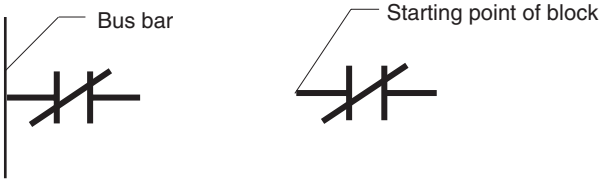


3-2-2 LOAD NOT: LD NOT

Purpose

Indicates a logical start and creates an ON/OFF execution condition based on the reverse of the ON/OFF status of the specified operand bit.

Ladder Symbol



Variations

Variations		
	Restarts Logic and Creates ON Each Cycle Operand Bit is OFF	LD NOT
	Restarts Logic and Creates ON Once for Upward Differentiation	@LD NOT
	Restarts Logic and Creates ON Once for Downward Differentiation	%LD NOT

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	LD NOT bit operand
CIO Area	CIO 0000.00 to CIO 6143.15
Work Area	W000.00 to W255.15
Auxiliary Bit Area	A000.00 to A959.15
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
Task Flag	TK0000

Area	LD NOT bit operand
Condition Flags	ER, CY, N, OF, UF, >, =, <, >=, <>, <=, ON, OFF, AER
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
TR Area	---
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Note TR bits are used only when inputting programs in mnemonic form. It is not necessary to input them when programming in ladder diagram form.

Description

LD NOT is used for the first normally closed bit from the bus bar, or for the first normally closed bit of a logic block. The specified bit in I/O memory is read and reversed. LD NOT is used in the following circumstances as an instruction for indicating a logical start.

- When directly connecting to the bus bar.
- When logic blocks are connected by AND LD or OR LD. (Used at the beginning of a logic block.)

The AND LOAD and OR LOAD instructions are used to connect in series or in parallel logic blocks beginning with LD or LD NOT.

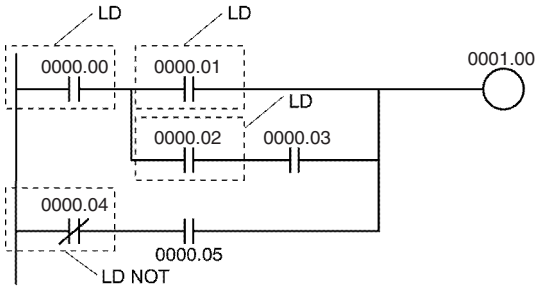
At least one LOAD or LOAD NOT instruction is required for the execution condition when output-related instructions cannot be connected directly to the bus bar. If there is no LOAD or LOAD NOT instruction, a program error will occur with the program check by the CX-Programmer.

When logic blocks are connected by AND LOAD or OR LOAD instructions, the total number of AND LOAD/OR LOAD instructions must match the total number of LOAD/LOAD NOT instructions minus1. If they do not match, a programming error will occur.

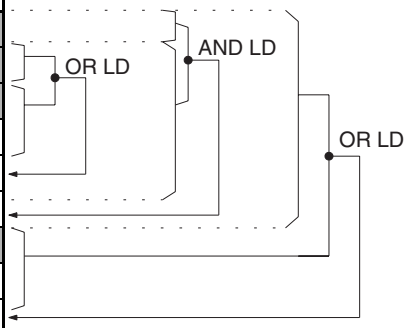
Flags

There are no flags affected by this instruction.

Example



Instruction	Operand
LD	0000.00
LD	0000.01
LD	0000.02
AND	0000.03
OR LD	---
AND LD	---
LD NOT	0000.04
AND	0000.05
OR LD	---
OUT	0001.00



3-2-3 AND: AND

Purpose

Takes a logical AND of the status of the specified operand bit and the current execution condition.

Ladder Symbol

Variations

Variations		
	Creates ON Each Cycle AND Result is ON	AND
	Creates ON Once for Upward Differentiation	@AND
	Creates ON Once for Downward Differentiation	%AND

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	AND bit operand
CIO Area	CIO 0000.00 to CIO 6413.15
Work Area	W000.00 to W255.15
Auxiliary Bit Area	A000.00 to A959.15
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
Task Flag	TK0000
Condition Flags	ER, CY, N, OF, UF, >, =, <, >=, <>, <=, ON, OFF, AER
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
TR Area	---
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

AND is used for a normally open bit connected in series. AND cannot be directly connected to the bus bar, and cannot be used at the beginning of a logic block. The specified bit in I/O memory is read.

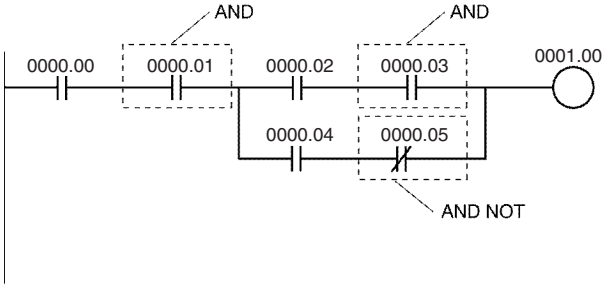
Flags

There are no flags affected by this instruction.

Precautions

Differentiate up (@) or differentiate down (%) can be specified for AND. If differentiate up (@) is specified, the execution condition is turned ON for one cycle only after the status of the operand bit goes from OFF to ON. If differentiate down (%) is specified, the execution condition is turned ON for one cycle only after the status of the operand bit goes from ON to OFF.

Example



Instruction	Operand
LD	0000.00
AND	0000.01
LD	0000.02
AND	0000.03
LD	0000.04
AND NOT	0000.05

Instruction	Operand
OR LD	---
AND LD	---
OUT	0001.00

3-2-4 AND NOT: AND NOT

Purpose

Reverses the status of the specified operand bit and takes a logical AND with the current execution condition.

Ladder Symbol

Variations

Variations		
	Creates ON Each Cycle AND NOT Result is ON	AND NOT
	Creates ON Once for Upward Differentiation	@AND NOT
	Creates ON Once for Downward Differentiation	%AND NOT

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	AND NOT bit operand
CIO Area	CIO 0000.00 to CIO 6413.15
Work Area	W000.00 to W255.15
Auxiliary Bit Area	A000.00 to A959.15
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
Task Flag	TK0000
Condition Flags	ER, CY, N, OF, UF, >, =, <, >=, <>, <=, ON, OFF, AER
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
TR Area	---
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

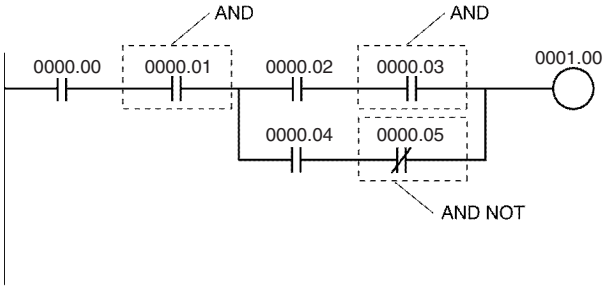
Description

AND NOT is used for a normally closed bit connected in series. AND NOT cannot be directly connected to the bus bar, and cannot be used at the beginning of a logic block. The specified bit in I/O memory is read.

Flags

There are no flags affected by this instruction.

Example



Instruction	Operand
LD	0000.00
AND	0000.01
LD	0000.02
AND	0000.03
LD	0000.04
AND NOT	0000.05
OR LD	---
AND LD	---
OUT	0001.00

3-2-5 OR: OR

Purpose

Takes a logical OR of the ON/OFF status of the specified operand bit and the current execution condition.

Ladder Symbol

Variations

Variations		
	Creates ON Each Cycle OR Result is ON	OR
	Creates ON Once for Upward Differentiation	@OR
	Creates ON Once for Downward Differentiation	%OR

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	OR bit operand
CIO Area	CIO 0000.00 to CIO 6143.15
Work Area	W000.00 to W255.15
Auxiliary Bit Area	A000.00 to A959.15
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
Task Flag	TK0000
Condition Flags	ER, CY, N, OF, UF, >, =, <, >=, <>, <=, ON, OFF, AER
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
DM Area	---
Indirect DM addresses in binary	---

Area	OR bit operand
Indirect DM addresses in BCD	---
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

OR is used for a normally open bit connected in parallel. A normally open bit is configured to form a logical OR with a logic block beginning with a LOAD or LOAD NOT instruction (connected to the bus bar or at the beginning of the logic block). The specified bit in I/O memory is read.

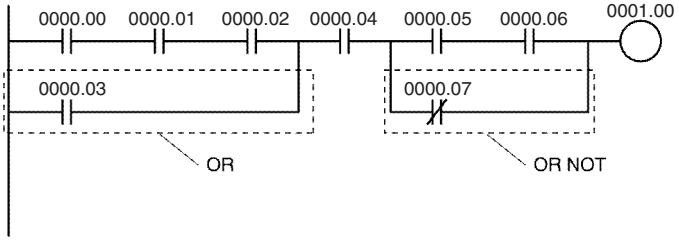
Flags

There are no flags affected by this instruction.

Precautions

Differentiate up (@) or differentiate down (%) can be specified for OR. If differentiate up (@) is specified, the execution condition is turned ON for one cycle only after the status of the operand bit goes from OFF to ON. If differentiate down (%) is specified, the execution condition is turned ON for one cycle only after the status of the operand bit goes from ON to OFF.

Example



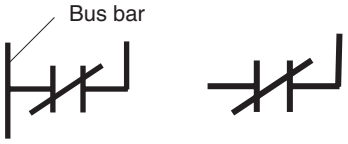
Instruction	Operand
LD	0000.00
AND	0000.01
AND	0000.02
OR	0000.03
AND	0000.04
LD	0000.05
AND	0000.06
OR NOT	0000.07
AND LD	---
OUT	0001.00

3-2-6 OR NOT: OR NOT

Purpose

Reverses the status of the specified bit and takes a logical OR with the current execution condition.

Ladder Symbol



Variations

Variations		
	Creates ON Each Cycle OR NOT Result is ON	OR NOT
	Creates ON Once for Upward Differentiation	@OR NOT
	Creates ON Once for Downward Differentiation	%OR NOT

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	OR NOT bit operand
CIO Area	CIO 0000.00 to CIO 6143.15
Work Area	W000.00 to W255.15
Auxiliary Bit Area	A000.00 to A959.15
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
Task Flag	TK0000
Condition Flags	ER, CY, N, OF, UF, >, =, <, >=, <>, <=, ON, OFF, AER
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
TR Area	---
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

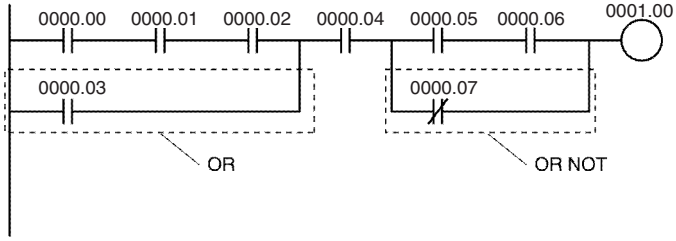
Description

OR NOT is used for a normally closed bit connected in parallel. A normally closed bit is configured to form a logical OR with a logic block beginning with a LOAD or LOAD NOT instruction (connected to the bus bar or at the beginning of the logic block). The specified bit in I/O memory is read.

Flags

There are no flags affected by this instruction.

Example



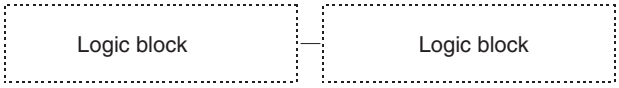
Instruction	Operand
LD	0000.00
AND	0000.01
AND	0000.02
OR	0000.03
AND	0000.04
LD	0000.05
AND	0000.06
OR NOT	0000.07
AND LD	---
OUT	0001.00

3-2-7 AND LOAD: AND LD

Purpose

Takes a logical AND between logic blocks.

Ladder Symbol



Variations

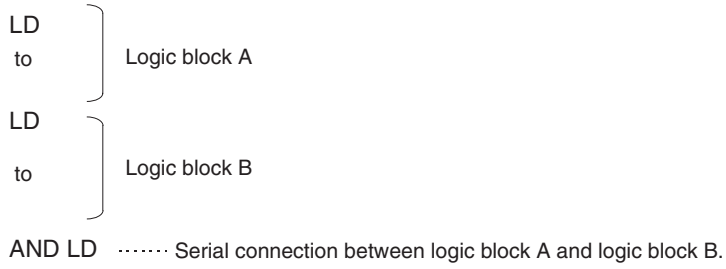
Variations	Creates ON Each Cycle AND Result is ON	AND LD

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

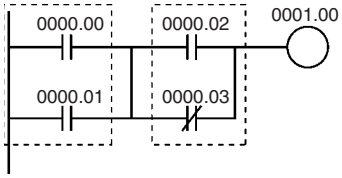
Description

AND LD connects in series the logic block just before this instruction with another logic block.



The logic block consists of all the instructions from a LOAD or LOAD NOT instruction until just before the next LOAD or LOAD NOT instruction on the same rungs.

In the following diagram, the two logic blocks are indicated by dotted lines. Studying this example shows that an ON execution condition will be produced when either of the execution conditions in the left logic block is ON (i.e., when either CIO 0000.00 or CIO 0000.01 is ON) **and** either of the execution conditions in the right logic block is ON (i.e., when either CIO 0000.02 is ON or CIO 0000.03 is OFF).



Coding

Address	Instruction	Operand
000000	LD	0000.00
000001	OR	0000.01
000002	LD	0000.02
000003	OR NOT	0000.03
000004	AND LD	---
000005	OUT	0001.00

Second LD: Used for first bit of next block connected in series to previous block.

Flags

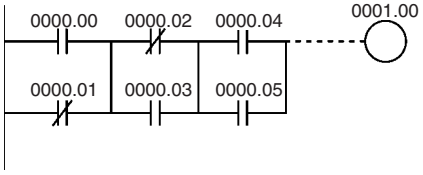
There are no flags affected by this instruction.

Precautions

Three or more logic blocks can be connected in series using this instruction to first connect two of the logic blocks and then to connect the next and subsequent ones in order. It is also possible to continue placing this instruction after three or more logic blocks and connect them together in series.

When a logic block is connected by AND LOAD or OR LOAD instructions, the total number of AND LOAD/OR LOAD instructions must match the total number of LOAD/LOAD NOT instructions minus 1. If they do not match, a program error will occur.

Example



Coding Example (1)

Instruction	Operand
LD	0000.00
OR NOT	0000.01
LD NOT	0000.02
OR	0000.03
AND LD	---
LD	0000.04
OR	0000.05
AND LD	---
. .	. .
OUT	0001.00

Coding Example (2)

Instruction	Operand
LD	0000.00
OR NOT	0000.01

Instruction	Operand
LD NOT	0000.02
OR	0000.03
LD	0000.04
OR	0000.05
. .	. .
AND LD	---
AND LD	---
. .	. .
OUT	0001.00

The AND LOAD instruction can be used repeatedly. In programming method (2) above, however, the number of AND LOAD instructions becomes one less than the number of LOAD and LOAD NOT instructions before that.

In method (2), make sure that the total number of LOAD and LOAD NOT instructions before AND LOAD is not more than eight. To use nine or more, program using method (1). If there are nine or more with method (2), then a program error will occur during the program check by the CX-Programmer.

3-2-8 OR LOAD: OR LD

Purpose

Takes a logical OR between logic blocks.

Ladder Symbol

Variations

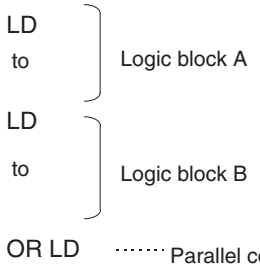
Variations	Creates ON Each Cycle AND Result is ON	OR LD

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Description

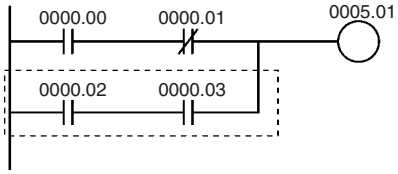
AND LD connects in parallel the logic block just before this instruction with another logic block.



The logic block consists of all the instructions from a LOAD or LOAD NOT instruction until just before the next LOAD or LOAD NOT instruction on the same rungs.

The following diagram requires an OR LOAD instruction between the top logic block and the bottom logic block. An ON execution condition would be produced either when CIO 0000.00 is ON and CIO 0000.01 is OFF or when

CIO 0000.02 and CIO 0000.03 are both ON. The operation of and mnemonic code for the OR LOAD instruction is exactly the same as those for a AND LOAD instruction except that the current execution condition is ORed with the last unused execution condition.



Coding

Address	Instruction	Operand
000100	LD	0000.00
000101	AND NOT	0000.01
000102	LD	0000.02
000103	AND	0000.03
000104	OR LD	---
000105	OUT	0005.01

Second LD: Used for first bit of next block connected in parallel to previous block.

Flags

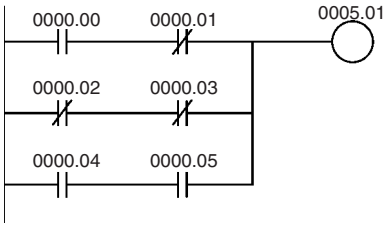
There are no flags affected by this instruction.

Precautions

Three or more logic blocks can be connected in parallel using this instruction to first connect two of the logic blocks and then to connect the next and subsequent ones in order. It is also possible to continue placing this instruction after three or more logic blocks and connect them together in parallel.

When a logic block is connected by AND LOAD or OR LOAD instructions, the total number of AND LOAD/OR LOAD instructions must match the total number of LOAD/LOAD NOT instructions minus 1. If they do not match, a programming error will occur.

Example



Coding Example (1)

Instruction	Operand
LD	0000.00
AND NOT	0000.01
LD NOT	0000.02
AND NOT	0000.03
OR LD	---
LD	0000.04
AND	0000.05
OR LD	---

Instruction	Operand
. .	. .
OUT	0005.01

Coding Example (2)

Instruction	Operand
LD	0000.00
AND NOT	0000.01
LD NOT	0000.02
AND NOT	0000.03
LD	0000.04
AND	0000.05
. .	. .
OR LD	---
OR LD	---
. .	. .
OUT	0005.01

The OR LOAD instruction can be used repeatedly. In programming method (2) above, however, the number of OR LOAD instructions becomes one less than the number of LOAD and LOAD NOT instructions before that.

In method (2), make sure that the total number of LOAD and LOAD NOT instructions before OR LOAD is not more than eight. To use nine or more, program using method (1). If there are nine or more with method (2), then a program error will occur during the program check by the Peripheral Device.

3-2-9 Differentiated Instructions

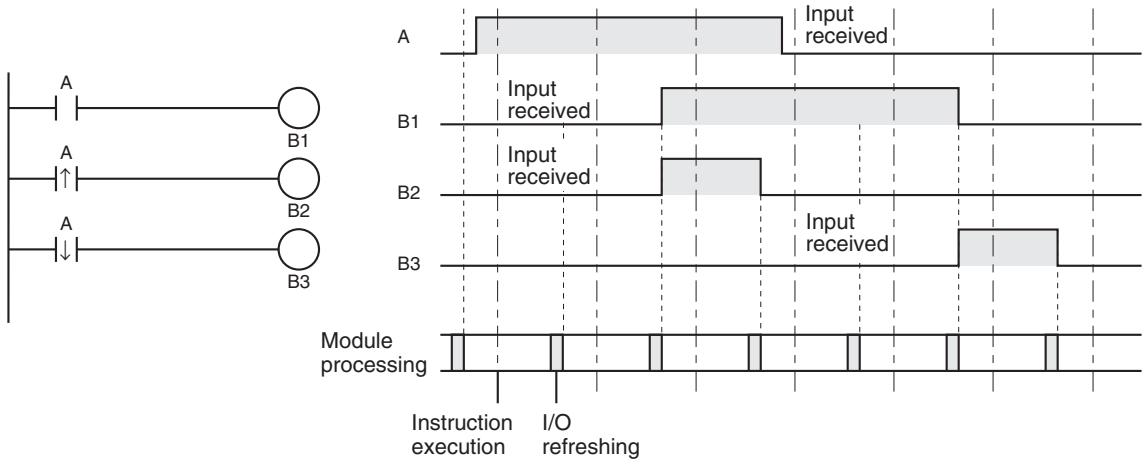
The LOAD, AND, and OR instructions have differentiated variations in addition to their ordinary forms. The I/O timing for data handled by instructions differs for ordinary and differentiated instructions.

Ordinary and differentiated instructions are executed using data input by previous I/O refresh processing, and the results are output with the next I/O processing. Here “I/O refreshing” means the data exchanged between the internal memory and the built-in I/O.

Instruction variation	Mnemonic	Function	I/O refresh
Ordinary	LD, AND, OR, LD NOT, AND NOT, OR NOT	The ON/OFF status of the specified bit is taken by the Module with cyclic refreshing, and it is reflected in the next instruction execution.	Cyclic refreshing
	OUT, OUT NOT	After the instruction is executed, the ON/OFF status of the specified bit is output with the next cyclic refreshing.	
Differentiated up	@LD, @AND, @OR	The instruction is executed once when the specified bit turns from OFF to ON and the ON state is held for one cycle.	
Differentiated down	%LD, %AND, %OR	The instruction is executed once when the specified bit turns from ON to OFF and the ON state is held for one cycle.	

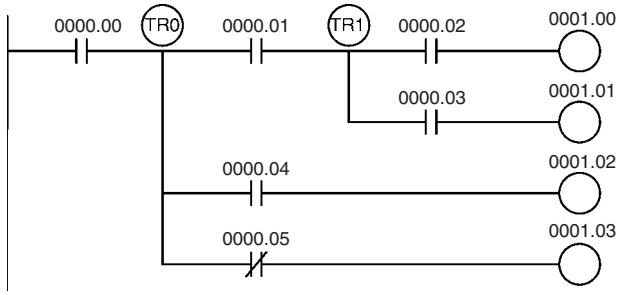
3-2-10 Operation Timing for I/O Instructions

The following chart shows the differences in the timing of instruction operations for a program configured from LD and OUT.



3-2-11 TR Bits

TR bits are used to temporarily retain the ON/OFF status of execution conditions in a program when programming in mnemonic code. They are not used when programming directly in ladder program form because the processing is automatically executed by the CX-Programmer. The following diagram shows a simple application using two TR bits.

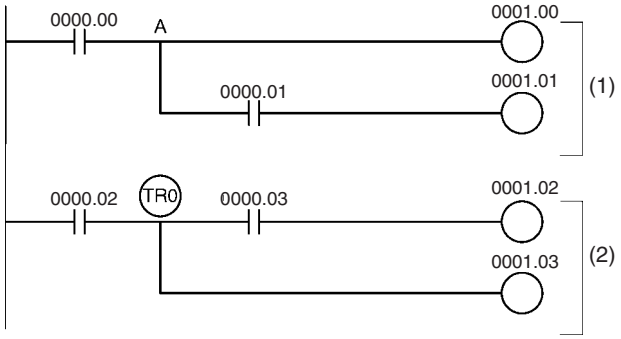


Address	Instruction	Operands
000000	LD	0000.00
000001	OUT	TR0
000002	AND	0000.01
000003	OUT	TR1
000004	AND	0000.02
000005	OUT	0001.00
000006	LD	TR1
000007	AND	0000.03
000008	OUT	0001.01
000009	LD	TR0
000010	AND	0000.04
000011	OUT	0001.02
000012	LD	TR0
000013	AND NOT	0000.05
000014	OUT	0001.03

Using TR0 to TR15

TR0 to TR15 are used only with LOAD and OUTPUT instructions. There are no restrictions on the order in which the bit addresses are used.

Sometimes it is possible to simplify a program by rewriting it so that TR bits are not required. The following diagram shows one case in which a TR bit is unnecessary and one in which a TR bit is required.



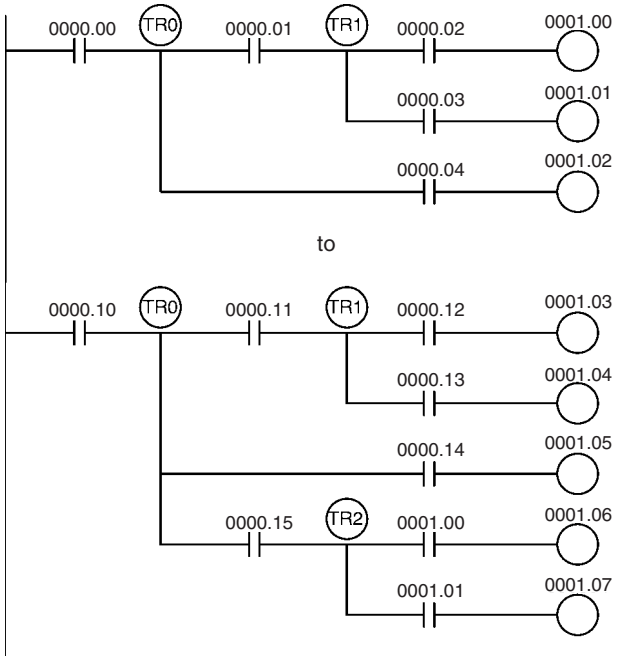
In instruction block (1), the ON/OFF status at point A is the same as for output CIO 0001.00, so AND 0000.01 and OUT 0001.01 can be coded without requiring a TR bit. In instruction block (2), the status of the branching point and that of output CIO 0001.02 are not necessarily the same, so a TR bit must be used. In this case, the number of steps in the program could be reduced by using instruction block (1) in place of instruction block (2).

TR0 to TR15 Considerations

TR bits are used only for retaining (OUT TR0 to TR15) and restoring (LD TR0 to TR15) the ON/OFF status of branching points in programs with many output branches. They are thus different from general bits, and cannot be used with AND or OR instructions, or with instructions that include NOT.

TR0 to TR15 output Duplication

A TR bit address cannot be repeated within the same block in a program with many output branches, as shown in the following diagram. It can, however, be used again in a different block.

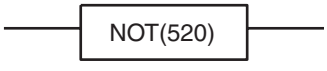


3-2-12 NOT: NOT(520)

Purpose

Reverses the execution condition.

Ladder Symbol



Variations

Variations	Reverses the Execution Condition Each Cycle	NOT(520)

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Description

NOT(520) is placed between an execution condition and another instruction to invert the execution condition.

Flags

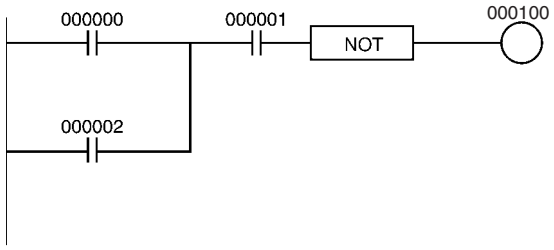
There are no flags affected by NOT(520)

Precautions

NOT(520) is an intermediate instruction, i.e., it cannot be used as a right-hand instruction. Be sure to program a right-hand instruction after NOT(520).

Example

NOT(520) reverses the execution condition in the following example.



The following table shows the operation of this program section.

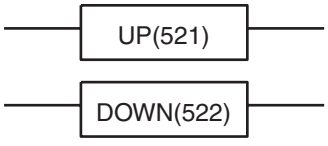
Input bit status			Output bit status
CIO 000000	CIO 000001	CIO 000002	CIO 000100
1	1	1	0
1	1	0	0
1	0	1	1
0	1	1	0
1	0	0	1
0	1	0	1
0	0	1	1
0	0	0	1

3-2-13 CONDITION ON/OFF: UP(521) and DOWN(522)

Purpose

UP(521) turns ON the execution condition for the next instruction for one cycle when the execution condition it receives goes from OFF to ON. DOWN(522) turns ON the execution condition for the next instruction for one cycle when the execution condition it receives goes from ON to OFF.

Ladder Symbols



Variations

Variations	Creates ON Once for Upward Differentiation	UP(521)

Variations	Creates ON Once for Downward Differentiation	UP(522)

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Description

UP(521) is placed between an execution condition and another instruction to turn the execution condition into an up-differentiated condition. UP(521) causes the connecting instruction to be executed just once when the execution condition goes from OFF to ON.

DOWN(522) is placed between an execution condition and another instruction to turn the execution condition into a down-differentiated condition. DOWN(522) causes the connecting instruction to be executed just once when the execution condition goes from ON to OFF.

The DIFU(013) and DIFD(014) instructions can also be used for the same purpose, but they require work bits. UP(521) and DOWN(522) simplify programming by reducing the number of work bits and program addresses needed.

Flags

There are no flags affected by UP(521) and DOWN(522).

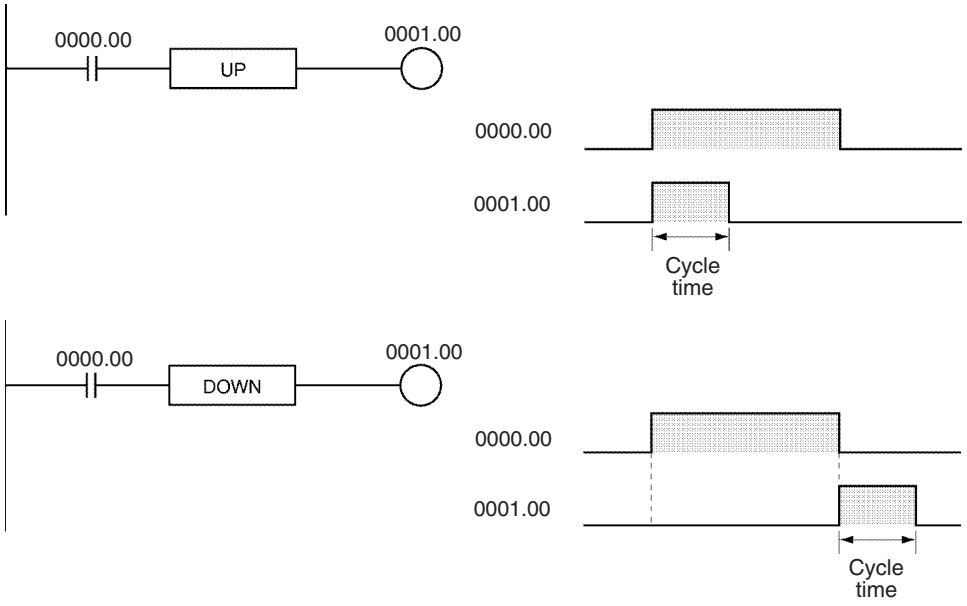
Precautions

UP(521) and DOWN(522) are intermediate instructions, i.e., they cannot be used as right-hand instructions. Be sure to program a right-hand instruction after UP(521) or DOWN(522).

The operation of UP(521) and DOWN(522) depends on the execution condition for the instruction as well as the execution condition for the program section when it is programmed in an interlocked program section, a jumped program section, or a subroutine. Refer to *3-4-3 INTERLOCK and INTERLOCK CLEAR: IL(002) and ILC(003), 3-4-4 JUMP and JUMP END: JMP(004) and JME(005)*, and *3-19 Interrupt Control Instructions* for details.

Examples

When CIO 0000.00 goes from OFF to ON in the following example, CIO 0001.00 is turned ON for just one cycle.



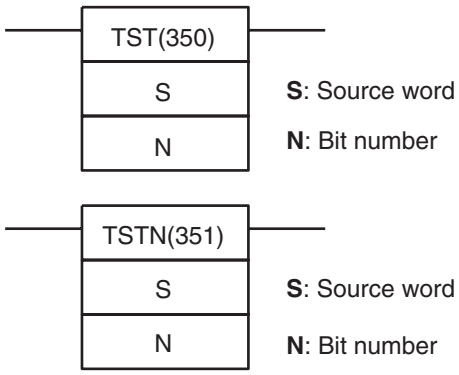
3-2-14 BIT TEST: TST(350) and TSTN(351)

Purpose

LD TST(350), AND TST(350), and OR TST(350) are used in the program like LD, AND, and OR; the execution condition is ON when the specified bit in the specified word is ON, and OFF when the bit is OFF.

LD TSTN(351), AND TSTN(351), and OR TSTN(351) are used in the program like LD NOT, AND NOT, and OR NOT; the execution condition is OFF when the specified bit in the specified word is ON, and ON when the bit is OFF.

Ladder Symbols



Variations

Variations	Executed Each Cycle	TST(350)

Variations	Executed Each Cycle	TSTN(351)

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

N: Bit number

The bit number must be between 0000 and 000F hexadecimal or between &0000 and &0015 decimal. Only the rightmost bit (0 to F hexadecimal) of the contents of the word is valid when a word address is specified.

Operand Specifications

Area	S	N
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	
Auxiliary Bit Area	A000 to A959	
Timer Area	T0000 to T0255	
Counter Area	C0000 to C0255	
DM Area	D00000 to D32767	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	#0000 to #000F (binary) or &0 to &15
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 , IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

LD TST(350), AND TST(350), and OR TST(350) can be used in the program like LD, AND, and OR; the execution condition is ON when the specified bit in the specified word is ON and OFF when the bit is OFF. Unlike LD, AND, and OR, bits in the DM area can be used as operands in TST(350).

LD TSTN(351), AND TSTN(351), and OR TSTN(351) can be used in the program like LD NOT, AND NOT, and OR NOT; the execution condition is OFF

when the specified bit in the specified word is ON and ON when the bit is OFF. Unlike LD NOT, AND NOT, and OR NOT, bits in the DM area can be used as operands in TSTN(351).

Flags

Name	Label	Operation
Error Flag	ER	Unchanged
Equals Flag	=	Unchanged
Negative Flag	N	Unchanged

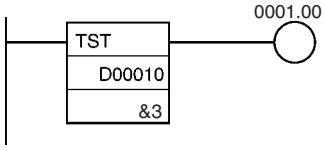
Precautions

TST(350) and TSTN(351) are intermediate instructions, i.e., they cannot be used as right-hand instructions. Be sure to program a right-hand instruction after TST(350) or TSTN(351).

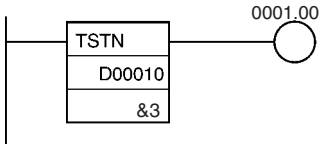
Examples

LD TST(350) and LD TSTN(351)

In the following example, CIO 0001.00 is turned ON when bit 3 of D00010 is ON.

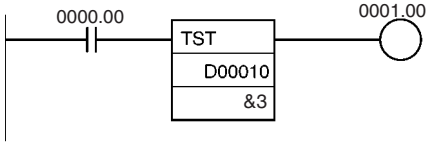


In the following example, CIO 0001.00 is turned ON when bit 3 of D00010 is OFF.

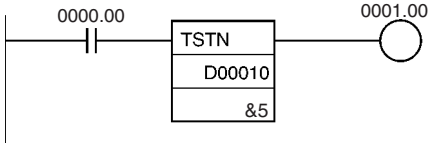


AND TST(350) and AND TSTN(351)

In the following example, CIO 0001.00 is turned ON when CIO 0000.00 and bit 3 of D00010 are both ON.

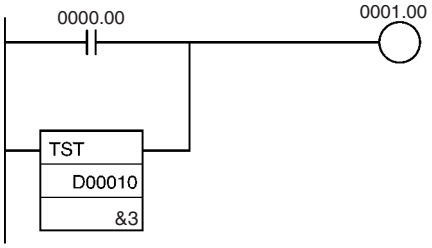


In the following example, CIO 0001.00 is turned ON when CIO 0000.00 is ON and bit 5 of D00010 is OFF.

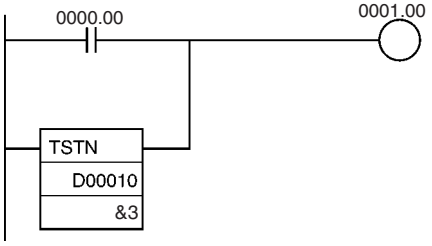


OR TST(350) and OR TSTN(351)

In the following example, CIO 0001.00 is turned ON when CIO 0000.00 or bit 3 of D00010 is ON.



In the following example, CIO 0001.00 is turned ON when CIO 0000.00 is ON or bit 3 of D00010 is OFF.



3-3 Sequence Output Instructions

This section describes the sequence output instructions.

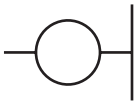
Instruction	Mnemonic	Function code	Page
OUTPUT	OUT	---	117
OUTPUT NOT	OUT NOT	---	118
KEEP	KEEP	011	119
DIFFERENTIATE UP	DIFU	013	122
DIFFERENTIATE DOWN	DIFD	014	122
SET	SET	---	125
RESET	RSET	---	125
MULTIPLE BIT SET	SETA	530	126
MULTIPLE BIT RESET	RSTA	531	126
SINGLE BIT SET	SETB	532	129
SINGLE BIT RESET	RSTB	533	129
SINGLE BIT OUTPUT	OUTB	534	132

3-3-1 OUTPUT: OUT

Purpose

Outputs the result (execution condition) of the logical processing to the specified bit.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	OUT
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	OK

Operand Specifications

Area	OUT bit operand
CIO Area	CIO 0000.00 to CIO 6143.15
Work Area	W000.00 to W255.15
Auxiliary Bit Area	A448.00 to A959.15
Timer Area	---
Counter Area	---
TR Area	TR0 to TR15
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	---
Data Resisters	---

Area	OUT bit operand
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

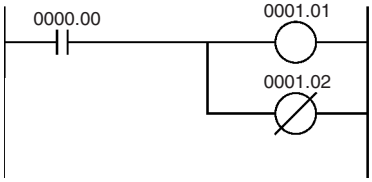
Description

The status of the execution condition (power flow) is written to the specified bit in I/O memory.

Flags

There are no flags affected by this instruction.

Example



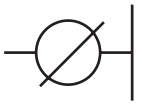
Instruction	Operand
LD	0000.00
OUT	0001.01
OUT NOT	0001.02

3-3-2 OUTPUT NOT: OUT NOT

Purpose

Reverses the result (execution condition) of the logical processing, and outputs it to the specified bit.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	OUT NOT
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	OK

Operand Specifications

Area	OUT bit operand
CIO Area	CIO 0000.00 to CIO 6143.15
Work Area	W000.00 to W255.15
Auxiliary Bit Area	A448.00 to A959.15
Timer Area	---
Counter Area	---
TR Area	TR0 to TR15
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---

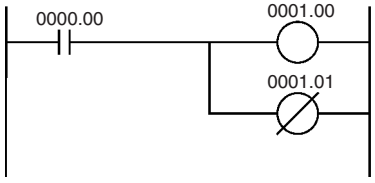
Area	OUT bit operand
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Note The TR Area is used only when programming with mnemonic code. It is not used for ladder programs.

Description The status of the execution condition (power flow) is reversed and written to a specified bit in I/O memory.

Flags There are no flags affected by this instruction.

Example

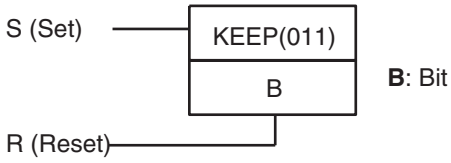


Instruction	Operand
LD	0000.00
OUT	0001.00
OUT NOT	0001.01

3-3-3 KEEP: KEEP(011)

Purpose Operates as a latching relay.

Ladder Symbol



B: Bit

Variations

Variations		
	Executed Each Cycle for ON Condition	KEEP(011)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	OK

Operand Specifications

Area	B
CIO Area	CIO 0000.00 to CIO 6143.15
Work Area	W000.00 to W255.15
Auxiliary Bit Area	A448.00 to A959.15
Timer Area	---

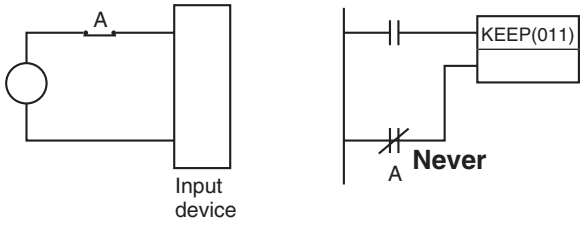
Area	B
Counter Area	---
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Flags

There are no flags affected by this instruction.

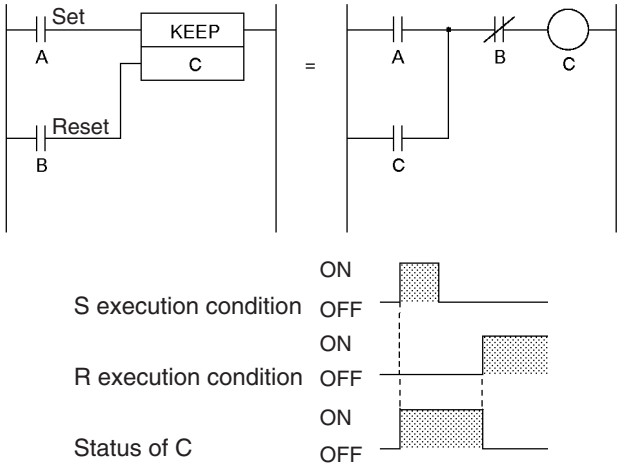
Precautions

Do not use the input from an external device that has a normally closed contact for the Reset input of KEEP(011). If the AC power supply is interrupted or a momentary power interruption occurs, there will be a delay in shutting down the FQM1's internal DC power supply, which can cause the operand bit of KEEP(011) to be reset.

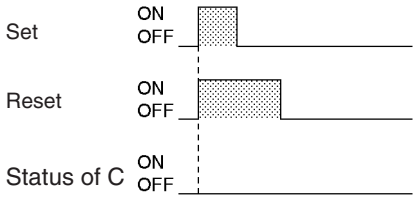


Description

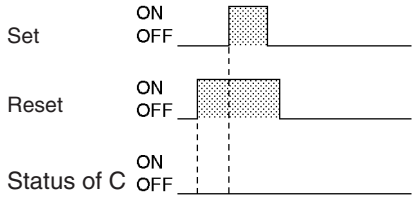
When S turns ON, the designated bit will go ON and stay ON until reset, regardless of whether S stays ON or goes OFF. When R turns ON, the designated bit will go OFF. The relationship between execution conditions and KEEP(011) bit status is shown below.



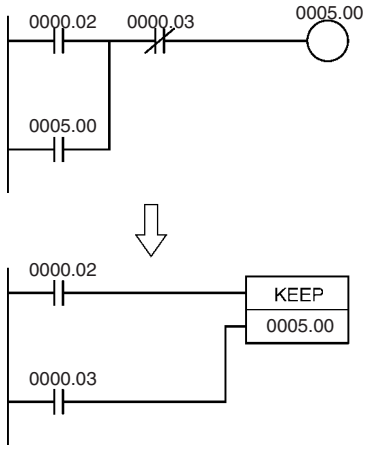
If S and R are ON simultaneously, the reset input takes precedence.



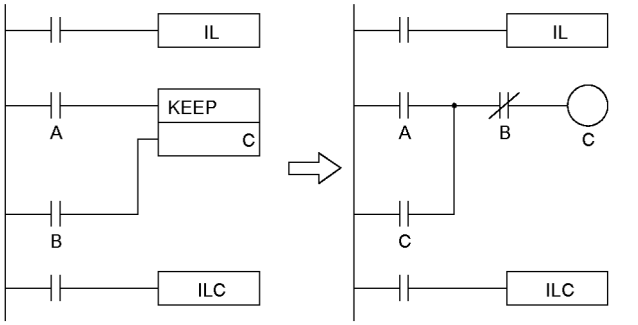
The set input (S) cannot be received while R is ON.



KEEP(011) operates like the self-maintaining bit, but a self-maintaining bit programmed with KEEP(011) requires one less instruction.



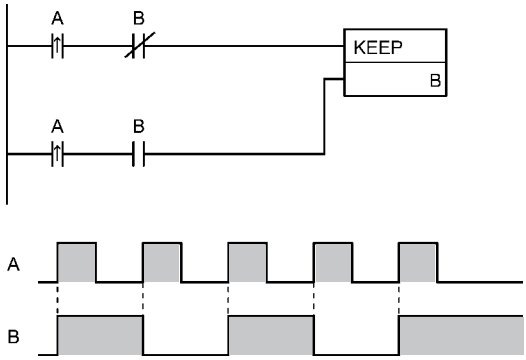
Self-maintaining bits programmed with KEEP(011) will maintain status even in an interlock program section, unlike the self-maintaining bit programmed without KEEP(011).



Output bit C will maintain its previous status in an interlock.

Output bit C will be turned OFF in an interlock.

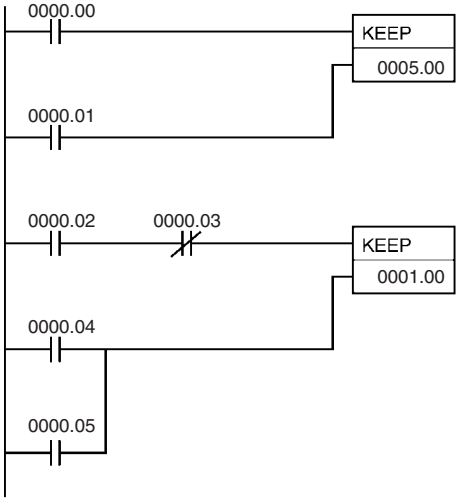
KEEP(011) can be used to create flip-flops as shown below.



Example

When CIO 0000.00 goes ON in the following example, CIO 0005.00 is turned ON. CIO 0005.00 remains ON until CIO 0000.01 goes ON.

When CIO 0000.02 goes ON and CIO 0000.03 goes OFF in the following example, CIO 0001.00 is turned ON. CIO 0001.00 remains ON until CIO 0000.04 or CIO 0000.05 goes ON.



Coding

Address	Instruction	Operand
000100	LD	0000.00
000101	LD	0000.01
000102	KEEP (011)	0005.00
000103	LD	0000.02
000104	AND NOT	0000.03
000105	LD	0000.04
000106	OR	0000.05
000107	KEEP (011)	0001.00

Note KEEP(011) is input in different orders on in ladder and mnemonic form. In ladder form, input the set input, KEEP(011), and then the reset input. In mnemonic form, input the set input, the reset input, and then KEEP(011).

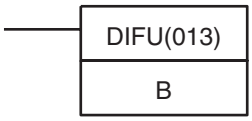
3-3-4 DIFFERENTIATE UP/DOWN: DIFU(013) and DIFD(014)

Purpose

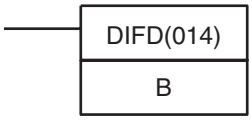
DIFU(013) turns the designated bit ON for one cycle when the execution condition goes from OFF to ON (rising edge).

DIFD(014) turns the designated bit ON for one cycle when the execution condition goes from ON to OFF (falling edge).

Ladder Symbols



B: Bit



B: Bit

Variations

Variations		
	Executed Each Cycle for ON Condition	Not supported
	Executed Once for Upward Differentiation	DIFU(013)
	Executed Once for Downward Differentiation	Not supported

Variations		
	Executed Each Cycle for ON Condition	Not supported
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	DIFD(014)

Applicable Program Areas

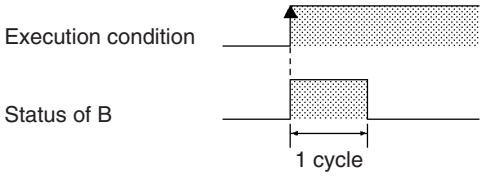
Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	OK

Operand Specifications

Area	B
CIO Area	CIO 0000.00 to CIO 6143.15
Work Area	W000.00 to W255.15
Auxiliary Bit Area	A448.00 to A959.15
Timer Area	---
Counter Area	---
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

When the execution condition goes from OFF to ON, DIFU(013) turns B ON. When DIFU(013) is reached in the next cycle, B is turned OFF.



When the execution condition goes from ON to OFF, DIFD(014) turns B ON. When DIFD(014) is reached in the next cycle, B is turned OFF.

Flags

No flags are affected by DIFU(013) and DIFD(014).

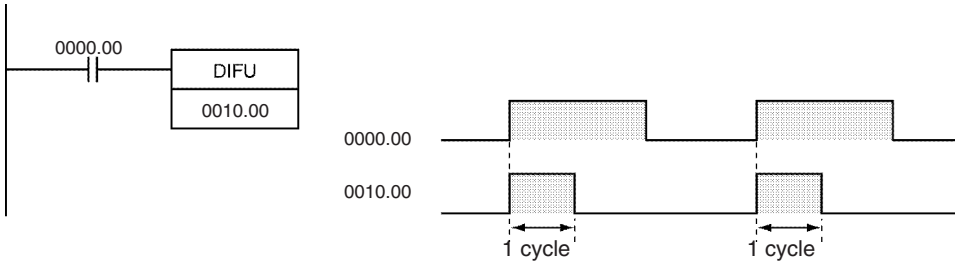
Precautions

The operation of DIFU(013) or DIFD(014) depends on the execution condition for the instruction itself as well as the execution condition for the program section when it is programmed in an interlocked program section, a jumped program section, or a subroutine. Refer to *3-4-3 INTERLOCK and INTERLOCK CLEAR: IL(002) and ILC(003), 3-4-4 JUMP and JUMP END: JMP(004) and JME(005)*, and *3-19 Interrupt Control Instructions* for details.

Examples

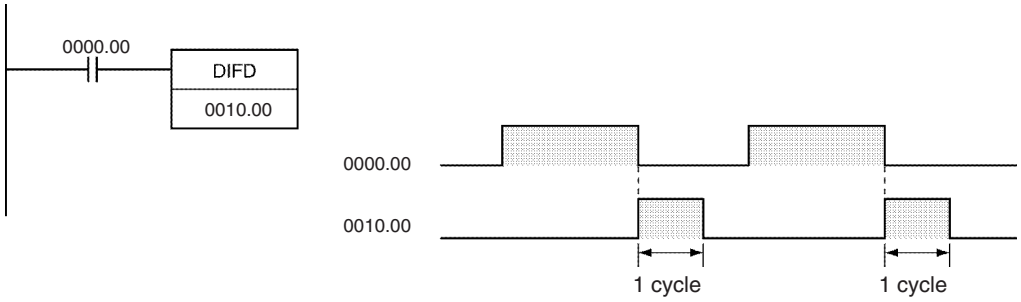
Operation of DIFU(013)

When CIO 0000.00 goes from OFF to ON in the following example, CIO 0010.00 is turned ON for one cycle.



Operation of DIFD(014)

When CIO 0000.00 goes from ON to OFF in the following example, CIO 0010.00 is turned ON for one cycle.

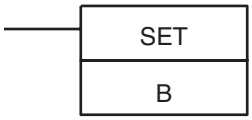


3-3-5 SET and RESET: SET and RSET

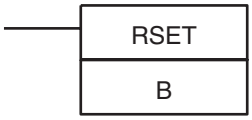
Purpose

SET turns the operand bit ON when the execution condition is ON.
RSET turns the operand bit OFF when the execution condition is ON.

Ladder Symbols



B: Bit



B: Bit

Variations

Variations		
	Executed Each Cycle for ON Condition	SET
	Executed Once for Upward Differentiation	@SET
	Executed Once for Downward Differentiation	%SET

Variations		
	Executed Each Cycle for ON Condition	RSET
	Executed Once for Upward Differentiation	@RSET
	Executed Once for Downward Differentiation	%RSET

Applicable Program Areas

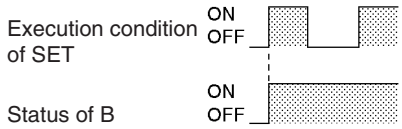
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

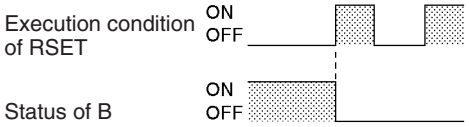
Area	B
CIO Area	CIO 0000.00 to CIO 6143.15
Work Area	W000.00 to W255.15
Auxiliary Bit Area	A448.00 to A959.15
Timer Area	---
Counter Area	---
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

SET turns the operand bit ON when the execution condition is ON, and does not affect the status of the operand bit when the execution condition is OFF. Use RSET to turn OFF a bit that has been turned ON with SET.



RSET turns the operand bit OFF when the execution condition is ON, and does not affect the status of the operand bit when the execution condition is OFF. Use SET to turn ON a bit that has been turned OFF with RSET.



The set and reset inputs for a KEEP(011) instruction must be programmed with the instruction, but the SET and RSET instructions can be programmed completely independently. Furthermore, the same bit may be used as the operand in any number of SET or RSET instructions.

Flags

No flags are affected by SET and RSET.

Precautions

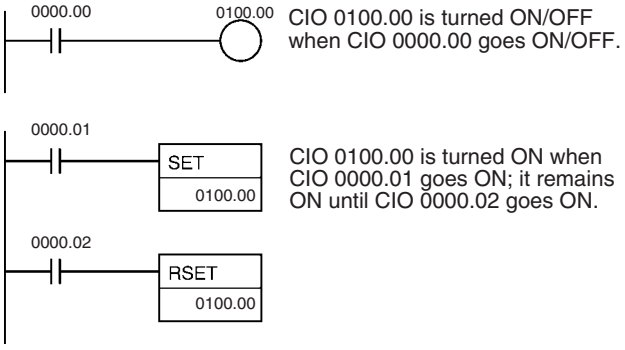
SET and RSET cannot be used to set and reset timers and counters.

When SET or RSET is programmed between IL(002) and ILC(003) or JMP(004) and JME(005), the status of the specified bit will not be changed if the execution condition for IL(002) or JMP(004) is OFF.

Example

Differences between OUT/OUT NOT and SET/RSET

The operation of SET differs from that of OUT because the OUT instruction turns the operand bit OFF when its execution condition is OFF. Likewise, RSET differs from OUT NOT because OUT NOT turns the operand bit ON when its execution condition is OFF.

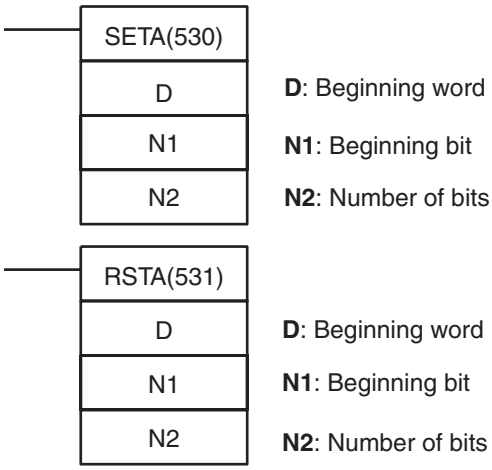


3-3-6 MULTIPLE BIT SET/RESET: SETA(530)/RSTA(531)

Purpose

SETA(530) turns ON the specified number of consecutive bits.
RSTA(531) turns OFF the specified number of consecutive bits.

Ladder Symbols



Variations

Variations		
Variations	**Executed Each Cycle for ON Condition**	SETA(530)
	Executed Once for Upward Differentiation	@SETA(530)
	Executed Once for Downward Differentiation	Not supported

Variations		
Variations	**Executed Each Cycle for ON Condition**	RSTA(531)
	Executed Once for Upward Differentiation	@RSTA(531)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

D: Beginning Word

Specifies the first word in which bits will be turned ON or OFF.

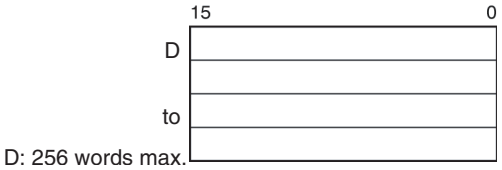
N1: Beginning Bit

Specifies the first bit which will be turned ON or OFF. N1 must be #0000 to #000F (&0 to &15).

N2: Number of Bits

Specifies the number of bits which will be turned ON or OFF. N2 must be #0000 to #FFFF (&0 to &65535).

Note The bits being turned ON or OFF must be in the same data area. (The range of words is roughly D to D+N2÷16.)



Operand Specifications

Area	D	N1	N2
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A448 to A959	A000 to A959	
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		

Area	D	N1	N2
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---	#0000 to #000F (binary) or &0 to &15	#0000 to #FFFF (binary) or &0 to &65535
Data Registers	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to , –(– –) IR15		

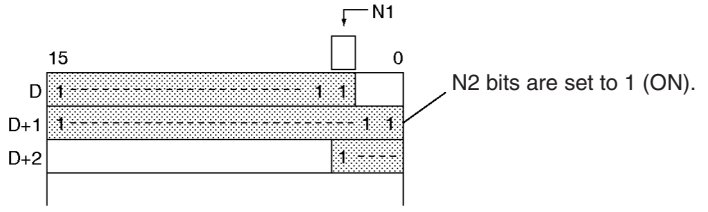
Description

The operation of SETA(530) and RSTA(531) are described separately below.

Operation of SETA(530)

SETA(530) turns ON N2 bits, beginning from bit N1 of D, and continuing to the left (more-significant bits). All other bits are left unchanged. (No changes will be made if N2 is set to 0.)

Bits turned ON by SETA(530) can be turned OFF by any other instructions, not just RSTA(531).

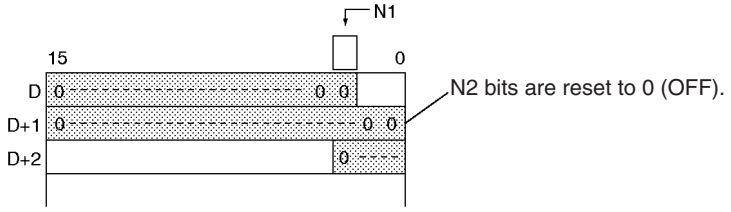


SETA(530) can be used to turn ON bits in data areas that are normally accessed by words only, such as the DM area.

Operation of RSTA(531)

RSTA(531) turns OFF N2 bits, beginning from bit N1 of D, and continuing to the left (more-significant bits). All other bits are left unchanged. (No changes will be made if N2 is set to 0.)

Bits turned OFF by RSTA(531) can be turned ON by any other instructions, not just SETA(530).



RSTA(531) can be used to turn OFF bits in data areas that are normally accessed by words only, such as the DM area.

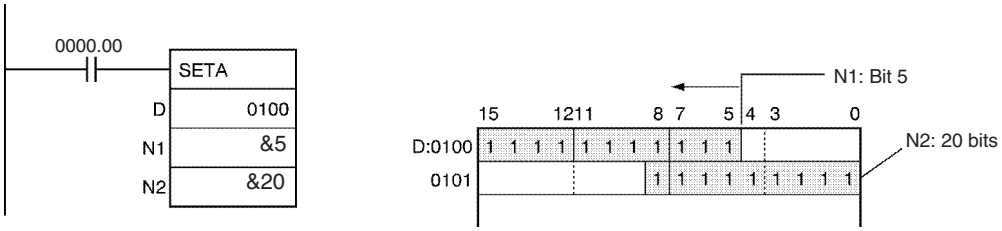
Flags

Name	Label	Operation
Error Flag	ER	ON if N1 is not within the specified range of 0000 to 000F. OFF in all other cases.

Examples

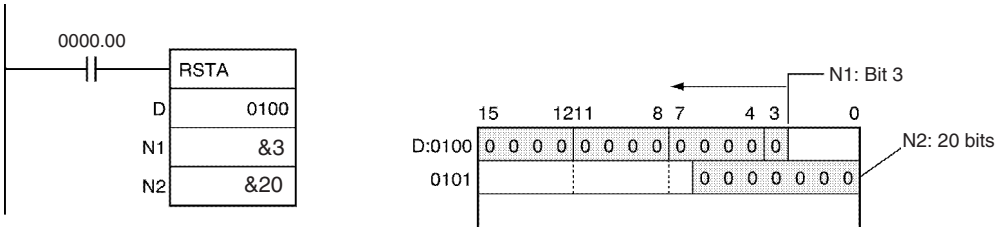
SETA(530) Example

When CIO 0000.00 is turned ON in the following example, the 20 bits (0014 hexadecimal) beginning with bit 5 of CIO 0100 are turned ON.



RSTA(531) Example

When CIO 0000.00 is turned ON in the following example, the 20 bits (0014 hexadecimal) beginning with bit 3 of CIO 0100 are turned OFF.

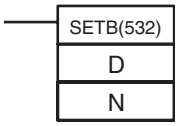


3-3-7 SINGLE BIT SET/RESET: SETB(532)/RSTB(533)

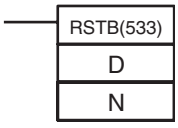
Purpose

SETB(532) turns ON the specified bit.
RSTB(533) turns OFF the specified bit.

Ladder Symbols



D: Word address
N: Bit number



D: Word address
N: Bit number

Variations

Variations	**Executed Each Cycle for ON Condition**	SETB(532)
	Executed Once for Upward Differentiation	@SETB(532)
	Executed Once for Downward Differentiation	Not supported
Combined Variations	**Executed Once and Bit Refreshed Immediately for Upward Differentiation (See note.)**	!@SETB(532)
	Executed Once and Bit Refreshed Immediately for Downward Differentiation	Not supported
Variations	**Executed Each Cycle for ON Condition**	RSTB(533)
	Executed Once for Upward Differentiation	@RSTB(533)
	Executed Once for Downward Differentiation	Not supported
Combined Variations	**Executed Once and Bit Refreshed Immediately for Upward Differentiation (See note.)**	!@RSTB(533)
	Executed Once and Bit Refreshed Immediately for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

D: Word Address

Specifies the word in which the bit will be turned ON or OFF.

N: Beginning Bit

Specifies the bit which will be turned ON or OFF. N must be #0000 to #000F (&0 to &15).

Operand Specifications

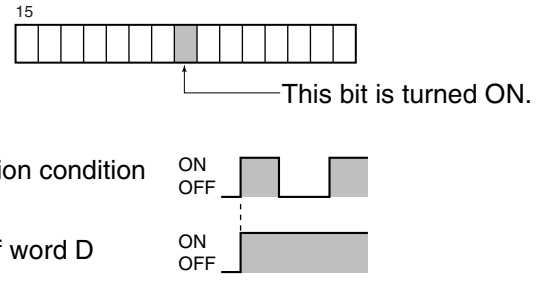
Area	D	N
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	
Auxiliary Bit Area	A448 to A959	A000 to A959
Timer Area	T0000 to T255	
Counter Area	C0000 to C255	
DM Area	D00000 to D32767	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	#0000 to #000F (binary) or &0 to &15
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to , –(– –) IR15	

Description

The functions of SETB(532) and RSTB(533) are described separately below.

Operation of SETB(532)

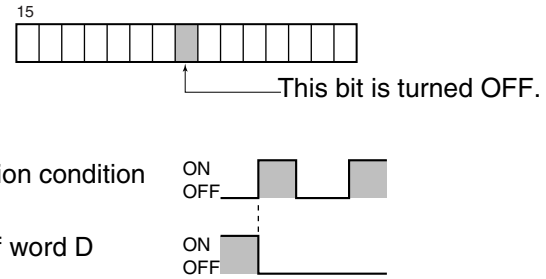
SETB(532) turns ON bit N of word D when the execution condition is ON. The status of the bit is not affected when the execution condition is OFF. Unlike SET, SETB(532) can turn ON a bit in the DM area.



Bits turned ON by SETB(532) can be turned OFF by any other instruction, not just RSTB(533).

Operation of RSTB(533)

RSTB(533) turns OFF bit N of word D when the execution condition is ON. The status of the bit is not affected when the execution condition is OFF. (Use SETB(532) to turn ON the bit.) Unlike RST, RSTB(533) can turn OFF a bit in the DM area.



Bits turned OFF by RSTB(533) can be turned ON by any other instruction, not just SETB(532).

Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0000 to 000F (&0 to &15). OFF in all other cases.

Precautions

SETB(532) and RSTB(533) cannot set/reset timers and counters.

When SETB(532) or RSTB(533) is programmed between IL(002) and ILC(003) or JMP(004) and JME(005), the status of the specified bit will not be changed if the program section is interlocked or jumped, i.e., when the interlock condition or jump condition is OFF.

SETB(532) and RSTB(533) have immediate refreshing variations (!SETB(532) and !RSTB(533)). When an external output bit has been specified in one of these instructions, any changes to the specified bit will be refreshed when the instruction is executed and reflected immediately in the output bit. (The changes will not be reflected immediately if the bit is allocated to a Group-2 High-density I/O Unit, High-density Special I/O Unit, or a Unit mounted in a SYSMAC BUS Remote I/O Slave Rack.)

Differences between SET/RSET and SETB(532)/RSTB(533)

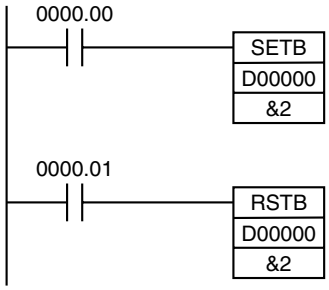
The SET and RSET instructions operate somewhat differently from SETB(532) and RSTB(533).

1. The instructions operate in the same way when the specified bit is in the CIO, W, H, or A Area.
2. The SETB(532) and RSTB(533) instructions can control bits in the DM Area, unlike SET and RSET.

Differences between OUTB(534) and SETB(532)/RSTB(533)

The OUTB(534) instruction operates somewhat differently from SETB(532) and RSTB(533).

1. The SETB(532) and RSTB(533) instructions change the status of the specified bit only when their execution condition is ON. These instructions have no effect on the status of the specified bit when their execution condition is OFF.
2. The OUTB(534) instruction turns ON the specified bit when its execution condition is ON and turns OFF the specified bit when its execution condition is OFF.
3. The set and reset inputs for a KEEP(011) instruction must be programmed with the instruction, but the SETB(532) and RSTB(533) instructions can be programmed completely independently. Furthermore, the same bit may be used as the operand in any number of SETB(532) and RSTB(533) instructions.

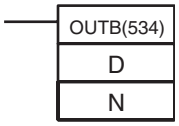


3-3-8 SINGLE BIT OUTPUT: OUTB(534)

Purpose

OUTB(534) outputs the status of the instruction's execution condition to the specified bit. OUTB(534) can control a bit in the DM Area, unlike OUT.

Ladder Symbols



D: Word address
N: Bit number

Variations

Variations		
	Executed Each Cycle for ON Condition	OUTB(534)
	Executed Once for Upward Differentiation	@OUTB(534)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	OK

Operands

D: Word Address

Specifies the word containing the bit to be controlled.

N: Beginning Bit

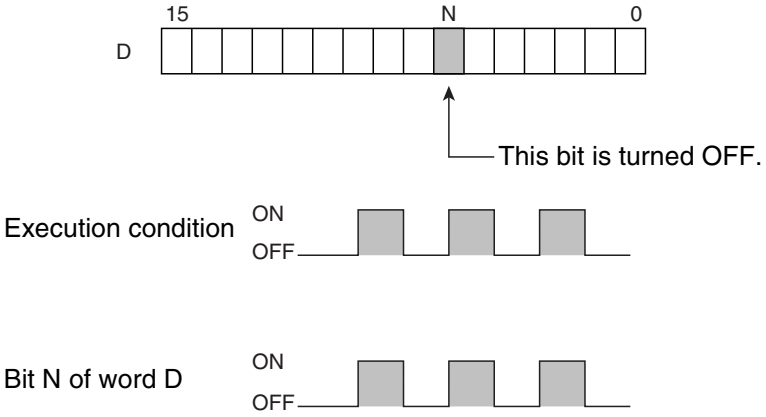
Specifies the bit to be controlled. N must be #0000 to #000F (&0 to &15).

Operand Specifications

Area	D	N
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	
Auxiliary Bit Area	A448 to A959	A000 to A959
Timer Area	T0000 to T255	
Counter Area	C0000 to C255	
DM Area	D00000 to D32767	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	#0000 to #000F (binary) or &0 to &15
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to , –(– –) IR15	

Description

When the execution condition is ON, OUTB(534) turns ON bit N of word D. When the execution condition is OFF, OUTB(534) turns OFF bit N of word D.



If the immediate refreshing version is not used, the status of the execution condition (power flow) is written to the specified bit in I/O memory. If the immediate refreshing version is used, the status of the execution condition (power flow) is written to the Basic Output Unit's output terminal as well as the output bit in I/O memory.

Flags

There are no flags affected by this instruction.

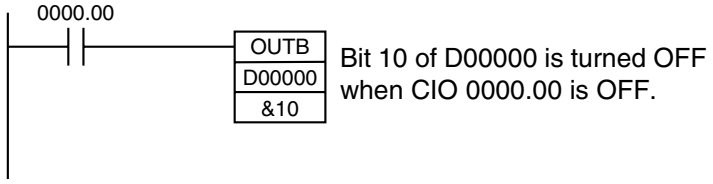
Precautions

Immediate refreshing (!OUTB(534)) can be specified. An immediate refresh instruction updates the status of the output terminal just after the instruction is executed on an output bit allocated to a Basic Output Unit (but not for C200H Group 2 Multi-point Output Units or Basic Output Units on Slave Racks), at the same time as it writes the status of the execution condition (power flow) to the specified output bit in I/O memory.

When OUTB(534) is programmed between IL(002) and ILC(003), the specified bit will be turned OFF if the program section is interlocked. (This is the same as an OUT instruction in an interlocked program section.)

When a word is specified for the bit number (N), only bits 00 to 03 of N are used. For example, if N contains FFFA hex, OUTB(534) will control bit 10 of word D.

Example



3-4 Sequence Control Instructions

This section describes the sequence control instructions.

Instruction	Mnemonic	Function code	Page
END	END	001	134
NO OPERATION	NOP	000	134
INTERLOCK/INTERLOCK CLEAR	IL/ILC	002/003	135
JUMP/JUMP END	JMP/JME	004/005	138
CONDITIONAL JUMP	CJP/CJPN	510/511	141
MULTIPLE JUMP/JUMP END	JMP0/JME0	515/516	145
FOR-NEXT LOOPS	FOR	512	147
	NEXT	513	147
BREAK LOOP	BREAK	514	150

3-4-1 END: END(001)

Purpose

Indicates the end of a program.

Ladder Symbol

Variations

Variations	Executed Each Cycle for ON Condition	END(001)

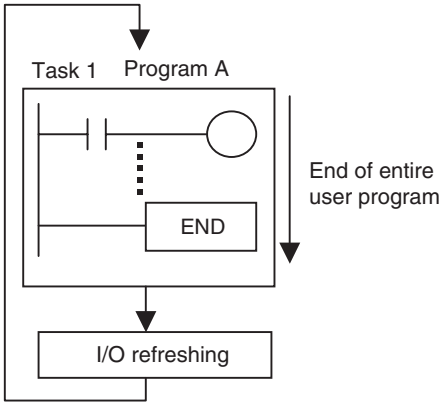
Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	Not allowed	Not allowed	OK

Description

END(001) completes the execution of a program for that cycle. No instructions written after END(001) will be executed.

Execution proceeds to the program with the next task number. When the program being executed has the highest task number in the program, END(001) marks the end of the overall main program.



Precautions

Always place END(001) at the end of each program. A programming error will occur if there is not an END(001) instruction in the program.

3-4-2 NO OPERATION: NOP(000)

Purpose

This instruction has no function. (No processing is performed for NOP(000).)

Ladder Symbol

There is no ladder symbol associated with NOP(000).

Variations

Variations	Executed Each Cycle for ON Condition	NOP(000)

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Description No processing is performed for NOP(000), but this instruction can be used to set aside lines in the program where instructions will be inserted later. When the instructions are inserted later, there will be no change in program addresses.

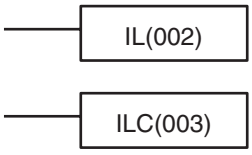
Flags No flags are affected by NOP(000).

Precautions NOP(000) can only be used with mnemonic displays, not with ladder programs.

3-4-3 INTERLOCK and INTERLOCK CLEAR: IL(002) and ILC(003)

Purpose Interlocks all outputs between IL(002) and ILC(003) when the execution condition for IL(002) is OFF. IL(002) and ILC(003) are normally used in pairs.

Ladder Symbols



Variations

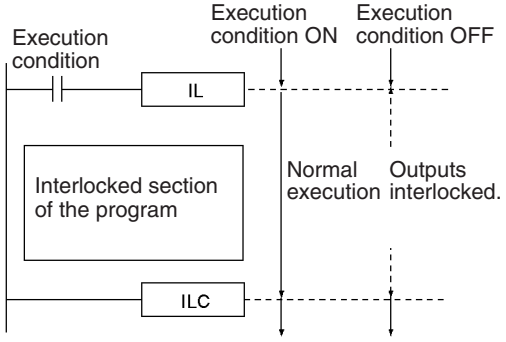
Variations	Interlocks when OFF/Does Not interlock when ON	IL(002)

Variations	Executed Each Cycle for ON Condition	ILC(003)

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	Not allowed	OK	OK

Description When the execution condition for IL(002) is OFF, the outputs for all instructions between IL(002) and ILC(003) are interlocked. When the execution condition for IL(002) is ON, the instructions between IL(002) and ILC(003) are executed normally.



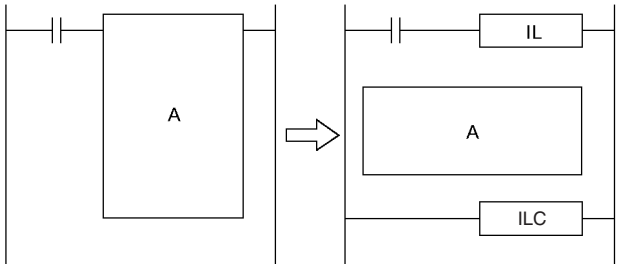
The following table shows the treatment of various outputs in an interlocked section between IL(002) and ILC(003).

Instruction		Treatment
Bits specified in OUT or OUT NOT		OFF
TIM, TIMH(015), and TMHH(540)	Completion Flag	OFF (reset)
	PV	Time set value (reset)
Bits/words specified in all other instructions (See note.)		Retain previous status.

Note Bits and words in all other instructions including SET, RSET, CNT, CNTR(012), SFT, and KEEP(011) retain their previous status.

If there are bits which you want to keep ON in an interlocked program section, set these bits to ON with SET just before IL(002).

It is often more efficient to switch a program section with IL(002) and ILC(003). When several processes are controlled with the same execution condition, it takes fewer program steps to put these processes between IL(002) and ILC(003).



The following table shows the differences between IL(002)/ILC(003) and JMP(004)/JME(005).

Item	Treatment in IL(002)/ILC(003)	Treatment in JMP(004)/JME(005)
Instruction execution	Instructions other than OUT, OUT NOT, and timer instructions are not executed.	No instructions are executed.
Output status in instructions	Except for outputs in OUT, OUT NOT, and timer instructions, all outputs retain their previous status.	All outputs retain their previous status.
Bits in OUT, OUT NOT	OFF	All outputs retain their previous status.
Status of timer instructions	Reset	Operating timers (TIM, TIMH(015), TMHH(540), only) continue timing because the PVs are updated even when the timer instruction is not being executed.

Flags

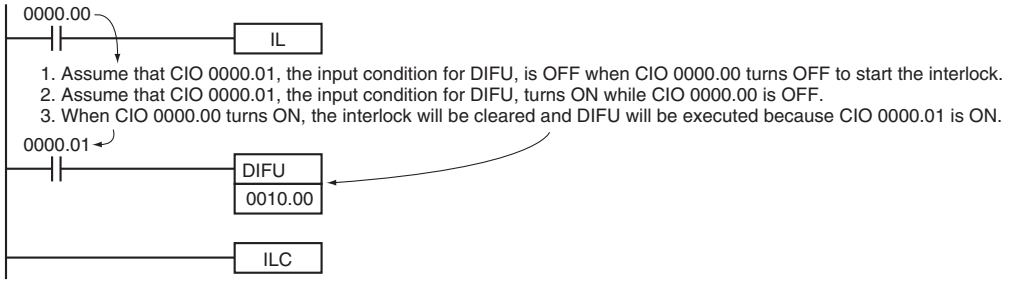
There are no flags affected by this instruction.

Precautions

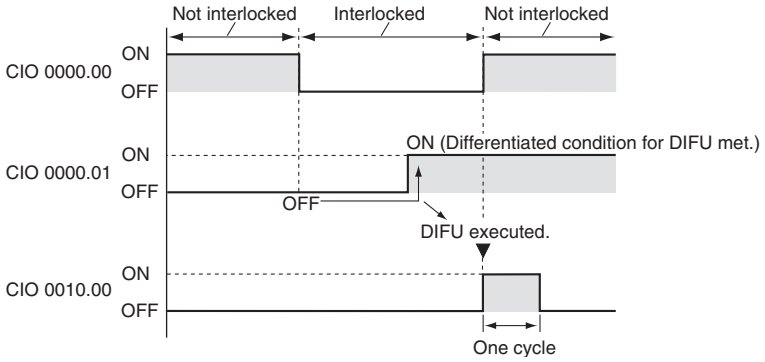
The cycle time is not shortened when a section of the program is interlocked because the interlocked instructions are executed internally.

The operation of DIFU(013), DIFD(014), and differentiated instructions is not dependent solely on the status of the execution condition when they are programmed between IL(002) and ILC(003). Changes in the execution condition for DIFU(013), DIFD(014), or a differentiated instruction are not recorded if the DIFU(013) or DIFD(014) is in an interlocked section and the execution condition for the IL(002) is OFF.

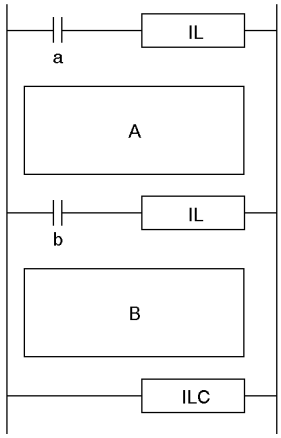
Example: DIFU(013)
If the execution condition for DIFU(013) is OFF when an interlock is started and is ON when the interlock is cleared, DIFU(013) will be executed when the interlock is cleared.



Timing Chart



In general, IL(002) and ILC(003) are used in pairs, although it is possible to use more than one IL(002) with a single ILC(003) as shown in the following diagram. If IL(002) and ILC(003) are not paired, an error message will appear when the program check is performed but the program will be executed properly.



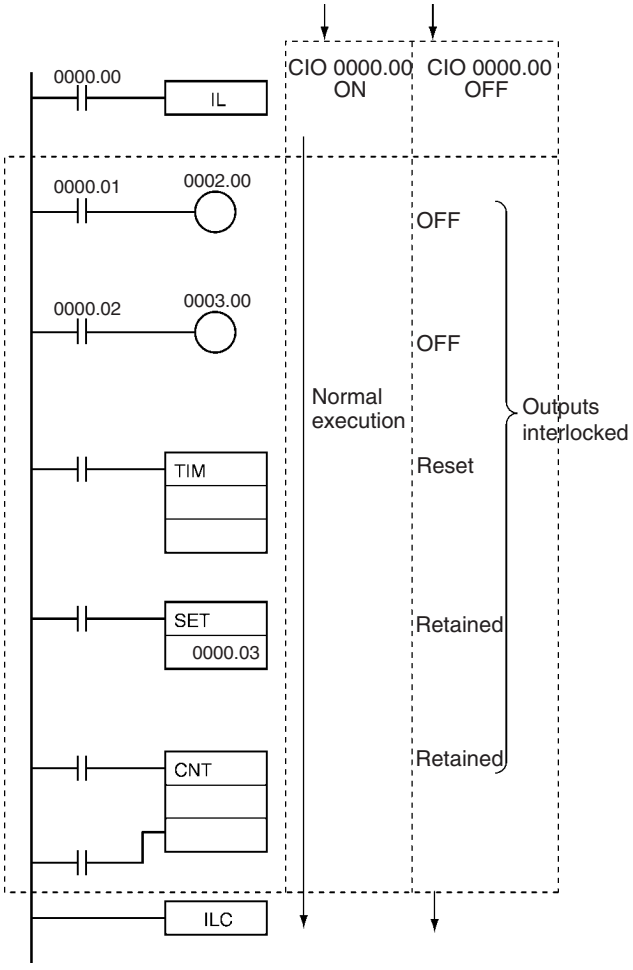
Execution condition		Program section	
a	**b**	**A**	**B**
OFF	ON	Interlocked	Interlocked
OFF	OFF	Interlocked	Interlocked
ON	OFF	Not interlocked	Interlocked
ON	ON	Not interlocked	Not interlocked

IL(002) and ILC(003) cannot be nested.

Examples

When CIO 0000.00 is OFF in the following example, all outputs between IL(002) and ILC(003) are interlocked. When CIO 0000.00 is ON in the follow-

ing example, the instructions between IL(002) and ILC(003) are executed normally.

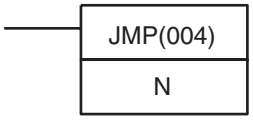


3-4-4 JUMP and JUMP END: JMP(004) and JME(005)

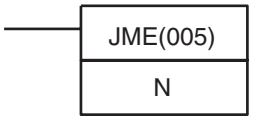
Purpose

When the execution condition for JMP(004) is OFF, program execution jumps directly to the first JME(005) in the program with the same jump number. JMP(004) and JME(005) are used in pairs.

Ladder Symbols



N: Jump number



N: Jump number

Variations

Variations	Jumps when OFF/Does Not Jump when ON	JMP(004)

Variations	Executed Each Cycle for ON Condition	JME(005)

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	Not allowed	OK	OK

Operands

N: Jump Number

The jump number must be 0000 to 00FF (&0 to &255 decimal).

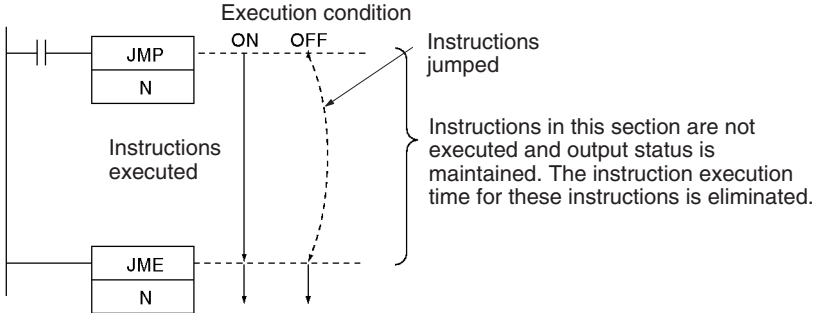
Operand Specifications

Area	N	
	JMP(004)	JME(005)
CIO Area	CIO 0000 to CIO 6143	---
Work Area	W000 to W255	---
Auxiliary Bit Area	A000 to A959	---
Timer Area	T0000 to T0255	---
Counter Area	C0000 to C0255	---
DM Area	D00000 to D32767	---
Indirect DM addresses in binary	@ D00000 to @ D32767	---
Indirect DM addresses in BCD	*D00000 to *D32767	---
Constants	#0000 to #00FF (binary) or &0 to &255	#0000 to #00FF (binary) or &0 to &255
Data Resisters	DR0 to DR15	---
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15	---

Description

When the execution condition for JMP(004) is ON, no jump is made and the program is executed consecutively as written.

When the execution condition for JMP(004) is OFF, program execution jumps directly to the first JME(005) in the program with the same jump number. The instructions between JMP(004) and JME(005) are not executed, so the status of outputs between JMP(004) and JME(005) is maintained. In block programs, the instructions between JMP(004) and JME(005) are skipped regardless of the status of the execution condition.



Because all of instructions between JMP(004) and JME(005) are skipped when the execution condition for JMP(004) is OFF, the cycle time is reduced by the total execution time of the skipped instructions.

Flags (JMP)

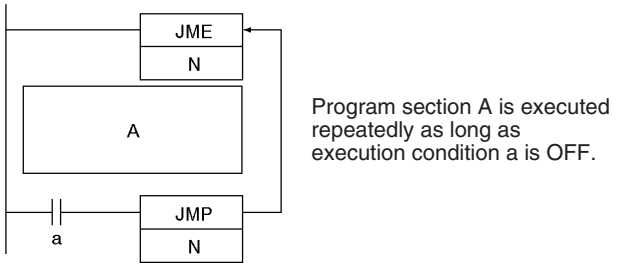
Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0000 to 00FF. ON if there is a JMP(004) in the program without a JME(005) with the same jump number. ON if there is a JMP(004) in the task without a JME(005) with the same jump number in the same task. OFF in all other cases.

Precautions

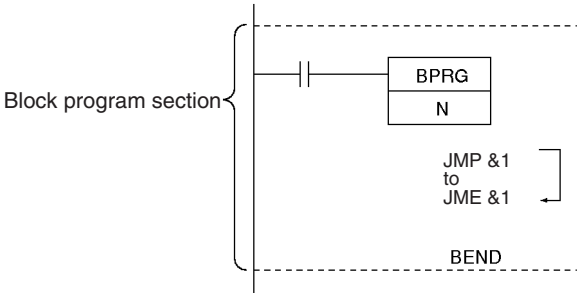
All of the outputs (bits and words) in jumped instructions retain their previous status. Operating timers (TIM, TIMH(015), and TMHH(540)) continue timing because the PVs are updated even when the timer instruction is not being executed.

When there are two or more JME(005) instructions with the same jump number, only the instruction with the lower address will be valid. The JME(005) with the higher program address will be ignored.

When JME(005) precedes JMP(004) in the program, the instructions between JME(005) and JMP(004) will be executed repeatedly as long as the execution condition for JMP(004) is OFF. A Cycle Time Too Long error will occur if the execution condition is not turned ON or END(001) is not executed within the maximum cycle time.



In block programs, the instructions between JMP(004) and JME(005) are always skipped regardless of the status of the execution condition for JMP(004).



JMP(004) and JME(005) pairs must be in the same task because jumps between tasks are not allowed. An error will occur if a JME(005) instruction is not programmed in the same task as its corresponding JMP(004) instruction.

The operation of DIFU(013), DIFD(014), and differentiated instructions is not dependent solely on the status of the execution condition when they are programmed between JMP(004) and JME(005). When DIFU(013), DIFD(014), or a differentiated instruction is executed in an jumped section immediately after the execution condition for the JMP(004) has gone ON, the execution condition for the DIFU(013), DIFD(014), or differentiated instruction will be compared to the execution condition that existed before the jump became effective (i.e., before the execution condition for JMP(004) went OFF).

JMP(004) Specifications

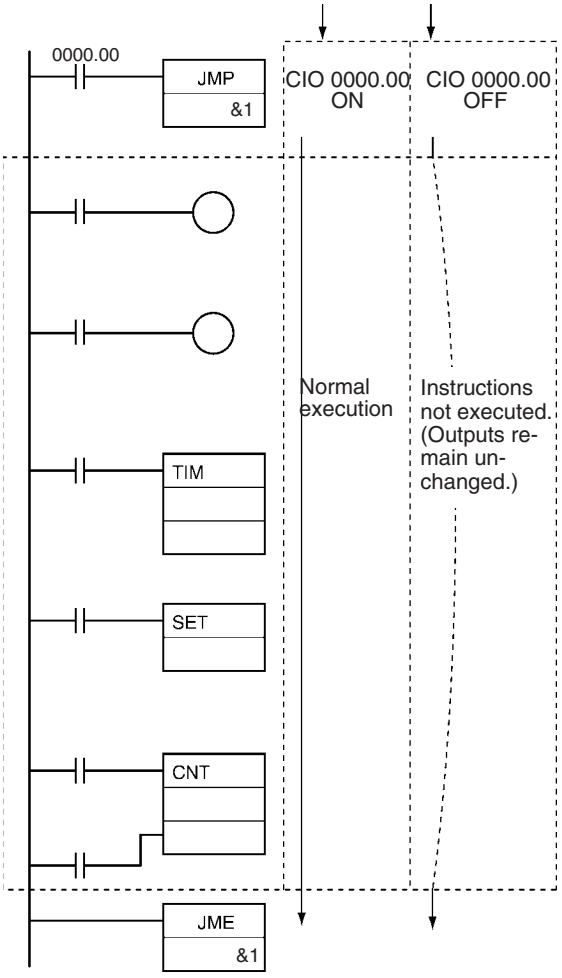
Item	JMP-JME
Execution condition for jumping	OFF
Number of JMP instructions	256 max.
Instruction processing when jumping	Not executed
Execution time when jumping	0
Outputs while jumping	Retains previous status
PV of active timer, while jumping	Continues timing
Processing in block program area	Jumps unconditionally

Examples

Basic Operation

When CIO 0000.00 is OFF in the following example, the instructions between JMP(004) and JME(005) are not executed and the outputs maintain their previous status.

When CIO 0000.00 is ON in the following example, the instructions between JMP(004) and JME(005) are executed normally.



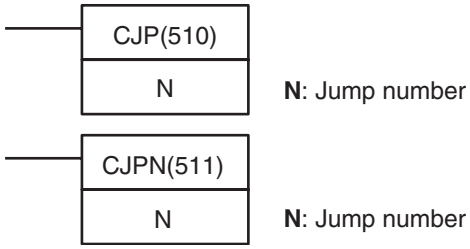
3-4-5 CONDITIONAL JUMP: CJP(510)/CJPN(511)

Purpose

The operation of CJP(510) is the basically the opposite of JMP(004). When the execution condition for CJP(510) is ON, program execution jumps directly to the first JME(005) in the program with the same jump number. CJP(510) and JME(005) are used in pairs.

The operation of CJPN(511) is almost identical to JMP(004). When the execution condition for CJP(004) is OFF, program execution jumps directly to the first JME(005) in the program with the same jump number. CJPN(511) and JME(005) are used in pairs.

Ladder Symbols



Variations

Variations	Jumps when ON/Does Not Jump when OFF	CJP(510)

Variations	Jumps when OFF/Does Not Jump when ON	CJPN(511)

Variations	Executed Each Cycle for ON Condition	JME(005)

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	Not allowed	OK	OK

Operands

N: Jump Number

The jump number must be 0000 to 00FF (0 to 255 decimal).

Operand Specifications

Area	N		
	CJP(510)	CJPN(511)	JME(005)
CIO Area	CIO 0000 to CIO 6143		---
Work Area	W000 to W255		---
Auxiliary Bit Area	A000 to A959		---
Timer Area	T0000 to T0255		---
Counter Area	C0000 to C0255		---
DM Area	D00000 to D32767		---
Indirect DM addresses in binary	@ D00000 to @ D32767		---
Indirect DM addresses in BCD	*D00000 to *D32767		---
Constants	#0000 to #00FF (binary) or &0 to &255		#0000 to #00FF (binary) or &0 to &255
Data Registers	DR0 to DR15		---
Index Registers	---		---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15		---

Description

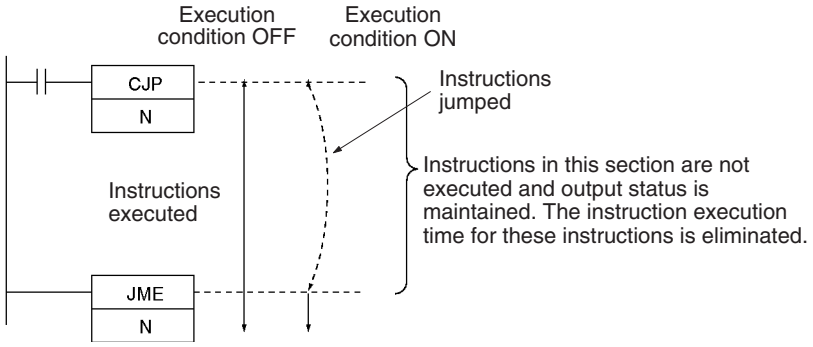
The operation of CJP(510) and CJPN(511) differs only in the execution condition. CJP(510) jumps to the first JME(005) when the execution condition is ON and CJPN(511) jumps to the first JME(005) when the execution condition is OFF.

Because the jumped instructions are not executed, the cycle time is reduced by the total execution time of the jumped instructions.

Operation of CJP(510)

When the execution condition for CJP(510) is OFF, no jump is made and the program is executed consecutively as written.

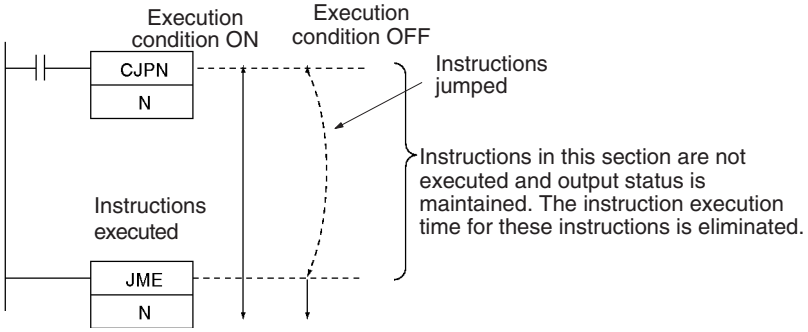
When the execution condition for CJP(510) is ON, program execution jumps directly to the first JME(005) in the program with the same jump number.



Operation of CJPN(511)

When the execution condition for CJPN(511) is ON, no jump is made and the program is executed consecutively as written.

When the execution condition for CJPN(511) is OFF, program execution jumps directly to the first JME(005) in the program with the same jump number.



Flags

The following table shows the flags affected by CJP(510) and CJPN(511).

Name	Label	Operation
Error Flag	ER	ON if there is not a JME(005) with the same jump number as CJP(510) or CJPN(511). (See note.) ON if N is not within the specified range of 0 to 255 decimal (0000 to 00FF hex). ON if there is a CJP(510) or CJPN(511) instruction in a task without a JME(005) with the same jump number. OFF in all other cases.

Note The jump number must be between the range 0 to 255 (0000 to 00FF hex).

Precautions

All of the outputs (bits and words) in jumped instructions retain their previous status. Operating timers (TIM, TIMX(550), TIMH(015), TIMHX(551), TMHH(540), and TMHHX(552)) continue timing be-cause the PVs are updated even when the timer instruction is not being executed.

When there are two or more JME(005) instructions with the same jump number, only the instruction with the lower address will be valid. The JME(005) with the higher program address will be ignored.

When JME(005) precedes the CJP(510) or CJPN(511) instruction in the program, the instructions in-between will be executed repeatedly as long as the

execution condition remains OFF (CJP(510)) or ON (CJPN(511)). A Cycle Time Too Long error will occur if the jump is not completed by changing the execution condition executing END(001) within the maximum cycle time.

The CJP(510) or CJPN(511) instructions will operate normally in block programs.

When the execution condition for the CJP(510) is ON or the execution condition for CJPN(511) is OFF, program execution will jump directly to the JME instruction without executing instructions between CJP(510)/CJPN(511) and JME. No execution time will be required for these instructions and the cycle time will thus be reduced.

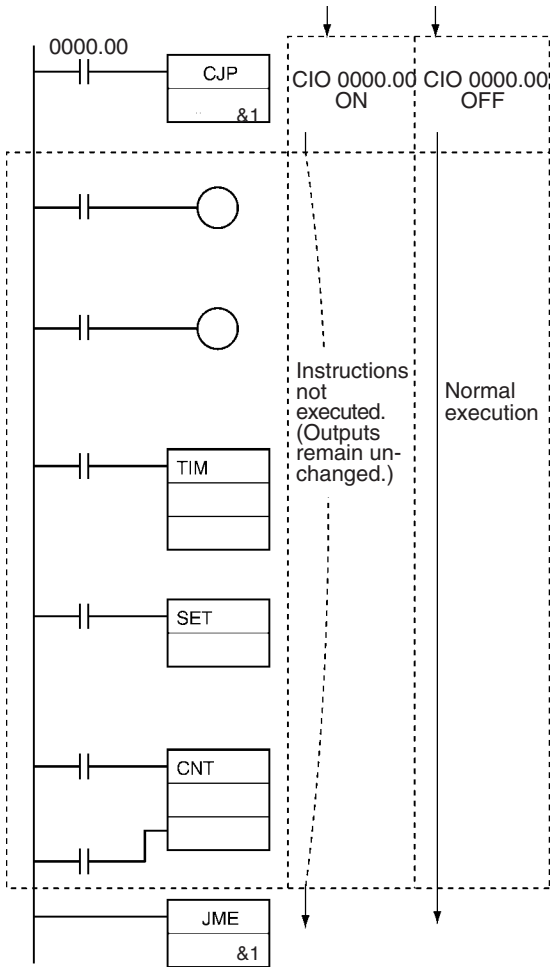
When the execution condition for the JMP0 is OFF, NOP processing is executed between the JMP0 and JME0, requiring execution time. Therefore, the cycle time will not be reduced.

When a CJP(510) or CJPN(511) instruction is programmed in a task, there must be a JME(005) with the same jump number because jumps between tasks are not allowed. An error will occur if a corresponding JME(005) instruction is not programmed in the same task.

The operation of DIFU(013), DIFD(014), and differentiated instructions is not dependent solely on the status of the execution condition when they are programmed in a jumped program section. When DIFU(013), DIFD(014), or a differentiated instruction is executed in an jumped section immediately after the execution condition for the CJP(510) has gone OFF (ON for CJPN(511)), the execution condition for the DIFU(013), DIFD(014), or differentiated instruction will be compared to the execution condition that existed before the jump became effective.

Example

When CIO 0000.00 is ON in the following example, the instructions between CJP(510) and JME(005) are not executed and the outputs maintain their previous status.
When CIO 0000.00 is OFF in the following example, the instructions between CJP(510) and JME(005) are executed normally.



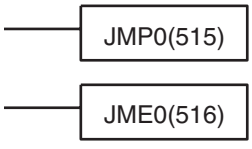
Note For CJPN(511), the ON/OFF status of CIO 000000 would be reversed.

3-4-6 MULTIPLE JUMP and JUMP END: JMP0(515) and JME0(516)

Purpose

When the execution condition for JMP0(515) is OFF, all instructions from JMP0(515) to the next JME0(516) in the program are processed as NOP(000). Use JMP0(515) and JME0(516) in pairs. There is no limit on the number of pairs that can be used in the program.

Ladder Symbols



Variations

Variations	Jumps when OFF/Does Not Jump when ON	JMP0(515)

Variations	Executed Each Cycle for ON Condition	JME0(516)

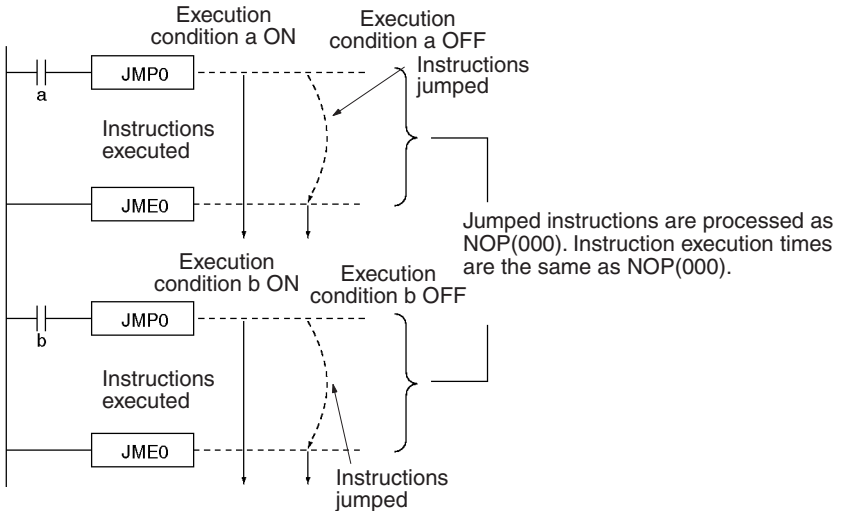
Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	Not allowed	OK	OK

Description

When the execution condition for JMP0(515) is ON, no jump is made and the program executed consecutively as written.

When the execution condition for JMP0(515) is OFF, all instructions from JMP0(515) to the next JME0(516) in the program are processed as NOP(000). Unlike JMP(004), CJP(510), and CJPN(511), JMP0(515) does not use jump numbers, so these instructions can be placed anywhere in the program.



Unlike JMP(004), CJP(510), and CJPN(511) which jump directly to the first JME(005) instruction in the program, all of the instructions between JMP0(515) and JME0(516) are executed as NOP(000). The execution time of the jumped instructions will be reduced, but not eliminated. The jumped instructions themselves are not executed and their outputs (bits and words) maintain their previous status.

Precautions

Multiple pairs of JMP0(515) and JME0(516) instructions can be used in the program, but the pairs cannot be nested.

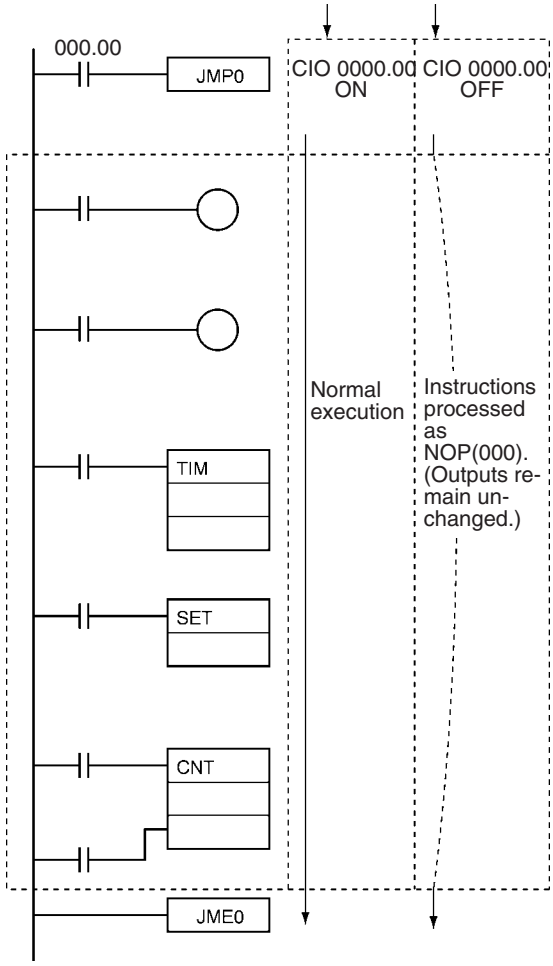
JMP0(515) and JME0(516) cannot be used in block programs.

JMP0(515) and JME0(516) pairs must be in the same tasks because jumps between tasks are not allowed.

The operation of DIFU(013), DIFD(014), and differentiated instructions is not dependent solely on the status of the execution condition when they are programmed between JMP0(515) and JME0(516). When DIFU(013), DIFD(014), or a differentiated instruction is executed in an jumped section immediately after the execution condition for the JMP0(515) has gone ON, the execution condition for the DIFU(013), DIFD(014), or differentiated instruction will be compared to the execution condition that existed before the jump became effective (i.e., before the execution condition for JMP0(515) went OFF).

Example

When CIO 0000.00 is OFF in the following example, the instructions between JMP0(515) and JME0(516) are processed as NOP(000) instructions and the outputs maintain their previous status.
When CIO 0000.00 is ON in the following example, the instructions between JMP0(515) and JME0(516) are executed normally.

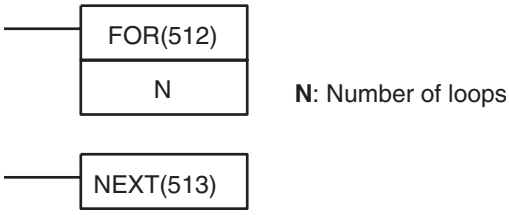


3-4-7 FOR-NEXT LOOPS: FOR(512)/NEXT(513)

Purpose

The instructions between FOR(512) and NEXT(513) are repeated a specified number of times. FOR(512) and NEXT(513) are used in pairs.

Ladder Symbols



N: Number of loops

Variations

Variations		
	Executed Each Cycle for ON Condition	FOR(512)
	Executed Each Cycle for ON Condition	NEXT(513)

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	OK

Operands

N: Number of Loops

The number of loops must be 0000 to FFFF (0 to 65,535 decimal).

Operand Specifications

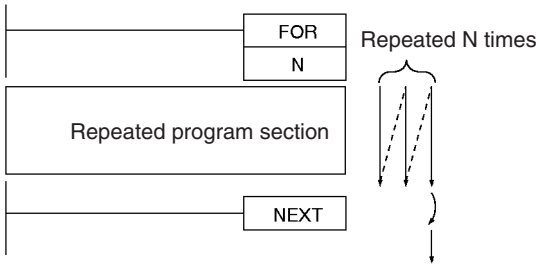
Area	N
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W255
Auxiliary Bit Area	A000 to A959
Timer Area	T0000 to T255
Counter Area	C0000 to C255
DM Area	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	#0000 to #FFFF (binary) or &0 to &65,535
Data Registers	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to , –(– –) IR15

Description

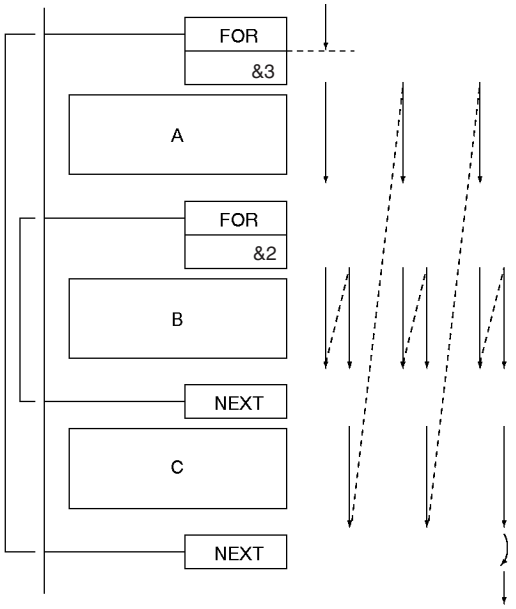
The instructions between FOR(512) and NEXT(513) are executed N times and then program execution continues with the instruction after NEXT(513). The BREAK(514) instruction can be used to cancel the loop.

If N is set to 0, the instructions between FOR(512) and NEXT(513) are processed as NOP(000) instructions.

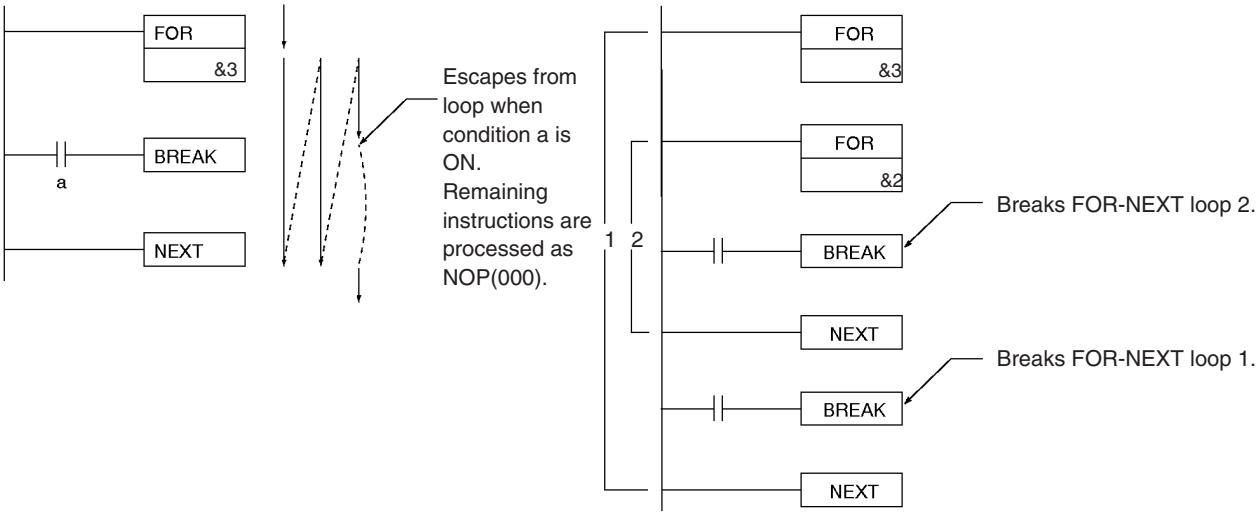
Loops can be used to process tables of data with a minimum amount of programming.



FOR-NEXT loops can be nested up to 15 levels. In the example below, program sections A, B, and C are executed as follows:
A → B → B → C, A → B → B → C, and A → B → B → C



Use BREAK(514) to escape from a FOR-NEXT loop. Several BREAK(514) instructions (the number of levels nested) are required to escape from nested loops. The remaining instructions in the loop after BREAK(514) are processed as NOP(000) instructions.



Alternative Looping Methods

There are two ways to repeat a program section until a given execution condition is input.

1,2,3... 1. FOR-NEXT Loop with BREAK

Start a FOR-NEXT loop with a maximum of N repetitions. Program BREAK(514) within the loop with the desired execution condition. The loop will end before N repetitions if the execution condition is input.

2. JME(005)-JMP(004) Loop

Program a loop with JME(005) before JMP(004). The instructions between JME(005) and JMP(004) will be executed repeatedly as long as the execution condition for JMP(004) is OFF. (A Cycle Time Too Long error will occur if the execution condition is not turned ON or END(001) is not executed within the maximum cycle time.)

Flags

Name	Label	Operation
Error Flag	ER	ON if more than 15 loops are nested. OFF in all other cases.
Equals Flag	=	OFF
Negative Flag	N	OFF

Precautions

Program FOR(512) and NEXT(513) in the same task. Execution will not be repeated if these instructions are not in the same task.

A jump instruction such as JMP(004) may be executed within a FOR-NEXT loop, but do not jump beyond the FOR-NEXT loop.

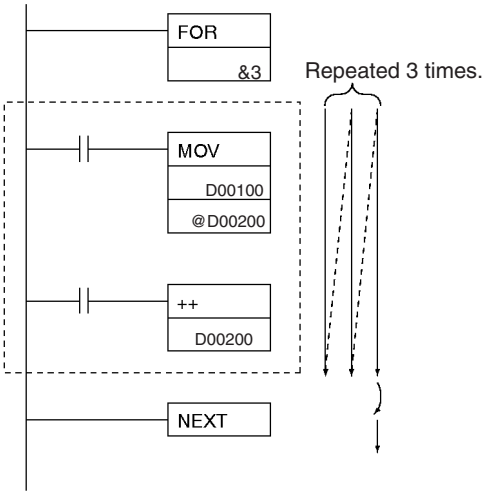
The following instructions cannot be used within FOR-NEXT loops:

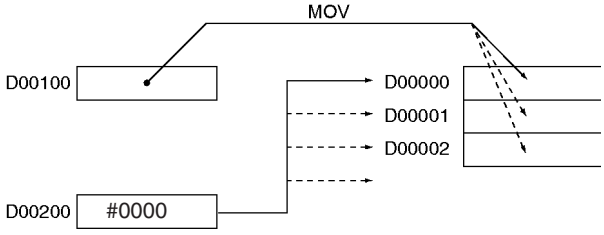
- Block programming instructions
- MULTIPLE JUMP and JUMP END: JMP(515) and JME(516)
- STEP DEFINE and STEP START: STEP(008)/SNXT(009)

Note If a loop repeats in one cycle and a differentiated bit is used in the FOR-NEXT loop, that bit will be always ON or always OFF within that loop.

Example

In the following example, the looped program section transfers the content of D00100 to the address indicated in D00200 and then increments the content of D00200 by 1.





3-4-8 BREAK LOOP: BREAK(514)

Purpose

Programmed in a FOR-NEXT loop to cancel the execution of the loop for a given execution condition. The remaining instructions in the loop are processed as NOP(000) instructions.

Ladder Symbol

Variations

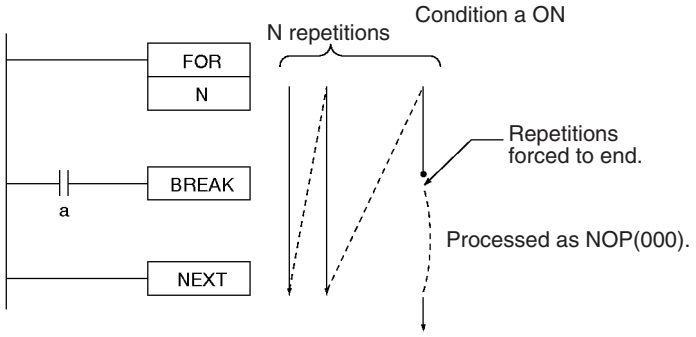
Variations		
	Executed Each Cycle for ON Condition	BREAK(514)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	OK

Description

Program BREAK(514) between FOR(512) and NEXT(513) to cancel the FOR-NEXT loop when BREAK(514) is executed. When BREAK(514) is executed, the rest of the instructions up to NEXT(513) are processed as NOP(000).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	OFF
Negative Flag	N	OFF

Precautions

A BREAK(514) instruction cancels only one loop, so several BREAK(514) instructions (the number of levels nested) are required to escape from nested loops.

BREAK(514) can be used only in a FOR-NEXT loop.

3-5 Timer and Counter Instructions

This section describes instructions used to define and handle timers and counters.

Instruction	Mnemonic	Function code	Page
TIMER	TIM	---	153
HIGH-SPEED TIMER	TIMH	015	156
ONE-MS TIMER	TMHH	540	158
COUNTER	CNT	---	160
REVERSIBLE COUNTER	CNTR	012	163

Refresh Methods for Timer/Counter PV

■ Overview

The timer and counter instructions all use BCD data and all set values for them are input using BCD.

Basic Timer Specifications

The following table shows the basic specifications of the timers.

Item		TIM	TIMH(015)	TMHH(540)
Timing method		Decrementing	Decrementing	Decrementing
Timing units		0.1 s	0.01 s	0.001 s
Max. SV		999.9 s	99.99 s	9.999 s
Outputs/instruction		1	1	1
Timer numbers		Used	Used	Used
Comp. flag refreshing		At execution	At execution	By interrupt every 1 ms
Timer PV refreshing		See note 1.	See note 2.	Every 1 ms
Value after reset	Comp. flags	OFF	OFF	OFF
	PVs	SV	SV	SV

Note

1. TIM PVs are refreshed at execution, at the end of program execution each cycle, or every 80 ms by interrupt if the cycle time exceeds 80 ms.
2. TIMH(015) PVs are refreshed at execution, at the end of program execution each cycle, and every 10 ms by interrupt.

Timer Operation

The following table shows the effects of operating and programming conditions on the operation of the timers.

Item		TIM	TIMH(015)	TMHH(540)
Operating mode change		PV = 0 Completion Flag = OFF		
Power interrupt/reset		PV = 0 Completion Flag = OFF		
Operation in jumped program section (JMP(004)-JME(005))		Operating timers continue timing.		
Operation in interlocked program section (IL(002)-ILC(003))		PV = SV Completion Flag = OFF		
Forced set	Comp. flags	ON		
	PVs	Set to 0.		

Item		TIM	TIMH(015)	TMHH(540)
Forced reset	Comp. flags	OFF		
	PVs	Reset to SV.		

3-5-1 TIMER: TIM

Purpose

TIM operates a decrementing timer with units of 0.1-s. The setting range for the set value (SV) is 0 to 999.9 s. The timer accuracy is –0.01 to 0 s.

Ladder Symbol

Symbol	Operands
TIM N S **N**: Timer number **S**: Set value	N: 0 to 0255 (decimal) S: #0000 to #9999 (BCD)

Variations

Variations		
	Executed Each Cycle for ON Condition	TIM
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	Not allowed

Operands

N: Timer Number

The timer number must be between 0000 and 0255 (decimal).

S: Set Value

The set value must be between #0000 and #9999 (BCD).
(If the set value is set to #0000, the Completion Flag will be turned ON when TIM is executed.)

Operand Specifications

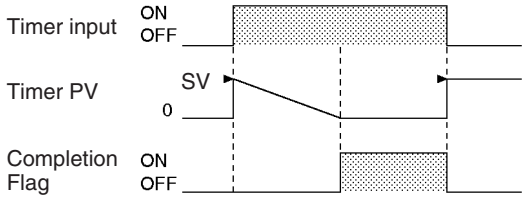
Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W255
Auxiliary Bit Area	---	A000 to A959
Timer Area	0 to 0255 (decimal)	T0000 to T0255
Counter Area	---	C0000 to C0255
DM Area	---	D00000 to D32767
Indirect DM addresses in binary	---	@ D00000 to @ D32767
Indirect DM addresses in BCD	---	*D00000 to *D32767
Constants	---	#0000 to #9999 (BCD) “&” cannot be used.
Data Resisters	---	DR0 to DR15
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15	

Description

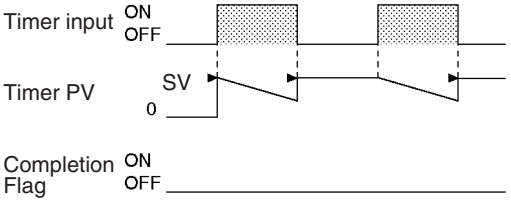
When the timer input is OFF, the timer specified by N is reset, i.e., the timer's PV is reset to the SV and its Completion Flag is turned OFF.

When the timer input goes from OFF to ON, TIM starts decrementing the PV. The PV will continue timing down as long as the timer input remains ON and the timer's Completion Flag will be turned ON when the PV reaches 0000.

The status of the timer's PV and Completion Flag will be maintained after the timer times out. To restart the timer, the timer input must be turned OFF and then ON again or the timer's PV must be changed to a non-zero value (by MOV(021), for example).



The following timing chart shows the behavior of the timer's PV and Completion Flag when the timer input is turned OFF before the timer times out.



Flags

Name	Label	Operation
Error Flag	ER	ON if N is indirectly addressed through an Index Register but the address in the Index Register is not the PV address of a timer. ON if S does not contain BCD data. OFF in all other cases.

Precautions

Timer numbers are shared by the TIM, TIMH(015), and TMHH(540) instructions. Two timers can share the same timer number only if they are not executed at the same time. A duplication error will occur when the program is checked, but the timers will operate normally as long as they are not executed at the same time. Timers which share the same timer number will not operate properly if they are executed simultaneously.

Timers will be reset or paused in the following cases. (When a timer is reset, its PV is reset to the SV and its Completion Flag is turned OFF.)

Condition	PV	Completion Flag
Operating mode changed from RUN or MONITOR mode to PROGRAM mode or vice versa.	0000	OFF
Power supply interrupted and reset	0000	OFF
Operation in interlocked program section (IL(002)–ILC(003))	Reset to SV.	OFF
Operation in jumped program section (JMP(004)–JME(005))	PV continues decrementing.	Retains previous status.

Note The PV will be set to the SV when TIM is executed.

When TIM is in a program section between IL(002) and ILC(003) and the program section is interlocked, the PV will be reset to the SV and the Completion Flag will be turned OFF.

When an operating TIM timer is in a jumped program section (JMP(004) and JME(005)), the timer's PV will continue timing. The jumped TIM instruction will not be executed, but the PV will be refreshed each cycle after all tasks have been executed.

When a TIM timer is forced set, its Completion Flag will be turned ON and its PV will be set to 0000. When a TIM timer is forced reset, its Completion Flag will be turned OFF and its PV will be reset to the SV.

The timer's Completion Flag is refreshed only when TIM is executed, so a delay of up to one cycle may be required for the Completion Flag to be turned ON after the timer times out.

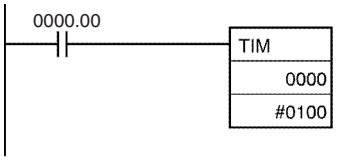
If online editing is used to convert a timer to another kind of timer with the same timer number (such as TIM ↔ TIMH(015) or TIM ↔ TMHH(540)), be sure to reset the Completion Flag. The timer will not operate properly unless the Completion Flag is reset.

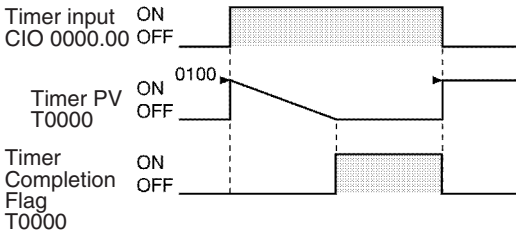
A TIM instruction's PV and Completion Flag are refreshed in the following ways.

Execution of TIM	The PV is updated every time that TIM is executed. The Completion Flag is turned ON if the PV is 0000. The Completion Flag is turned OFF if the PV is not 0000.

Example

When timer input CIO 0000.00 goes from OFF to ON in the following example, the timer PV will begin counting down from the SV. Timer Completion Flag T0000 will be turned ON when the PV reaches 0000.
When CIO 0000.00 goes OFF, the timer PV will be reset to the SV and the Completion Flag will be turned OFF.





3-5-2 HIGH-SPEED TIMER: TIMH(015)

Purpose

TIMH(015) operates a decrementing timer with units of 10-ms. The setting range for the set value (SV) is 0 to 99.99 s for TIMH(015). The timer accuracy is –0.01 to 0 s.

Ladder Symbol

Symbol		Operands
TIMH(015) N S	**N**: Timer number **S**: Set value	N: 0 to 255 (decimal) S: #0000 to #9999 (BCD)

Variations

Variations		
	Executed Each Cycle for ON Condition	TIMH(015)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	Not allowed

Operands

N: Timer Number

The timer number must be between 0000 and 0255 (decimal).

S: Set Value

The set value must be between #0000 and #9999 (BCD).

Operand Specifications

Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W255
Auxiliary Bit Area	---	A000 to A959
Timer Area	0 to 255 (decimal)	T0000 to T0255
Counter Area	---	C0000 to C0255
DM Area	---	D00000 to D32767
Indirect DM addresses in binary	---	@ D00000 to @ D32767
Indirect DM addresses in BCD	---	*D00000 to *D32767
Constants	---	#0000 to #9999 (BCD) “&” cannot be used.
Data Resisters	---	DR0 to DR15
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15	

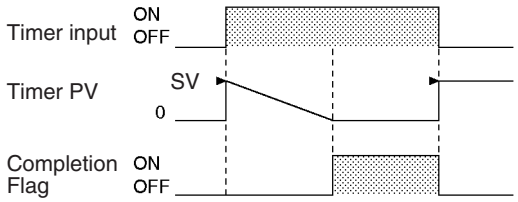
Description

When the timer input is OFF, the timer specified by N is reset, i.e., the timer’s PV is reset to the SV and its Completion Flag is turned OFF.

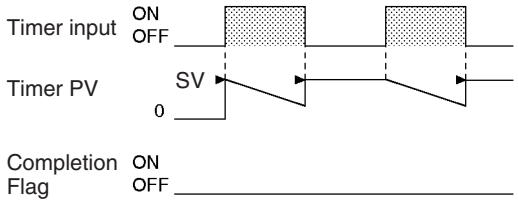
When the timer input goes from OFF to ON, TIMH(015) starts decrementing the PV. The PV will continue timing down as long as the timer input remains

ON and the timer's Completion Flag will be turned ON when the PV reaches 0000.

The status of the timer's PV and Completion Flag will be maintained after the timer times out. To restart the timer, the timer input must be turned OFF and then ON again or the timer's PV must be changed to a non-zero value (by MOV(021), for example).



The following timing chart shows the behavior of the timer's PV and Completion Flag when the timer input is turned OFF before the timer times out.



Flags

Name	Label	Operation
Error Flag	ER	ON if N is indirectly addressed through an Index Register but the address in the Index Register is not the PV address of a timer. ON if S does not contain BCD data. OFF in all other cases.

Precautions

Timer numbers are shared by the TIM, TIMH(015), and TMHH(540) instructions. Two timers can share the same timer number only if they are not executed at the same time. A duplication error will occur when the program is checked, but the timers will operate normally as long as they are not executed at the same time. Timers which share the same timer number will not operate properly if they are executed simultaneously.

The Completion Flags for TIMH(015) timers will be updated when the instruction is executed.

Timers will be reset or paused in the following cases. (When a timer is reset, its PV is reset to the SV and its Completion Flag is turned OFF.)

Condition	PV	Completion Flag
Operating mode changed from RUN or MONITOR mode to PROGRAM mode or vice versa.	0000	OFF
Power supply interrupted and reset	0000	OFF
Operation in interlocked program section (IL(002)–ILC(003))	Reset to SV.	OFF
Operation in jumped program section (JMP(004)–JME(005))	PV continues decrementing.	Retains previous status.

Note The PV will be set to the SV when TIMH(015) is executed.

When an operating TIMH(015) timer is in a jumped program section (JMP(004), and JME(005)), the timer's PV will continue timing. (The jumped TIMH(015) instruction will not be executed, but the PV will be refreshed every 10 ms and each cycle after all tasks have been executed.)

When TIMH(015)) is in a program section between IL(002) and ILC(003) and the program section is interlocked, the PV will be reset to the SV and the Completion Flag will be turned OFF.

When a TIMH(015) timer is forced set, its Completion Flag will be turned ON and its PV will be set to 0000. When a TIMH(015) timer is forced reset, its Completion Flag will be turned OFF and its PV will be reset to the SV.

The timer's Completion Flag is refreshed only when TIMH(015) is executed, so a delay of up to one cycle may be required for the Completion Flag to be turned ON after the timer times out.

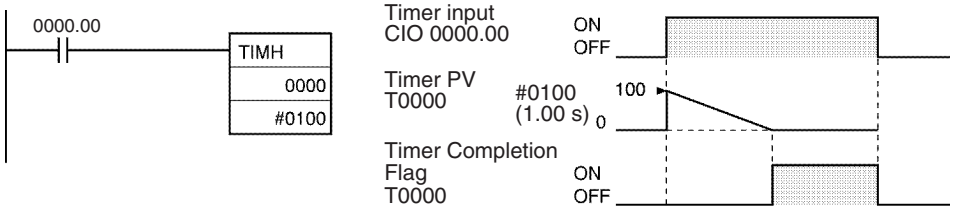
If online editing is used to convert a timer to another kind of timer with the same timer number (such as TIMH(015) ↔ TIM or TIMH(015) ↔ TMHH(540)), be sure to reset the Completion Flag. The timer will not operate properly unless the Completion Flag is reset.

A TIMH(015) instruction's PV and Completion Flag are refreshed in the following ways.

Execution of TIMH(015)	The Completion Flag is turned ON if the PV is 0000. The Completion Flag is turned OFF if the PV is not 0000.
10-ms interval refreshing	The timer's PV is updated every 10 ms.

Example

When timer input CIO 0000.00 goes from OFF to ON in the following example, the timer PV will begin counting down from the SV. The Timer Completion Flag, T0000, will be turned ON when the PV reaches 0000.
When CIO 0000.00 goes OFF, the timer PV will be reset to the SV and the Completion Flag will be turned OFF.



3-5-3 ONE-MS TIMER: TMHH(540)

Purpose

TMHH(540) operates a decrementing timer with units of 1-ms. The setting range for the set value (SV) is 0 to 9.999 s for TMHH(540). The timer accuracy is –0.001 to 0 s.

Ladder Symbol

Symbol	Operands
TMHH(540) N S **N**: Timer number **S**: Set value	N: 0 to 15 (decimal) S: #0000 to #9999 (BCD)

Variations

Variations		
	Executed Each Cycle for ON Condition	TMHH(540)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	Not allowed

Operands

N: Timer Number

The timer number must be between 0 and 15 (decimal).

S: Set Value

The set value must be between #0000 and #9999 (BCD).

Operand Specifications

Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W255
Auxiliary Bit Area	---	A000 to A959
Timer Area	0 to 15 (decimal)	T0000 to T0255
Counter Area	---	C0000 to C0255
DM Area	---	D00000 to D32767
Indirect DM addresses in binary	---	@ D00000 to @ D32767
Indirect DM addresses in BCD	---	*D00000 to *D32767
Constants	---	#0000 to #9999 (BCD) "&" cannot be used.
Data Resisters	---	DR0 to DR15
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15	

Description

When the timer input is OFF, the timer specified by N is reset, i.e., the timer's PV is reset to the SV and its Completion Flag is turned OFF.

When the timer input goes from OFF to ON, TMHH(540) starts decrementing the PV. The PV will continue timing down as long as the timer input remains ON and the timer's Completion Flag will be turned ON when the PV reaches 0000.

The status of the timer's PV and Completion Flag will be maintained after the timer times out. To restart the timer, the timer input must be turned OFF and then ON again or the timer's PV must be changed to a non-zero value (by MOV(021), for example).

Flags

Name	Label	Operation
Error Flag	ER	ON if N is indirectly addressed through an Index Register but the address in the Index Register is not the PV address of a timer. ON if S does not contain BCD data. OFF in all other cases.

Precautions

Timer numbers are shared by the TIM, TIMH(015), and TMHH(540) instructions. Two timers can share the same timer number only if they are not executed at the same time. A duplication error will occur when the program is checked, but the timers will operate normally as long as they are not executed at the same time. Timers which share the same timer number will not operate properly if they are executed simultaneously.

The Completion Flag is updated only when TMHH(540) is executed. The Completion Flag can thus be delayed by up to one cycle time from the actual set value.

Timers will be reset or paused in the following cases. (When a timer is reset, its PV is reset to the SV and its Completion Flag is turned OFF.)

Condition	PV	Completion Flag
Operating mode changed from RUN or MONITOR mode to PROGRAM mode or vice versa.	0000	OFF
Power supply interrupted and reset	0000	OFF
Operation in interlocked program section (IL(002)–ILC(003))	Reset to SV.	OFF
Operation in jumped program section (JMP(004)–JME(005))	PV continues decrement-ing.	Retains previous status.

Note The PV will be set to the SV when TMHH(540) is executed.

When an operating TMHH(540) timer is in a jumped program section (JMP(004), JME(005)), the timer's PV will continue timing. (The jumped TMHH(540) instruction will not be executed, but the PV will be refreshed every 1 ms.)

When TMHH(540) is in a program section between IL(002) and ILC(003) and the program section is interlocked, the PV will be reset to the SV and the Completion Flag will be turned OFF.

When a TMHH(540) timer is forced set, its Completion Flag will be turned ON and its PV will be set to 0000. When a TMHH(540) timer is forced reset, its Completion Flag will be turned OFF and its PV will be reset to the SV.

If online editing is used to convert a timer to another kind of timer with the same timer number (such as TMHH(540) ↔ TIM or TMHH(540) ↔ TIMH(015)), be sure to reset the Completion Flag. The timer will not operate properly unless the Completion Flag is reset.

A TMHH(540) instruction's PV and Completion Flag are refreshed as shown in the following table.

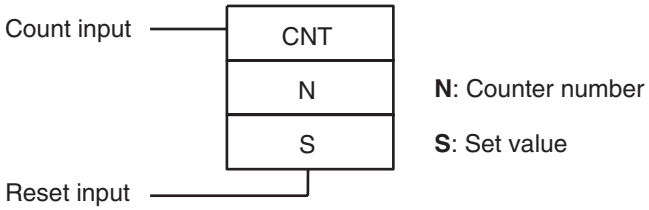
Execution of TMHH(540)	The Completion Flag is turned ON if the PV is 0000. The Completion Flag is turned OFF if the PV is not 0000.
1-ms interval refreshing	The timer's PV is updated every 1 ms.

3-5-4 COUNTER: CNT

Purpose

CNT operates a decrementing counter. The setting range 0 to 9,999 for CNT.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	CNT
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	OK

Operands

N: Counter Number
The counter number must be between 0 and 255 (decimal).

S: Set Value
The set value must be between #0000 and #9999 (BCD).

Operand Specifications

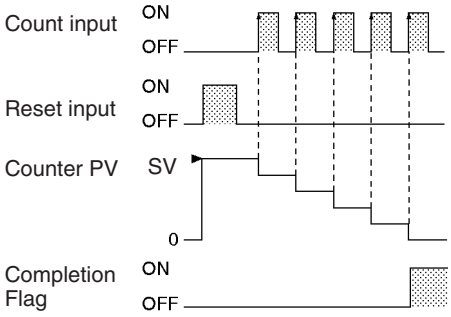
Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W255
Auxiliary Bit Area	---	A000 to A959
Timer Area	---	T0000 to T0255
Counter Area	0 to 255 (decimal)	C0000 to C0255
DM Area	---	D00000 to D32767
Indirect DM addresses in binary	---	@ D00000 to @ D32767
Indirect DM addresses in BCD	---	*D00000 to *D32767
Constants	---	#0000 to #9999 (BCD) “&” cannot be used.
Data Resisters	---	DR0 to DR15
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15	

Description

The counter PV is decremented by 1 every time that the count input goes from OFF to ON. The Completion Flag is turned ON when the PV reaches 0.

Once the Completion Flag is turned ON, reset the counter by turning the reset input ON. Otherwise, the counter cannot be restarted.

The counter is reset and the count input is ignored when the reset input is ON. (When a counter is reset, its PV is reset to the SV and the Completion Flag is turned OFF.)



Flags

Name	Label	Operation
Error Flag	ER	ON if N is indirectly addressed through an Index Register but the address in the Index Register is not the PV address of a counter. ON if S does not contain BCD data. OFF in all other cases.

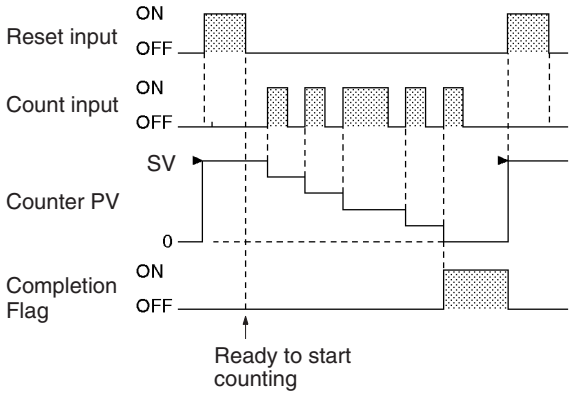
Precautions

Counter numbers are shared by the CNT, and CNTR(012) instructions. Two counters can share the same timer number only if they are not executed at the same time. A duplication error will occur when the program is checked, but the counters will operate normally as long as they are not executed at the same time. Counters which share the same counter number will not operate properly if they are executed simultaneously.

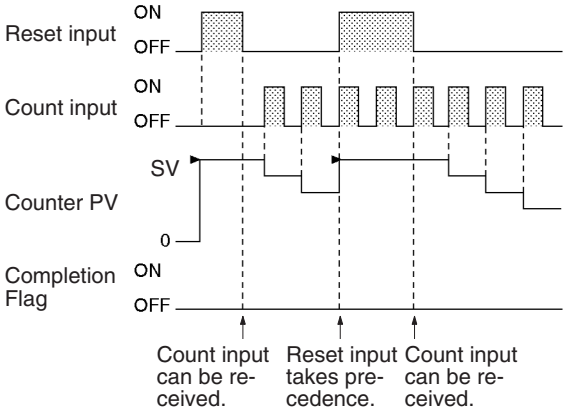
A counter's PV is refreshed when the count input goes from OFF to ON and the Completion Flag is refreshed each time that CNT is executed. The Completion Flag is turned ON if the PV is 0 and it is turned OFF if the PV is not 0.

When a CNT counter is forced set, its Completion Flag will be turned ON and its PV will be reset to 0000. When a CNT counter is forced reset, its Completion Flag will be turned OFF and its PV will be set to the SV.

Be sure to reset the counter by turning the reset input from OFF → ON → OFF before beginning counting with the count input, as shown in the following diagram. The count input will not be received if the reset input is ON.



The reset input will take precedence and the counter will be reset if the reset input and count input are both ON at the same time. (The PV will be reset to the SV and the Completion Flag will be turned OFF.)



Note If online editing is used to add a counter, the counter must be reset before it will work properly. If the counter is not reset, the previous value will be used as the counter's present value (PV), and the counter may not operate properly after it is written.

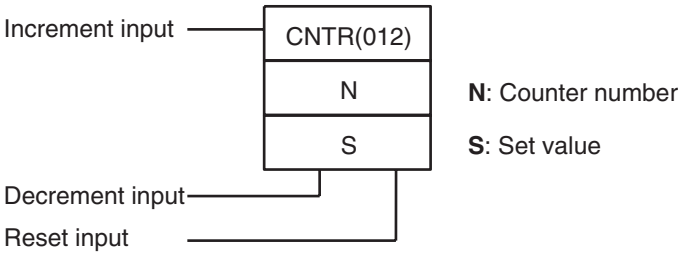
Counter PVs are not retained through a power interruption.

3-5-5 REVERSIBLE COUNTER: CNTR(012)

Purpose

CNTR(012) operates a reversible counter.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	CNTR(012)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	OK

Operands

N: Counter Number

The counter number must be between 0 and 255 (decimal).

S: Set Value

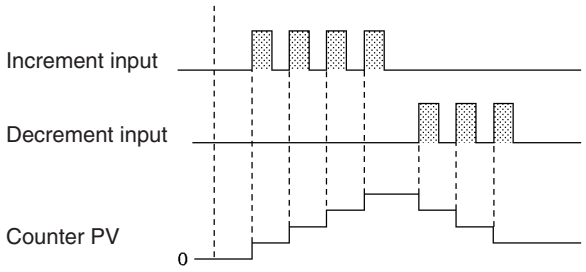
The set value must be between #0000 and #9999 (BCD).

Operand Specifications

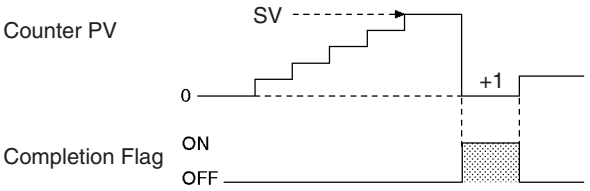
Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W255
Auxiliary Bit Area	---	A000 to A959
Timer Area	---	T0000 to T0255
Counter Area	0 to 255 (decimal)	C0000 to C0255
DM Area	---	D00000 to D32767
Indirect DM addresses in binary	---	@ D00000 to @ D32767
Indirect DM addresses in BCD	---	*D00000 to *D32767
Constants	---	#0000 to #9999 (BCD) “&” cannot be used.
Data Resisters	---	DR0 to DR15
Index Registers	---	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15	

Description

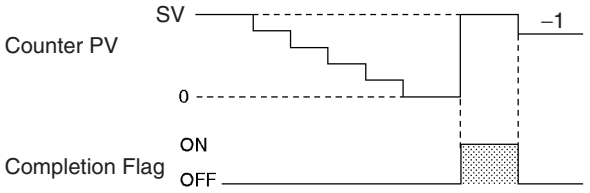
The counter PV is incremented by 1 every time that the increment input goes from OFF to ON and it is decremented by 1 every time that the decrement input goes from OFF to ON. The PV can fluctuate between 0 and the SV.



When incrementing, the Completion Flag will be turned ON when the PV is incremented from the SV back to 0 and it will be turned OFF again when the PV is incremented from 0 to 1.



When decrementing, the Completion Flag will be turned ON when the PV is decremented from 0 up to the SV and it will be turned OFF again when the PV is decremented from the SV to SV–1.



Flags

Name	Label	Operation
Error Flag	ER	ON if N is indirectly addressed through an Index Register but the address in the Index Register is not the PV address of a counter. ON in BCD mode and S does not contain BCD data. OFF in all other cases.

Precautions

Counter numbers are shared by the CNT and CNTR(012) instructions.Two counters can share the same timer number only if they are not executed at the same time. A duplication error will occur when the program is checked, but the counters will operate normally as long as they are not executed at the same time. Counters which share the same counter number will not operate properly if they are executed simultaneously.

The PV will not be changed if the increment and decrement inputs both go from OFF to ON at the same time. When the reset input is ON, the PV will be reset to 0 and both count inputs will be ignored.

The Completion Flag will be ON only when the PV has been incremented from the SV to 0 or decremented from 0 to the SV; it will be OFF in all other cases.

When inputting the CNTR(012) instruction with mnemonics, first enter the increment input (II), then the decrement input (DI), the reset input (R), and finally the CNTR(012) instruction. When entering with the ladder diagrams,

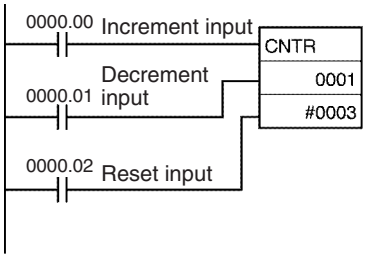
first input the increment input (II), then the CNTR(012) instruction, the decrement input (DI), and finally the reset input (R).

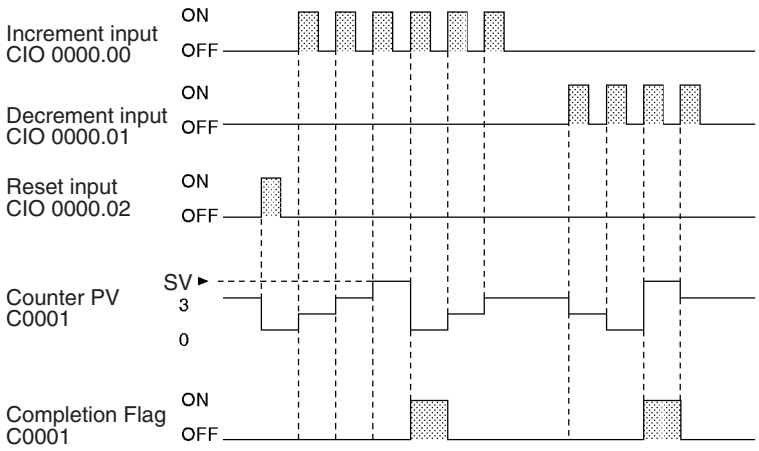
Examples

Basic Operation of CNTR(012)

The counter PV is reset to 0 by turning the reset input (CIO 0000.02) ON and OFF. The PV is incremented by 1 each time that the increment input (CIO 0000.00) goes from OFF to ON. When the PV is incremented from the SV (3), it is automatically reset to 0 and the Completion Flag is turned ON.

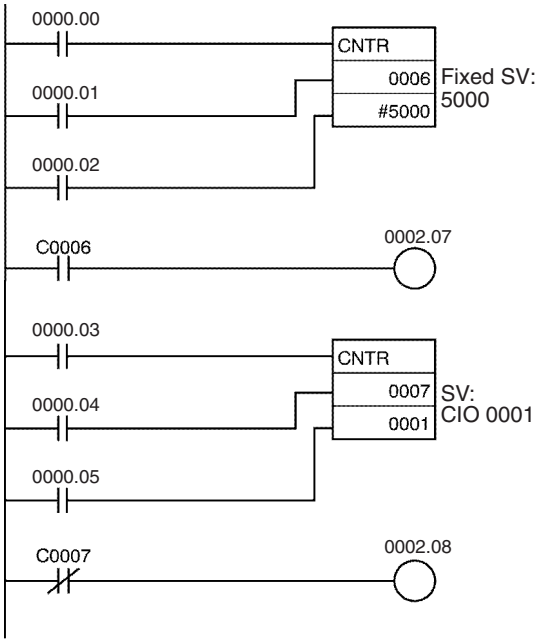
Likewise, the PV is decremented by 1 each time that the decrement input (CIO 0000.01) goes from OFF to ON. When the PV is decremented from 0, it is automatically set to the SV (3) and the Completion Flag is turned ON.

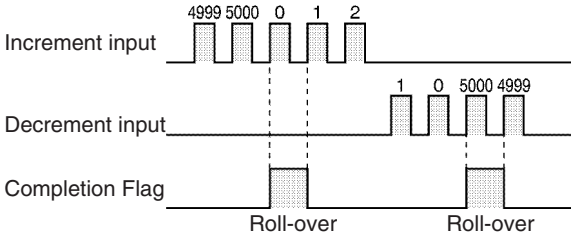




Specifying the SV in a Word

In the following example, the SV for CNTR(012) 0007 is determined by the content of CIO 0001. When the content of CIO 0001 is controlled by an external switch, the set value can be changed manually from the switch.





Coding

Address	Instruction	Operand
000000	LD	0000.00
000001	LD	0000.01
000002	LD	0000.02
000003	CNTR(012)	6
		#5000
000004	LD	C0006
000005	OUT	0002.07
000006	LD	0000.03
000007	LD	0000.04
000008	LD	0000.05
000009	CNTR(012)	7
		0001
000010	LD NOT	C0007
000011	OUT	0002.08

3-6 Comparison Instructions

This section describes instructions used to compare data of various lengths and in various ways.

Instruction	Mnemonic	Function code	Page
Input Comparison Instructions	=, <>, <, <=, >, >= (S, L) (LD, AND, OR)	300 to 328	167
COMPARE	CMP	020	172
DOUBLE COMPARE	CMPL	060	175
SIGNED BINARY COMPARE	CPS	114	177
DOUBLE SIGNED BINARY COMPARE	CPSL	115	180
MULTIPLE COMPARE	MCMP	019	182
TABLE COMPARE	TCMP	085	185
BLOCK COMPARE	BCMP	068	187
EXPANDED BLOCK COMPARE	BCMP2	502	190
AREA RANGE COMPARE	ZCP	088	193
DOUBLE AREA RANGE COMPARE	ZCPL	116	196

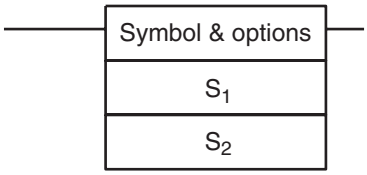
3-6-1 Input Comparison Instructions (300 to 328)

Purpose

Input comparison instructions compare two values (constants and/or the contents of specified words) and create an ON execution condition when the comparison condition is true. Input comparison instructions are available to compare signed or unsigned data of one-word or double length data.

Note Refer to *3-14-21 Single-precision Floating-point Comparison Instructions* for details on single-precision floating-point input comparison instructions.

Ladder Symbol



S1: Comparison data 1

S2: Comparison data 2

Variations

Variations	Creates ON Each Cycle Comparison is True	Input comparison instruction

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications for Instructions for One-word Data

Area	S_1	S_2
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	
Auxiliary Bit Area	A000 to A959	
Timer Area	T0000 to T0255	
Counter Area	C0000 to C0255	
DM Area	D00000 to D32767	

Area	S_1	S_2
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 to #FFFF (binary)	
Data Resisters	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Operand Specifications for Instructions for Double-length Data

Area	S_1	S_2
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	
Data Resisters	---	
Index Registers	IR0 to IR1 (for unsigned data only)	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

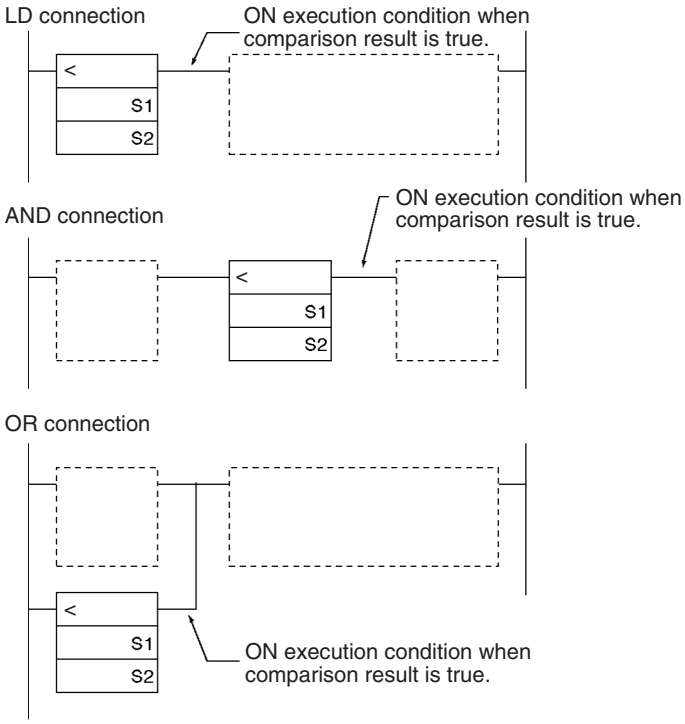
Description

The input comparison instruction compares S_1 and S_2 as signed or unsigned values and creates an ON execution condition when the comparison condition is true. Unlike instructions such as CMP(020) and CMPL(060), the result of an input comparison instruction is reflected directly as an execution condition, so it is not necessary to access the result of the comparison through an Arithmetic Flag and the program is simpler and faster.

Inputting the Instructions

The input comparison instructions are treated just like the LD, AND, and OR instructions to control the execution of subsequent instructions.

Input type	Operation
LD	The instruction can be connected directly to the left bus bar.
AND	The instruction cannot be connected directly to the left bus bar.
OR	The instruction can be connected directly to the left bus bar.



Options

The input comparison instructions can compare signed or unsigned data and they can compare one-word or double values. If no options are specified, the comparison will be for one-word unsigned data. With the three input types for each of 6 symbols and two options, there are 72 different input comparison instructions.

Symbol		Option (data format)	Option (data length)
=	(Equal)	None: Unsigned data	None: One-word data
< >	(Not equal)	S: Signed data	L: Double-length data
<	(Less than)		
<=	(Less than or equal)		
>	(Greater than)		
>=	(Greater than or equal)		

Unsigned input comparison instructions (i.e., instructions without the S option) can handle unsigned binary or BCD data. Signed input comparison instructions (i.e., instructions with the S option) handle signed binary data.

Summary of Input Comparison Instructions

The following table shows the function codes, mnemonics, names, and functions of the 72 input comparison instructions. (For one-word comparisons C1=S_1 and C2=S_2; for double comparisons C1=S_1+1, S_1 and C2=S_2+1, S_2.)

Code	Mnemonic	Name	Function
300	LD=	LOAD EQUAL	True if C1 = C2
	AND=	AND EQUAL	
	OR=	OR EQUAL	
301	LD=L	LOAD DOUBLE EQUAL	
	AND=L	AND DOUBLE EQUAL	
	OR=L	OR DOUBLE EQUAL	
302	LD=S	LOAD SIGNED EQUAL	
	AND=S	AND SIGNED EQUAL	
	OR=S	OR SIGNED EQUAL	
303	LD=SL	LOAD DOUBLE SIGNED EQUAL	
	AND=SL	AND DOUBLE SIGNED EQUAL	
	OR=SL	OR DOUBLE SIGNED EQUAL	
305	LD<>	LOAD NOT EQUAL	True if C1 ≠ C2
	AND<>	AND NOT EQUAL	
	OR<>	OR NOT EQUAL	
306	LD<>L	LOAD DOUBLE NOT EQUAL	
	AND<>L	AND DOUBLE NOT EQUAL	
	OR<>L	OR DOUBLE NOT EQUAL	
307	LD<>S	LOAD SIGNED NOT EQUAL	
	AND<>S	AND SIGNED NOT EQUAL	
	OR<>S	OR SIGNED NOT EQUAL	
308	LD<>SL	LOAD DOUBLE SIGNED NOT EQUAL	
	AND<>SL	AND DOUBLE SIGNED NOT EQUAL	
	OR<>SL	OR DOUBLE SIGNED NOT EQUAL	
310	LD<	LOAD LESS THAN	True if C1 < C2
	AND<	AND LESS THAN	
	OR<	OR LESS THAN	
311	LD<L	LOAD DOUBLE LESS THAN	
	AND<L	AND DOUBLE LESS THAN	
	OR<L	OR DOUBLE LESS THAN	
312	LD<S	LOAD SIGNED LESS THAN	
	AND<S	AND SIGNED LESS THAN	
	OR<S	OR SIGNED LESS THAN	
313	LD<SL	LOAD DOUBLE SIGNED LESS THAN	
	AND<SL	AND DOUBLE SIGNED LESS THAN	
	OR<SL	OR DOUBLE SIGNED LESS THAN	
315	LD<=	LOAD LESS THAN OR EQUAL	True if C1 ≤ C2
	AND<=	AND LESS THAN OR EQUAL	
	OR<=	OR LESS THAN OR EQUAL	
316	LD<=L	LOAD DOUBLE LESS THAN OR EQUAL	
	AND<=L	AND DOUBLE LESS THAN OR EQUAL	
	OR<=L	OR DOUBLE LESS THAN OR EQUAL	
317	LD<=S	LOAD SIGNED LESS THAN OR EQUAL	
	AND<=S	AND SIGNED LESS THAN OR EQUAL	
	OR<=S	OR SIGNED LESS THAN OR EQUAL	

Code	Mnemonic	Name	Function
318	LD<=SL	LOAD DOUBLE SIGNED LESS THAN OR EQUAL	True if C1 ≤ C2
	AND<=SL	AND DOUBLE SIGNED LESS THAN OR EQUAL	
	OR<=SL	OR DOUBLE SIGNED LESS THAN OR EQUAL	
320	LD>	LOAD GREATER THAN	True if C1 > C2
	AND>	AND GREATER THAN	
	OR>	OR GREATER THAN	
321	LD>L	LOAD DOUBLE GREATER THAN	
	AND>L	AND DOUBLE GREATER THAN	
	OR>L	OR DOUBLE GREATER THAN	
322	LD>S	LOAD SIGNED GREATER THAN	
	AND>S	AND SIGNED GREATER THAN	
	OR>S	OR SIGNED GREATER THAN	
323	LD>SL	LOAD DOUBLE SIGNED GREATER THAN	
	AND>SL	AND DOUBLE SIGNED GREATER THAN	
	OR>SL	OR DOUBLE SIGNED GREATER THAN	
325	LD>=	LOAD GREATER THAN OR EQUAL	True if C1 ≥ C2
	AND>=	AND GREATER THAN OR EQUAL	
	OR>=	OR GREATER THAN OR EQUAL	
326	LD>=L	LOAD DOUBLE GREATER THAN OR EQUAL	
	AND>=L	AND DOUBLE GREATER THAN OR EQUAL	
	OR>=L	OR DOUBLE GREATER THAN OR EQUAL	
327	LD>=S	LOAD SIGNED GREATER THAN OR EQUAL	
	AND>=S	AND SIGNED GREATER THAN OR EQUAL	
	OR>=S	OR SIGNED GREATER THAN OR EQUAL	
328	LD>=SL	LOAD DBL SIGNED GREATER THAN OR EQUAL	
	AND>=SL	AND DBL SIGNED GREATER THAN OR EQUAL	
	OR>=SL	OR DBL SIGNED GREATER THAN OR EQUAL	

Flags

Name	Label	Operation
Greater Than Flag	>	ON if $S_1 > S_2$ with one-word data. ON if $S_1+1, S_1 > S_2+1, S_2$ with double-length data. OFF in all other cases.
Greater Than or Equal Flag	> =	ON if $S_1 \geq S_2$ with one-word data. ON if $S_1+1, S_1 \geq S_2+1, S_2$ with double-length data. OFF in all other cases.
Equal Flag	=	ON if $S_1 = S_2$ with one-word data. ON if $S_1+1, S_1 = S_2+1, S_2$ with double-length data. OFF in all other cases.
Not Equal Flag	<>	ON if $S_1 \neq S_2$ with one-word data. ON if $S_1+1, S_1 \neq S_2+1, S_2$ with double-length data. OFF in all other cases.
Less Than Flag	<	ON if $S_1 < S_2$ with one-word data. ON if $S_1+1, S_1 < S_2+1, S_2$ with double-length data. OFF in all other cases.
Less Than or Equal Flag	< =	ON if $S_1 \leq S_2$ with one-word data. ON if $S_1+1, S_1 \leq S_2+1, S_2$ with double-length data. OFF in all other cases.

Precautions

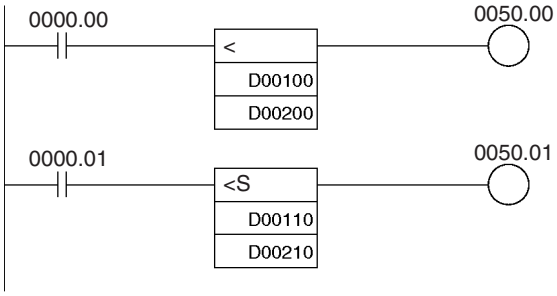
Input comparison instructions cannot be used as right-hand instructions, i.e., another instruction must be used between them and the right bus bar.

Make sure to program an output instruction or special instruction other than intermediate instructions after input comparison instructions as a right-hand instruction.

Examples

AND LESS THAN: AND<(310)

When CIO 0000.00 is ON in the following example, the contents of D00100 and D00200 are compared as unsigned binary data. If the content of D00100 is less than that of D00200, CIO 0050.00 is turned ON and execution proceeds to the next line. If the content of D00100 is not less than that of D00200, the remainder of the instruction line is skipped and execution moves to the next instruction line.



Unsigned LESS THAN Comparison

S_1: D00100 — 8714 — Decimal: 34,580

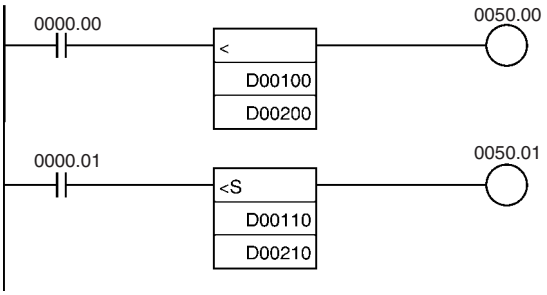
S_2: D00200 — 3A1C — Decimal: 14,876

34,580 > 14,876

(CIO 0050.00 will not be turned ON.)

AND SIGNED LESS THAN: AND<S(312)

When CIO 0000.01 is ON in the following example, the contents of D00110 and D00210 are compared as signed binary data. If the content of D00110 is less than that of D00210, CIO 0050.01 is turned ON and execution proceeds to the next line. If the content of D00110 is not less than that of D00210, the remainder of the instruction line is skipped and execution moves to the next instruction line.



Signed LESS THAN Comparison

S_1: D00110 — 8714 — Decimal: –30,956

S_2: D00210 — 3A1C — Decimal: 14,876

–30,956 < 14,876

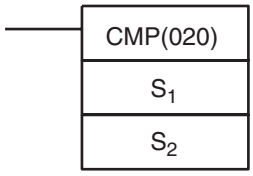
(CIO 0050.01 will be turned ON.)

3-6-2 COMPARE: CMP(020)

Purpose

Compares two unsigned binary values (constants and/or the contents of specified words) and outputs the result to the Arithmetic Flags in the Auxiliary Area.

Ladder Symbol



S1: Comparison data 1

S2: Comparison data 2

Variations

Variations		
	Executed Each Cycle for ON Condition	CMP(020)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

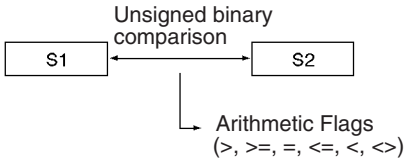
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S_1	S_2
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	
Auxiliary Bit Area	A000 to A959	
Timer Area	T0000 to T0255	
Counter Area	C0000 to C0255	
DM Area	D00000 to D32767	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 to #FFFF (binary)	
Data Resisters	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

CMP(020) compares the unsigned binary data in S_1 and S_2 and outputs the result to Arithmetic Flags (the Greater Than, Greater Than or Equal, Equal, Less Than or Equal, Less Than, and Not Equal Flags) in the Auxiliary Area.



Condition Flag Status

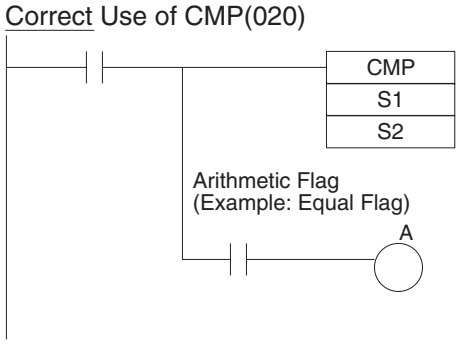
The following table shows the status of the Arithmetic Flags after execution of CMP(020). (A status of "---" indicates that the Flag may be ON or OFF.)

CMP(020) Result	Flag status					
	>	> =	=	< =	<	< >
$S_1 > S_2$	ON	ON	OFF	OFF	OFF	ON
$S_1 = S_2$	OFF	ON	ON	ON	OFF	OFF
$S_1 < S_2$	OFF	OFF	OFF	ON	ON	ON

Using CMP(020) Results in the Program

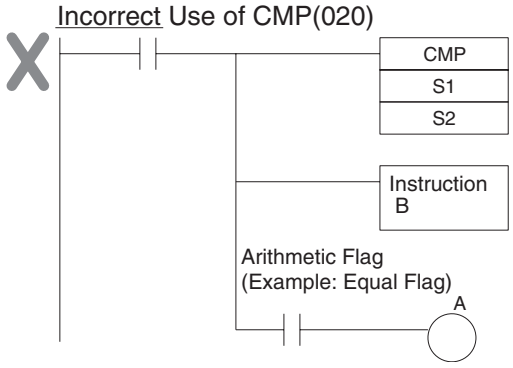
When CMP(020) is executed, the result is reflected in the Arithmetic Flags. Control the desired output or right-hand instruction with a branch from the

same input condition that controls CMP(020), as shown in the following diagram. In this case, the Equals Flag and output A will be turned ON when $S_1 = S_2$.



Using CMP(020) Results in the Program

Do not program another instruction between CMP(020) and the instruction controlled by the Arithmetic Flag because the other instruction might change the status of the Arithmetic Flag. In this case, the results of instruction B might change the results of CMP(020).



Flags

Name	CX-Programmer label	Label	Operation
Greater Than Flag	P_GT	>	ON if $S_1 > S_2$. OFF in all other cases.
Greater Than or Equal Flag	P_GE	> =	ON if $S_1 \geq S_2$. OFF in all other cases.
Equal Flag	P_EQ	=	ON if $S_1 = S_2$. OFF in all other cases.
Not Equal Flag	P_NE	<>	ON if $S_1 \neq S_2$. OFF in all other cases.
Less Than Flag	P_LT	<	ON if $S_1 < S_2$. OFF in all other cases.
Less Than or Equal Flag	P_LE	< =	ON if $S_1 \leq S_2$. OFF in all other cases.

Precautions

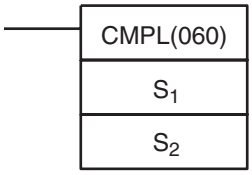
Do not program another instruction between CMP(020) and an input condition that accesses the result of CMP(020) because the other instruction might change the status of the Arithmetic Flags.

3-6-3 DOUBLE COMPARE: CMPL(060)

Purpose

Compares two double unsigned binary values (constants and/or the contents of specified words) and outputs the result to the Arithmetic Flags in the Auxiliary Area.

Ladder Symbol



S1: Comparison data 1

S2: Comparison data 2

Variations

Variations		
	Executed Each Cycle for ON Condition	CMPL(060)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

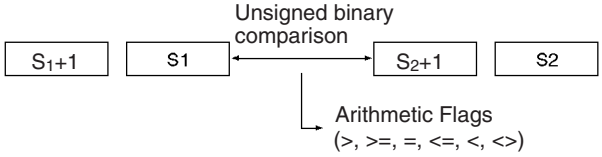
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S_1	S_2
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	
Data Resisters	---	
Index Registers	IR0 to IR15	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

CMPL(060) compares the unsigned binary data in S_1 +1, S_1 and S_2+1, S_2 and outputs the result to Arithmetic Flags (the Greater Than, Greater Than or Equal, Equal, Less Than or Equal, Less Than, and Not Equal Flags) in the Auxiliary Area.



Arithmetic Flag Status

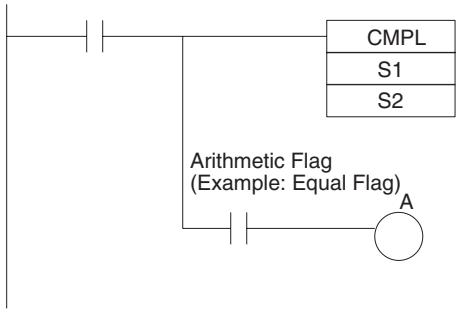
The following table shows the status of the Arithmetic Flags after execution of CMPL(060). (A status of "---" indicates that the Flag may be ON or OFF.)

CMPL(060)Result	Flag status					
	>	> =	=	< =	<	< >
S_1 +1, S_1 > S_2+1, S_2	ON	ON	OFF	OFF	OFF	ON
S_1+1, S_1 = S_2+1, S_2	OFF	ON	ON	ON	OFF	OFF
S_1+1, S_1 < S_2+1, S_2	OFF	OFF	OFF	ON	ON	ON

Using CMPL(060) Results in the Program

When CMPL(060) is executed, the result is reflected in the Arithmetic Flags. Control the desired output or right-hand instruction with a branch from the same input condition that controls CMPL(060), as shown in the following diagram. Here, the Equals Flag and output A will be turned ON when S_1 +1, S_1 = S_2+1, S_2.

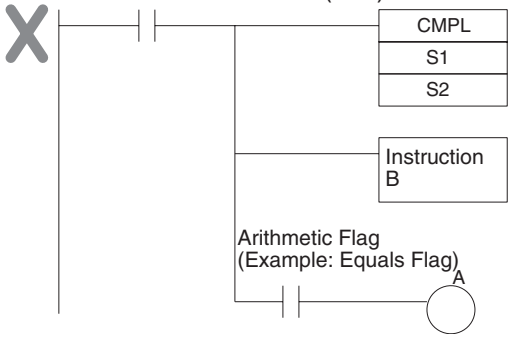
Correct Use of CMPL(060)



Using CMPL(060) Results in the Program

Do not program another instruction between CMPL(060) and the instruction controlled by the Arithmetic Flag because the other instruction might change the status of the Arithmetic Flag. In this case, the results of instruction B might change the results of CMPL(060).

Incorrect Use of CMPL(060)



Flags

Name	CX-Programmer label	Label	Operation
Greater Than Flag	P_GT	>	ON if S_1 +1, S_1 > S_2+1, S_2. OFF in all other cases.
Greater Than or Equal Flag	P_GE	> =	ON if S_1 +1, S_1 ≥ S_2+1, S_2. OFF in all other cases.

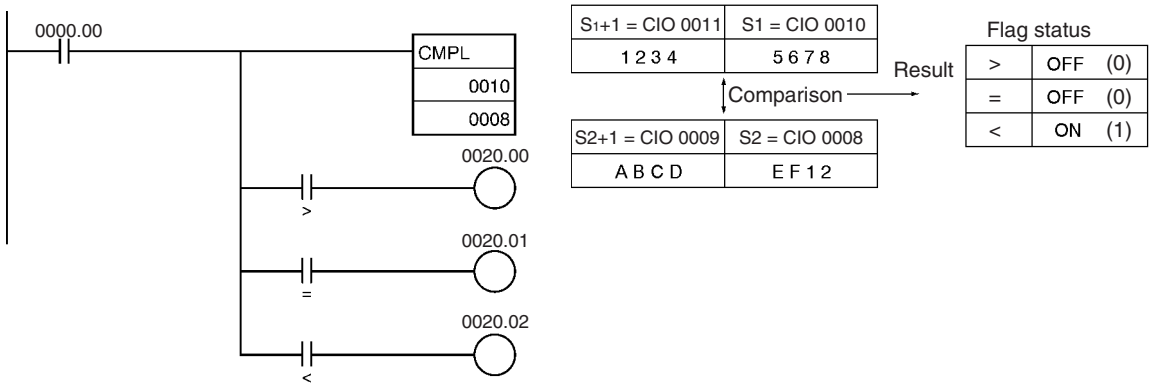
Name	CX-Programmer label	Label	Operation
Equal Flag	P_EQ	=	ON if S_1 +1, S_1 = S_2+1, S_2. OFF in all other cases.
Not Equal Flag	P_NE	<>	ON if S_1 +1, S_1 ≠ S_2+1, S_2. OFF in all other cases.
Less Than Flag	P_LT	<	ON if S_1 +1, S_1 < S_2+1, S_2. OFF in all other cases.
Less Than or Equal Flag	P_LE	< =	ON if S_1 +1, S_1 ≤ S_2+1, S_2. OFF in all other cases.

Precautions

Do not program another instruction between CMPL(060) and an input condition that accesses the result of CMPL(060) because the other instruction might change the status of the Arithmetic Flags.

Example

When CIO 0000.00 is ON in the following example, the eight-digit unsigned binary data in CIO 0011 and CIO 0010 is compared to the eight-digit unsigned binary data in CIO 0009 and CIO 0008 and the result is output to the Arithmetic Flags. The results recorded in the Greater Than, Equals, and Less Than Flags are immediately saved to CIO 0002.00 (Greater Than), CIO 0002.01 (Equals), and CIO 0002.02 (Less Than).

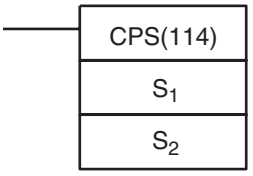


3-6-4 SIGNED BINARY COMPARE: CPS(114)

Purpose

Compares two signed binary values (constants and/or the contents of specified words) and outputs the result to the Arithmetic Flags in the Auxiliary Area.

Ladder Symbol



S1: Comparison data 1

S2: Comparison data 2

Variations

Variations		
	Executed Each Cycle for ON Condition	CPS(114)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

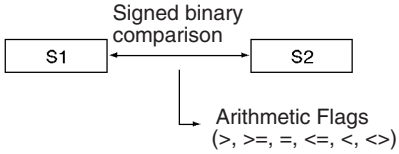
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S_1	S_2
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	
Auxiliary Bit Area	A000 to A959	
Timer Area	T0000 to T0255	
Counter Area	C0000 to C0255	
DM Area	D00000 to D32767	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 to #FFFF (binary)	
Data Resisters	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

CPS(114) compares the signed binary data in S_1 and S_2 and outputs the result to Arithmetic Flags (the Greater Than, Greater Than or Equal, Equal, Less Than or Equal, Less Than, and Not Equal Flags) in the Auxiliary Area.



Note CPS(114) treats the data in S_1 and S_2 as signed binary data which ranges from 8000 to 7FFF (–32,768 to 32,767 decimal).

Arithmetic Flag Status

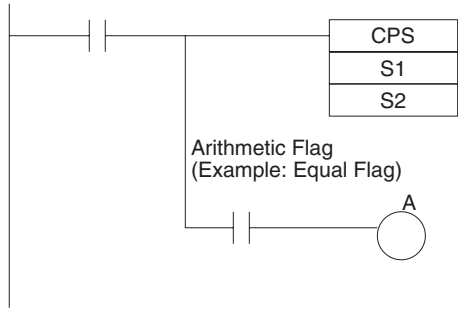
The following table shows the status of the Arithmetic Flags after execution of CPS(114). (A status of "---" indicates that the Flag may be ON or OFF.)

CPS(114) Result	Flag status					
	>	> =	=	< =	<	< >
$S_1 > S_2$	ON	ON	OFF	OFF	OFF	ON
$S_1 = S_2$	OFF	ON	ON	ON	OFF	OFF
$S_1 < S_2$	OFF	OFF	OFF	ON	ON	ON

Using CPS(114) Results in the Program

When CPS(114) is executed, the result is reflected in the Arithmetic Flags. Control the desired output or right-hand instruction with a branch from the same input condition that controls CPS(114), as shown in the following diagram. In this case, the Equals Flag and output A will be turned ON when $S_1 = S_2$.

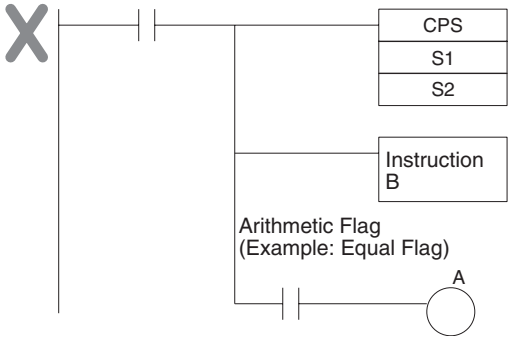
Correct Use of CPS(114)



Using CPS(114) Results in the Program

Do not program another instruction between CPS(114) and the instruction controlled by the Arithmetic Flag because the other instruction might change the status of the Arithmetic Flag. In this case, the results of instruction B might change the results of CPS(114).

Incorrect Use of CPS(114)



Flags

Name	Label	Operation
Greater Than Flag	>	ON if $S_1 > S_2$. OFF in all other cases.
Greater Than or Equal Flag	> =	ON if $S_1 \geq S_2$. OFF in all other cases.
Equal Flag	=	ON if $S_1 = S_2$. OFF in all other cases.
Not Equal Flag	<>	ON if $S_1 \neq S_2$. OFF in all other cases.
Less Than Flag	<	ON if $S_1 < S_2$. OFF in all other cases.
Less Than or Equal Flag	< =	ON if $S_1 \leq S_2$. OFF in all other cases.

Precautions

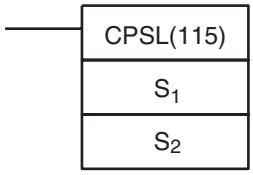
Do not program another instruction between CPS(114) and an input condition that accesses the result of CPS(114) because the other instruction might change the status of the Arithmetic Flags.

3-6-5 DOUBLE SIGNED BINARY COMPARE: CPSL(115)

Purpose

Compares two double signed binary values (constants and/or the contents of specified words) and outputs the result to the Arithmetic Flags in the Auxiliary Area.

Ladder Symbol



S1: Comparison data 1

S2: Comparison data 2

Variations

Variations		
	Executed Each Cycle for ON Condition	CPSL(115)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

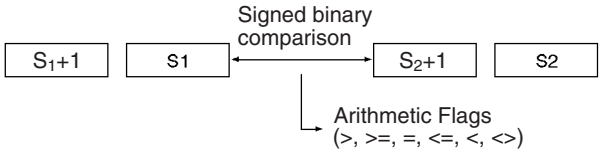
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S_1	S_2
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

CPSL(115) compares the double signed binary data in S_1 +1, S_1 and S_2+1, S_2 and outputs the result to Arithmetic Flags (the Greater Than, Greater Than or Equal, Equal, Less Than or Equal, Less Than, and Not Equal Flags) in the Auxiliary Area.



Note CPSL(115) treats the data in S_1 and S_2 as double signed binary data which ranges from 8000 0000 to 7FFF FFFF (–2,147,483,648 to 2,147,483,647 decimal).

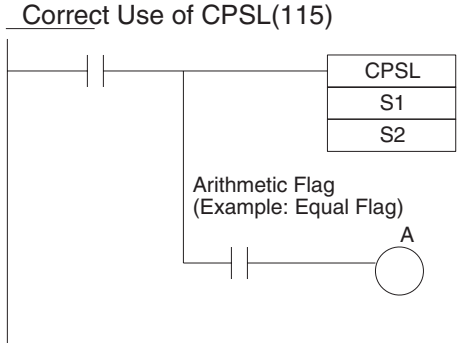
Arithmetic Flag Status

The following table shows the status of the Arithmetic Flags after execution of CPSL(115). (A status of "---" indicates that the Flag may be ON or OFF.)

CPSL(115)Result	Flag status					
	>	> =	=	< =	<	< >
S_1 +1, S_1 > S_2+1, S_2	ON	ON	OFF	OFF	OFF	ON
S_1+1, S_1 = S_2+1, S_2	OFF	ON	ON	ON	OFF	OFF
S_1+1, S_1 < S_2+1, S_2	OFF	OFF	OFF	ON	ON	ON

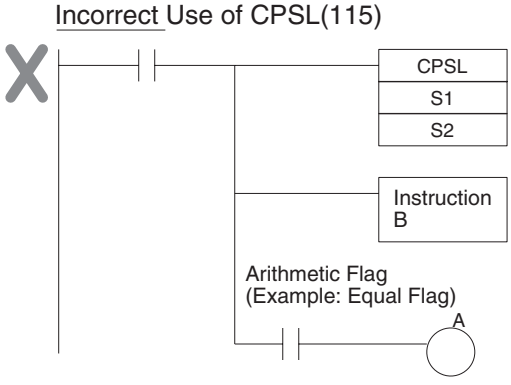
Using CPSL(115) Results in the Program

When CPSL(115) is executed, the result is reflected in the Arithmetic Flags. Control the desired output or right-hand instruction with a branch from the same input condition that controls CPSL(115), as shown in the following diagram. Here, the Equals Flag and output A will be turned ON when S_1 +1, S_1 = S_2+1, S_2.



Using CPSL(115) Results in the Program

Do not program another instruction between CPSL(115) and the instruction controlled by the Arithmetic Flag because the other instruction might change the status of the Arithmetic Flag. In this case, the results of instruction B might change the results of CPSL(115).



Flags

Name	Label	Operation
Greater Than Flag	>	ON if S_1 +1, $S_1 > S_2$+1, S_2. OFF in all other cases.
Greater Than or Equal Flag	> =	ON if S_1 +1, $S_1 \geq S_2$+1, S_2. OFF in all other cases.
Equal Flag	=	ON if S_1 +1, $S_1 = S_2$+1, S_2. OFF in all other cases.
Not Equal Flag	<>	ON if S_1 +1, $S_1 \neq S_2$+1, S_2. OFF in all other cases.
Less Than Flag	<	ON if S_1 +1, $S_1 < S_2$+1, S_2. OFF in all other cases.
Less Than or Equal Flag	< =	ON if S_1 +1, $S_1 \leq S_2$+1, S_2. OFF in all other cases.

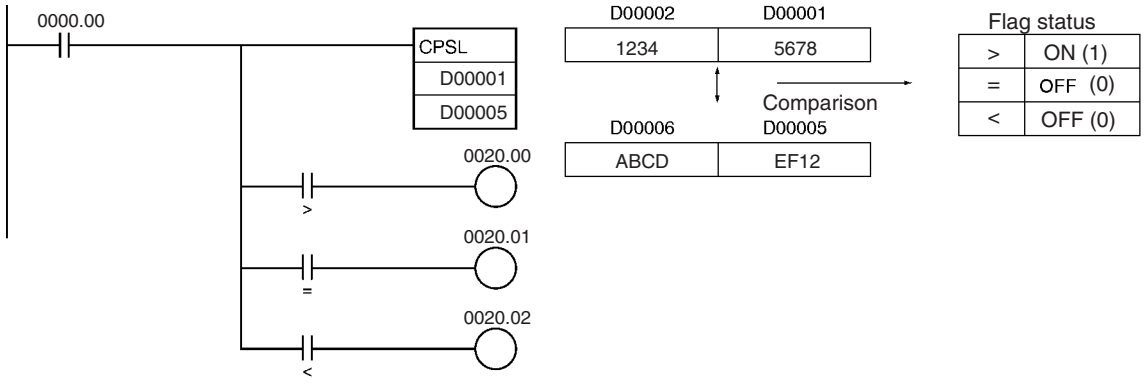
Precautions

Do not program another instruction between CPSL(115) and an input condition that accesses the result of CPSL(115) because the other instruction might change the status of the Arithmetic Flags.

Example

When CIO 0000.00 is ON in the following example, the eight-digit signed binary data in D00002 and D00001 is compared to the eight-digit signed binary data in D00006 and D00005 and the result is output to the Arithmetic Flags.

- If the content of D00002 and D00001 is greater than that of D00006 and D00005, the Greater Than Flag will be turned ON, causing CIO 0020.00 to be turned ON.
- If the content of D00002 and D00001 is equal to that of D00006 and D00005, the Equals Flag will be turned ON, causing CIO 0020.01 to be turned ON.
- If the content of D00002 and D00001 is less than that of D00006 and D00005, the Less Than Flag will be turned ON, causing CIO 0020.02 to be turned ON.

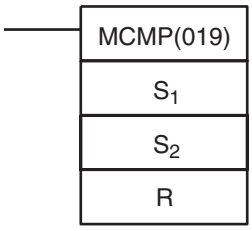


3-6-6 MULTIPLE COMPARE: MCMP(019)

Purpose

Compares 16 consecutive words with another 16 consecutive words and turns ON the corresponding bit in the result word where the contents of the words **are not** equal.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	MCMP(019)
	Executed Once for Upward Differentiation	@MCMP(019)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S_1: First word of set 1

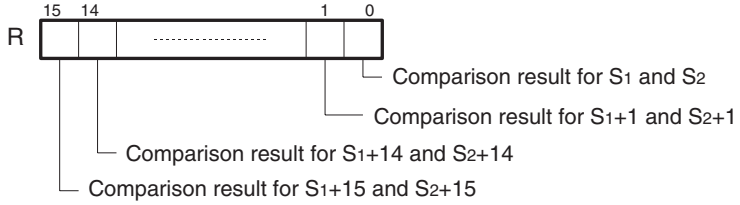
Specifies the beginning of the first 16-word range. S_1 and S_1+15 must be in the same data area.

S_2: First word of set 2

Specifies the beginning of the second 16-word range. S_2 and S_2+15 must be in the same data area.

R: Result word

Each bit of R contains the result of a comparison between two words in the 16-word sets. Bit n of R (n = 00 to 15) contains the result of the comparison between words S_1+n and S_2+n.



Operand Specifications

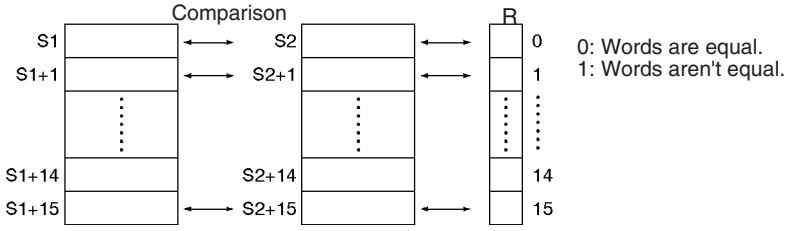
Area	S_1	S_2	R
CIO Area	CIO 0000 to CIO 6128		CIO 0000 to CIO 6143
Work Area	W000 to W240		W000 to W255
Auxiliary Bit Area	A000 to A944		A448 to A959
Timer Area	T0000 to T0240		T0000 to T0255
Counter Area	C0000 to C0240		C0000 to C0255
DM Area	D00000 to D32752		D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---		
Data Resisters	---		DR0 to DR15

Area	S_1	S_2	R
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

MCMP(019) compares the contents of the 16 words S_1 through S_1+15 to the contents of the 16 words S_2 through S_2+15, and turns ON the corresponding bit in word R when the contents **are not** equal.

The content of S_1 is compared to the content of S_2, the content of S_1+1 to the content of S_2+1, ..., and the content of S_1+15 to the content of S_2+15. Bit n of R is turned OFF if the content of S_1+n is equal to the content of S_2+n; bit n of R is turned ON if the contents are not equal. If the contents of all 16 pairs of words are the same, the Equals Flag will turn ON after the instruction has been executed.

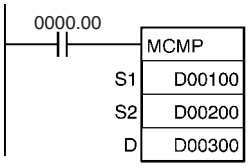


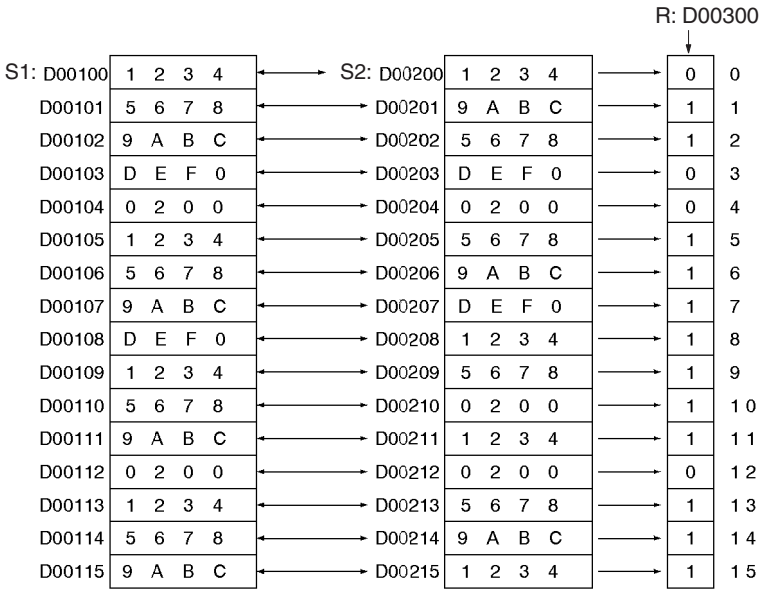
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result word is 0000. (The two 16-word sets contain the same data.) OFF in all other cases.

Example

When CIO 0000.00 is ON in the following example, MCMP(019) compares words D00100 through D00115 in order to words D00200 through D00215 and turns ON the corresponding bits in D00300 when the words **are not** equal.



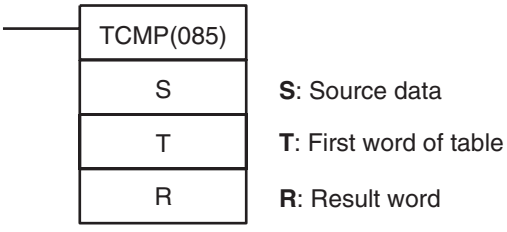


3-6-7 TABLE COMPARE: TCMP(085)

Purpose

Compares the source data to the contents of 16 consecutive words and turns ON the corresponding bit in the result word when the contents of the words **are** equal.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	TCMP(085)
	Executed Once for Upward Differentiation	@TCMP(085)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

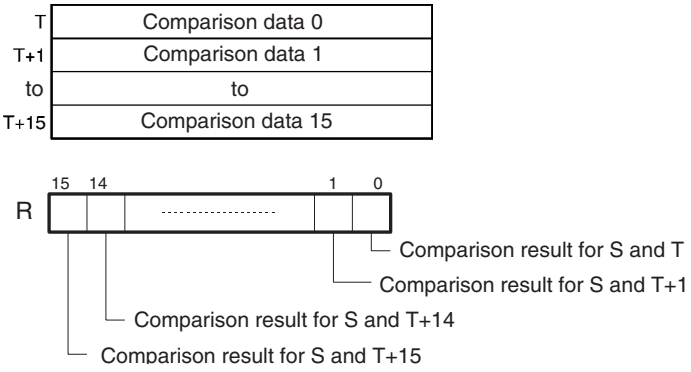
Operands

T: First word of table

Specifies the beginning of the 16-word table. T and T+15 must be in the same data area.

R: Result word

Each bit of R contains the result of a comparison between S and a word in the 16-word table. Bit n of R (n = 00 to 15) contains the result of the comparison between S and T+n.



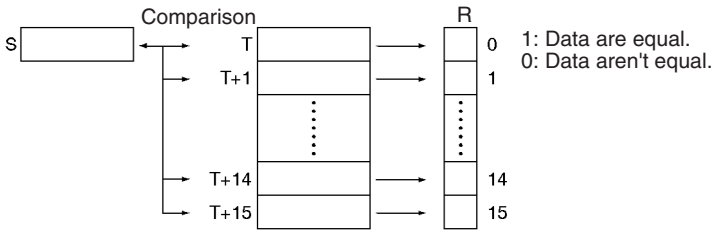
Operand Specifications

Area	S	T	R
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6128	CIO 0000 to CIO 6143
Work Area	W000 to W255	W000 to W240	W000 to W255
Auxiliary Bit Area	A000 to A959	A000 to A944	A448 to A959
Timer Area	T0000 to T0255	T0000 to T0240	T0000 to T0255
Counter Area	C0000 to C0255	C0000 to C0240	C0000 to C0255
DM Area	D00000 to D32767	D00000 to D32752	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)	---	
Data Resisters	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

TCMP(085) compares the source data (S) to each of the 16 words T through T+15 and turns ON the corresponding bit in word R when the data **are** equal. Bit n of R is turned ON if the content of T+n is equal to S and it is turned OFF if they are not equal.

S is compared to the content of T and bit 00 of R is turned ON if they are equal or OFF if they are not equal, S is compared to the content of T+1 and bit 01 of R is turned ON if they are equal or OFF if they are not equal, ..., and S is compared to the content of T+15 and bit 15 of R is turned ON if they are equal or OFF if they are not equal.

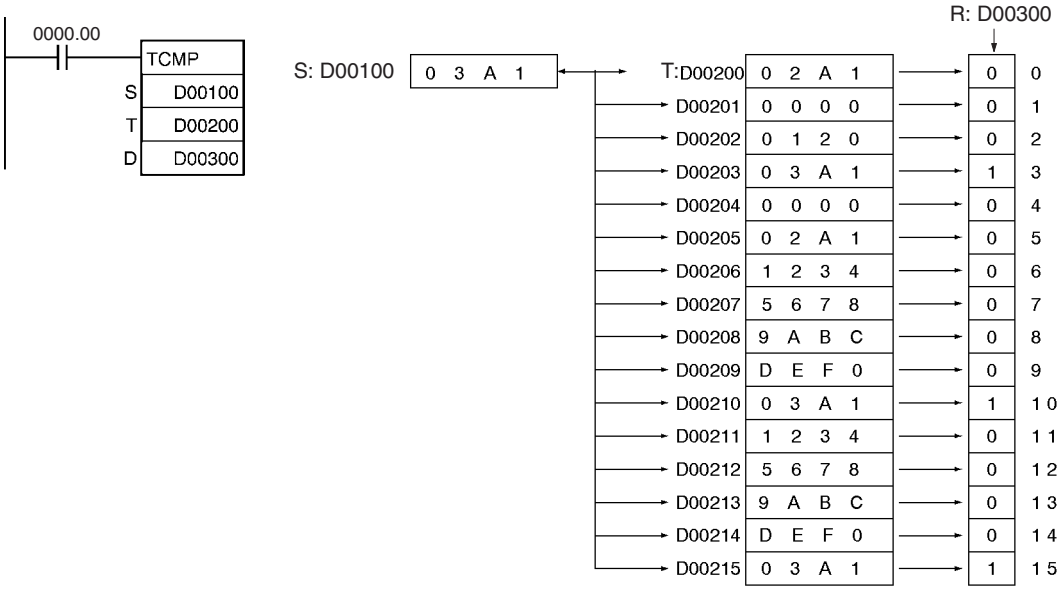


Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result word is 0000. (None of the 16 words in the table equals S.) OFF in all other cases.

Example

When CIO 0000.00 is ON in the following example, TCMP(085) compares the content of D00100 with the contents of words D00200 through D00215 and turns ON the corresponding bits in D00300 when the contents are equal or OFF when the contents are not equal.

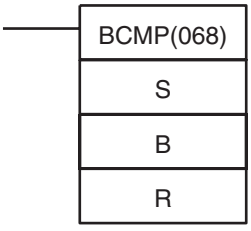


3-6-8 BLOCK COMPARE: BCMP(068)

Purpose

Compares the source data to 16 ranges (defined by 16 lower limits and 16 upper limits) and turns ON the corresponding bit in the result word when the source data is within a range.

Ladder Symbol



S: Source data

B: First word of block

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	BCMP(068)
	Executed Once for Upward Differentiation	@BCMP(068)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

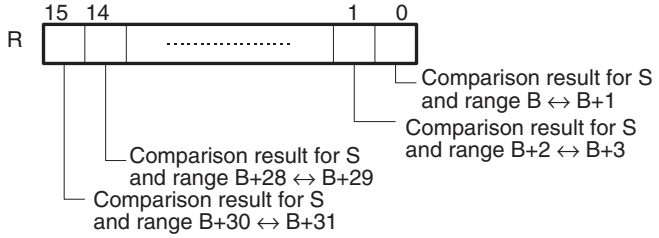
Operands

B: First word of block

Specifies the beginning of a 32-word block (16 lower/upper limit pairs). B and B+31 must be in the same data area.

R: Result word

Each bit of R contains the result of a comparison between S and one of the 16 ranges defined the 32-word block. Bit n of R (n = 00 to 15) contains the result of the comparison between S and the n^{th} pair of words.



Operand Specifications

Area	S	B	R
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6112	CIO 0000 to CIO 6143
Work Area	W000 to W255	W0000 to W224	W000 to W255
Auxiliary Bit Area	A000 to A959	A000 to A928	A448 to A959
Timer Area	T0000 to T0255	T0000 to T0224	T0000 to T0255
Counter Area	C0000 to C0255	C0000 to C0224	C0000 to C0255
DM Area	D00000 to D32767	D00000 to D32736	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)	---	
Data Resisters	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

BCMP(068) compares the source data (S) to the 16 ranges defined by pairs of lower and upper limit values in B through B+31. The first word in each pair (B+2n) provides the lower limit and the second word (B+2n+1) provides the upper limit of range n (n = 0 to 15). If S is within any of these ranges (inclusive of the upper and lower limits), the corresponding bit in R is turned ON. The rest of the bits in R will be turned OFF.

B	≤ S ≤	B+1	Bit 00 of R
B+2	≤ S ≤	B+3	Bit 01 of R
B+4	≤ S ≤	B+5	Bit 02 of R
B+6	≤ S ≤	B+7	Bit 03 of R
B+8	≤ S ≤	B+9	Bit 04 of R
B+10	≤ S ≤	B+11	Bit 05 of R
B+12	≤ S ≤	B+13	Bit 06 of R
B+14	≤ S ≤	B+15	Bit 07 of R

B+16	≤ S ≤	B+17	Bit 08 of R
B+18	≤ S ≤	B+19	Bit 09 of R
B+20	≤ S ≤	B+21	Bit 10 of R
B+22	≤ S ≤	B+23	Bit 11 of R
B+24	≤ S ≤	B+25	Bit 12 of R
B+26	≤ S ≤	B+27	Bit 13 of R
B+28	≤ S ≤	B+29	Bit 14 of R
B+30	≤ S ≤	B+31	Bit 15 of R

For example, bit 00 of R is turned ON if S is within the first range (B ≤ S ≤ B+1), bit 01 of R is turned ON if S is within the second range (B+2 ≤ S ≤ B+3), ..., and bit 15 of R is turned ON if S is within the fifteenth range (B+30 ≤ S ≤ B+31). All other bits in R are turned OFF.

Flags

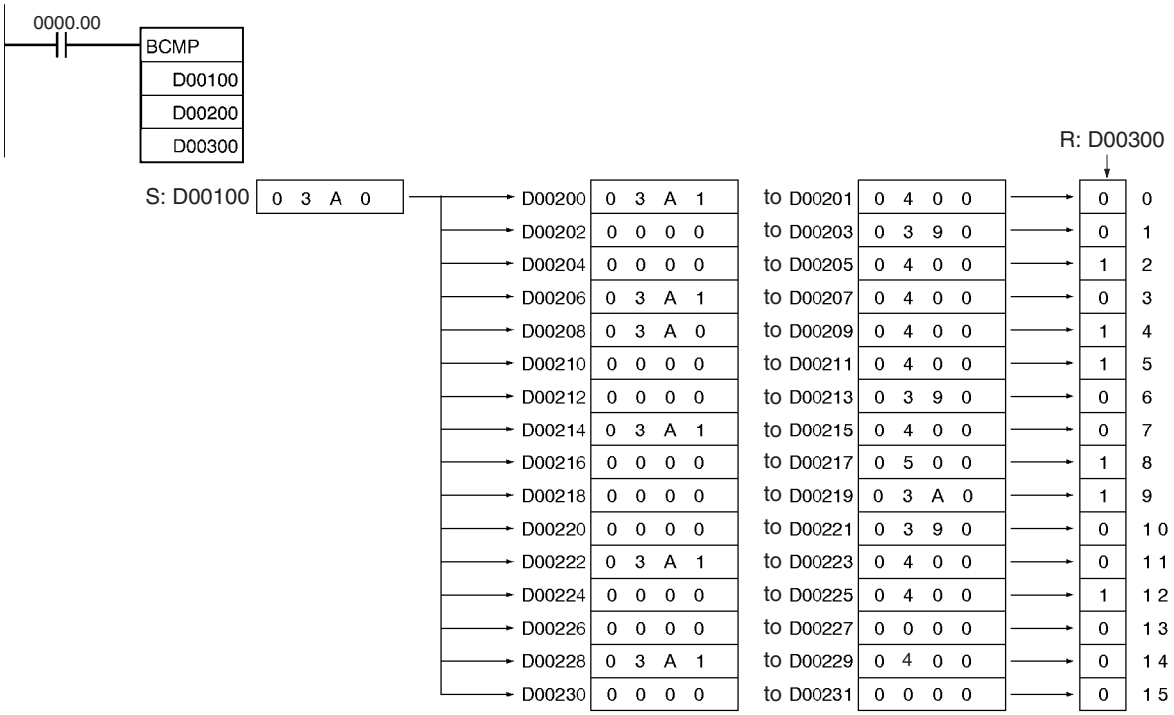
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result word is 0000. (S is not within any of the 16 ranges.) OFF in all other cases.

Precautions

An error will not occur if the lower limit is greater than the upper limit, but 0 (not within the range) will be output to the corresponding bit of R.

Example

When CIO 0000.00 is ON in the following example, BCMP(068) compares the content of D00100 with the 16 ranges defined in D00200 through D00231 and turns ON the corresponding bits in D00300 when S is within the range or OFF when S is not within the range.

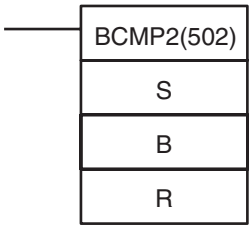


3-6-9 EXPANDED BLOCK COMPARE: BCMP2(502)

Purpose

Compares the source data to up to 256 ranges (defined by 256 lower limits and 256 upper limits) and turns ON the corresponding bit in the result word when the source data is within a range.

Ladder Symbol



S: Source data

B: First word of block

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	BCMP2(502)
	Executed Once for Upward Differentiation	@BCMP2(502)
	Executed Once for Downward Differentiation	Not supported

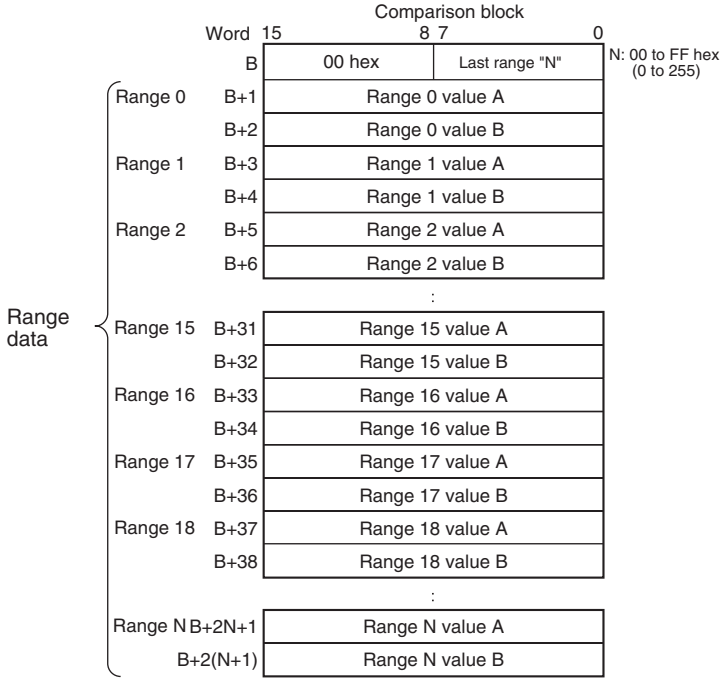
Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

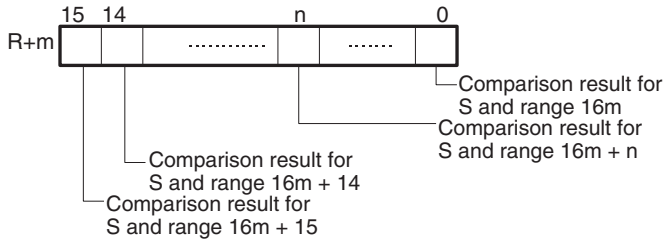
B: First word of block

Specifies the beginning of a comparison block containing up to 513 words including up to 256 lower/upper limit pairs). All words must be in the same data area.



R: First result word

Each bit of each R word contains the result of a comparison between S and one of the ranges defined by the comparison block. The maximum number of result words is 16, i.e., m equals 0 to 15.



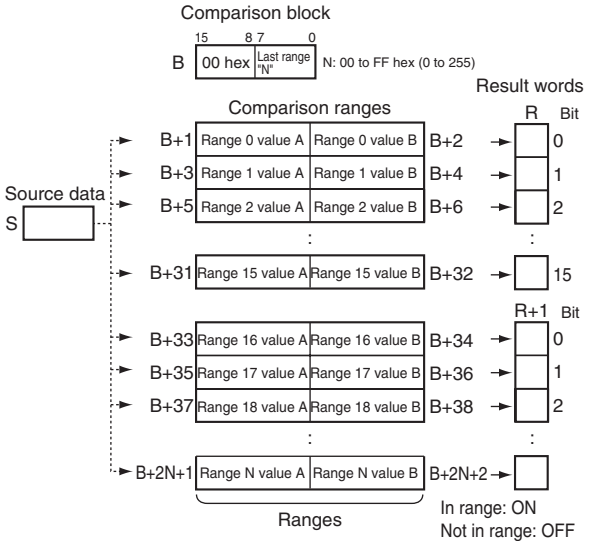
Operand Specifications

Area	S	B	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)	---	
Data Resisters	DR0 to DR15	---	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

BCMP2(502) compares the source data (S) to the ranges defined by pairs of lower and upper limit values in the comparison block. If S is within any of these ranges (inclusive of the upper and lower limits), the corresponding bits in the result words (R to R+15 max.) are turned ON. The rest of the bits in R will be turned OFF.

The number of ranges is determined by the value N set in the lower byte of B. N can be between 0 and 255. The upper byte of B must be 00 hex.



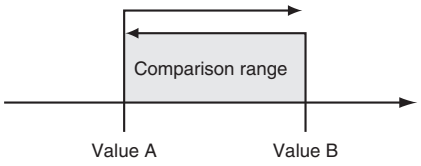
Number of Ranges

The number of ranges in the comparison block is set in the first word of the block. Up to 256 ranges can be set.

Setting Ranges

The values A and B for each range will determine how the comparison operates depending on which value is larger, as shown below.

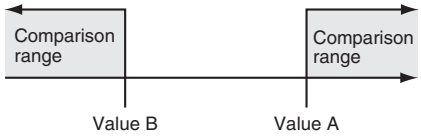
· If Value A ≤ Value B
Then, Value A ≤ Comparison range ≤ Value B



Example

When Value A ≤ Value B
If B+1 ≤ S ≤ B+2, then bit 0 of R will turn ON,
If B+3 ≤ S ≤ B+4, then bit 1 of R will turn ON,
If S < B+5 and B+6 < S, then bit 2 of R will turn OFF, and
If S < B+7 and B+8 < S, then bit 3 of R will turn OFF.

· If Value A > Value B
Then, Comparison range ≤ Value B and Value A ≤ Comparison range

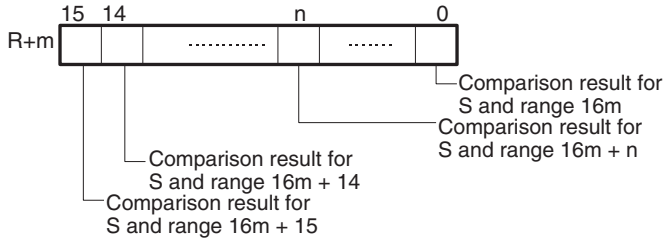


Example

When Value A > Value B
If S ≤ B+2 and B+1 ≤ S, then bit 0 of R will turn ON,
If S ≤ B+4 and B+3 ≤ S, then bit 1 of R will turn ON,
If B+6 < S < B+5, then bit 2 of R will turn OFF, and
If B+8 < S < B+7, then bit 3 of R will turn OFF.

Results Storage Location

The results are output to corresponding bits in word R. If there are more than 16 comparison ranges, consecutive words following R will be used. The maximum number of result words is 16, i.e., m equals 0 to 15.

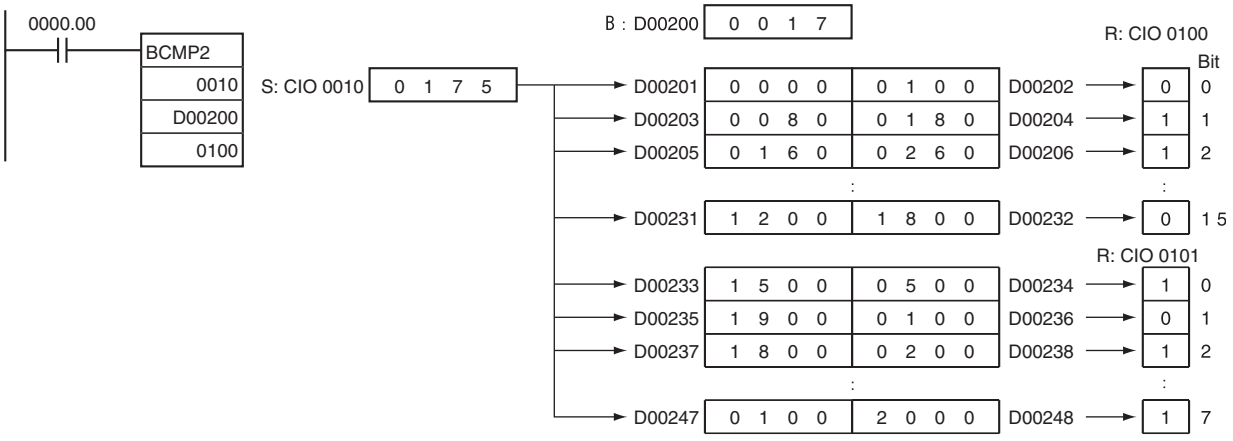


Flags

Name	Label	Operation
Error Flag	ER	OFF

Example

When CIO 0000.00 is ON in the following example, BCMP2(502) compares the content of CIO 0010 with the 24 ranges defined in D00200 through D00247 (N = 17 hex = 23 decimal, i.e., 24 ranges) and turns ON the corresponding bits in CIO 0100 and CIO 0101 when S is within the range and OFF when S is not within the range. For example, if the source data in CIO 0010 is in the range defined by D00201 and D00202, then bit 00 of CIO 0100 is turned ON and if it in not in the range, then bit 00 of CIO 0100 is turned OFF. Likewise, the source data in CIO 0010 is compared to the ranges defined by D00203 and D00204, D00247 and D00248, and the other words in the comparison block, and bit 1 in CIO 0100, bit 7 in CIO 0101, and the other bits in the result words are manipulated according to the results of comparison.

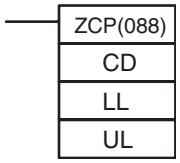


3-6-10 AREA RANGE COMPARE: ZCP(088)

Purpose

Compares a 16-bit unsigned binary value (CD) with the range defined by lower limit LL and upper limit UL. The results are output to the Arithmetic Flags.

Ladder Symbol



CD: Comparison Data
LL: Lower limit of range
UL: Upper limit of range

Variations

Variations		
	Executed Each Cycle for ON Condition	ZCP(088)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	CD	LL	UL
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)		
Data Resisters	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

ZCP(088) compares the 16-bit unsigned binary data in CD with the range defined by LL and UL and outputs the result to the Greater Than, Equals, and Less Than Flags in the Auxiliary Area. (The Less Than or Equal, Greater Than or Equal, and Not Equal Flags are left unchanged.)

Arithmetic Flag Status

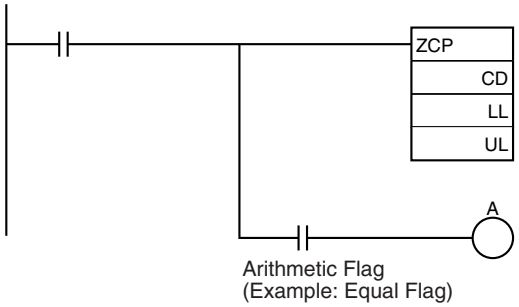
The following table shows the status of the Arithmetic Flags after execution of ZCP(088).

ZCP(088)Result	Flag status		
	>	=	<
CD > UL	ON	OFF	OFF
CD = UL	OFF	ON	
LL < CD < UL			
CD = LL			
CD < LL		OFF	ON

Using ZCP(088) Results in the Program

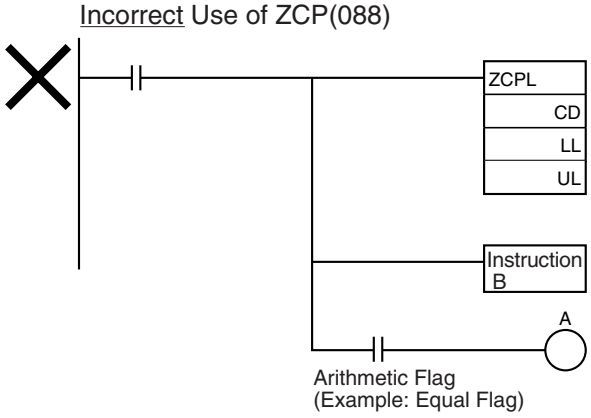
When ZCP(088) is executed, the result is reflected in the Arithmetic Flags. Control the desired output or right-hand instruction with a branch from the same input condition that controls ZCP(088), as shown in the following diagram. In this case, the Equals Flag and output A will be turned ON when LL ≤ CD ≤ UL.

Correct Use of ZCP(088)



Do not program another instruction between ZCP(088) and the instruction controlled by the Arithmetic Flag because the other instruction might change the status of the Arithmetic Flag. In this case, the results of instruction B might change the results of ZCP(088).

Incorrect Use of ZCP(088)



Flags

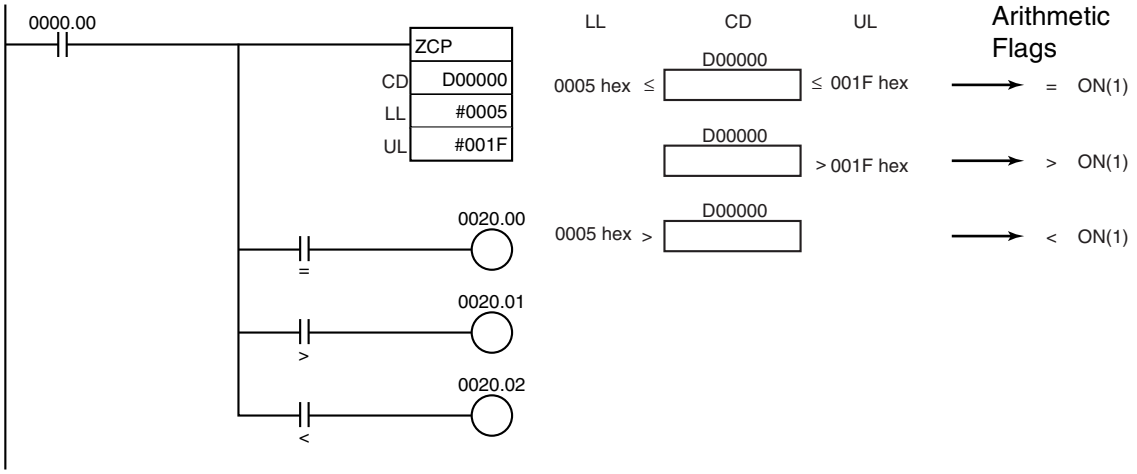
Name	Label	Operation
Error Flag	ER	ON if LL > UL.
Greater Than Flag	>	ON if CD > UL. OFF in all other cases.
Greater Than or Equal Flag	> =	Left unchanged.
Equal Flag	=	ON if LL ≤ CD ≤ UL. OFF in all other cases.
Not Equal Flag	<>	Left unchanged.
Less Than Flag	<	ON if CD < LL. OFF in all other cases.
Less Than or Equal Flag	< =	Left unchanged.
Negative Flag	N	Left unchanged.

Precautions

Do not program another instruction between ZCP(088) and an input condition that accesses the result of ZCP(088) because the other instruction might change the status of the Arithmetic Flags.

Example

When CIO 0000.00 is ON in the following example, the 16-bit unsigned binary data in D00000 is compared to the range 0005 to 001F hex (5 to 31 decimal) and the result is output to the Arithmetic Flags.
CIO 0020.00 is turned ON if 0005 hex ≤ content of D00000 ≤ 001F hex.
CIO 0020.01 is turned ON if the content of D00000 > 001F hex.
CIO 0020.02 is turned ON if the content of D00000 < 0005 hex.

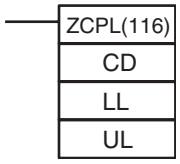


3-6-11 DOUBLE AREA RANGE COMPARE: ZCPL(116)

Purpose

Compares a 32-bit unsigned binary value (CD+1, CD) with the range defined by lower limit (LL+1, LL) and upper limit (UL+1, UL). The results are output to the Arithmetic Flags.

Ladder Symbol



CD: First word of Comparison Data
LL: First word of Lower Limit
UL: First word of Upper Limit

Variations

Variations		
	Executed Each Cycle for ON Condition	ZCPL(116)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	CD	LL	UL
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		

Area	CD	LL	UL
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)		
Data Resisters	---		
Index Registers	IR0 to IR15		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

ZCPL(116) compares the 32-bit unsigned binary data in CD+1, CD with the range defined by LL+1, LL and UL+1, UL and outputs the result to the Greater Than, Equals, and Less Than Flags in the Auxiliary Area. (The Less Than or Equal, Greater Than or Equal, and Not Equal Flags are left unchanged.)

Arithmetic Flag Status

The following table shows the status of the Arithmetic Flags after execution of ZCPL(116).

ZCPL(116) result	Flag status		
	>	=	<
CD+1, CD > UL+1, UL	ON	OFF	OFF
CD+1, CD = UL+1, UL	OFF	ON	
LL+1, LL < CD+1, CD < UL+1, UL			
CD+1, CD = LL+1, LL			
CD+1, CD < LL+1, LL		OFF	ON

Using ZCPL(116) Results in the Program

When ZCPL(116) is executed, the result is reflected in the Arithmetic Flags. Control the desired output or right-hand instruction with a branch from the same input condition that controls ZCPL(116).

Do not program another instruction between ZCPL(116) and the instruction controlled by the Arithmetic Flag because the other instruction might change the status of the Arithmetic Flag.

The operation of ZCPL(116) is almost identical to that of ZCP(088) except that ZCPL(116) compares 32-bit values instead of 16-bit values. Refer to *3-6-10 AREA RANGE COMPARE: ZCP(088)* for diagrams showing how to use results in the program and an example program section.

Flags

Name	Label	Operation
Error Flag	ER	ON if LL+1, LL > UL+1, UL.
Greater Than Flag	>	ON if CD+1, CD > UL+1, UL. OFF in all other cases.
Greater Than or Equal Flag	> =	Left unchanged.
Equal Flag	=	ON if LL+1, LL ≤ CD+1, CD ≤ UL+1, UL. OFF in all other cases.
Not Equal Flag	<>	Left unchanged.
Less Than Flag	<	ON if CD+1, CD < LL+1, LL. OFF in all other cases.

Name	Label	Operation
Less Than or Equal Flag	< =	Left unchanged.
Negative Flag	N	Left unchanged.

Precautions

Do not program another instruction between ZCPL(116) and an input condition that accesses the result of ZCPL(116) because the other instruction might change the status of the Arithmetic Flags.

3-7 Data Movement Instructions

This section describes instructions used to move data in various ways.

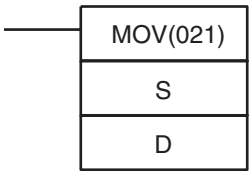
Instruction	Mnemonic	Function code	Page
MOVE	MOV	021	199
DOUBLE MOVE	MOVL	498	201
MOVE NOT	MVN	022	200
DOUBLE MOVE NOT	MVNL	499	203
MOVE BIT	MOVB	082	204
MOVE DIGIT	MOVD	083	206
MULTIPLE BIT TRANSFER	XFRB	062	208
BLOCK TRANSFER	XFER	070	211
BLOCK SET	BSET	071	213
DATA EXCHANGE	XCHG	073	215
SINGLE WORD DISTRIBUTE	DIST	080	217
DATA COLLECT	COLL	081	219
DOUBLE DATA EXCHANGE	XCGL	562	216
MOVE TO REGISTER	MOVR	560	221
MOVE TIMER/COUNTER PV TO REGISTER	MOVRW	561	222

3-7-1 MOVE: MOV(021)

Purpose

Transfers a word of data to the specified word.

Ladder Symbol



S: Source

D: Destination

Variations

Variations		
	Executed Each Cycle for ON Condition	MOV(021)
	Executed Once for Upward Differentiation	@MOV(021)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

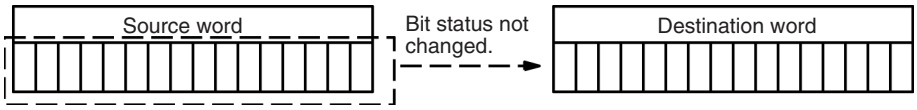
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	
Auxiliary Bit Area	A000 to A959	A448 to A959
Timer Area	T0000 to T0255	
Counter Area	C0000 to C0255	
DM Area	D00000 to D32767	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 to #FFFF (binary)	---

Area	S	D
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15,IR0 to DR0 to DR15,IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

Transfers S to D. If S is a constant, the value can be used for a data setting.

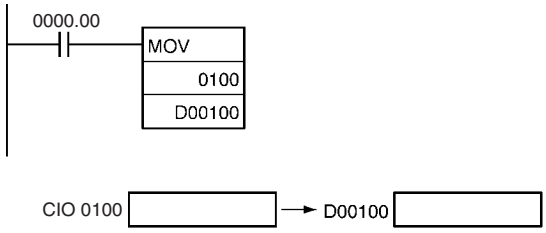


Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the data being transferred is 0000. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of the data being transferred is 1. OFF in all other cases.

Example

When CIO 0000.00 is ON in the following example, the content of CIO 0100 is copied to D00100.

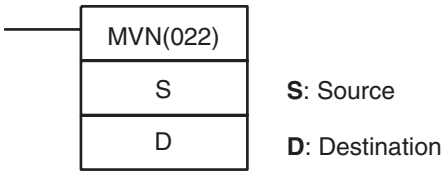


3-7-2 MOVE NOT: MVN(022)

Purpose

Transfers the complement of a word of data to the specified word.

Ladder Symbol



S: Source

D: Destination

Variations

Variations		
	Executed Each Cycle for ON Condition	MVN(022)
	Executed Once for Upward Differentiation	@MVN(022)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	

Area	S	D
Auxiliary Bit Area	A000 to A959	A448 to A959
Timer Area	T0000 to T0255	
Counter Area	C0000 to C0255	
DM Area	D00000 to D32767	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 to #FFFF (binary)	---
Data Resisters	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 , IR0 to –2048 to +2047 ,IR15 DR0 to DR15,IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

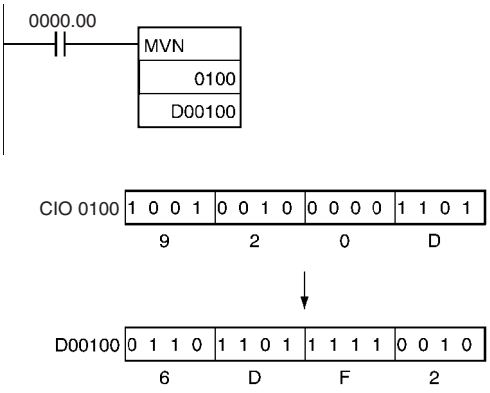
MVN(022) inverts the bits in S and transfers the result to D. The content of S is left unchanged.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the content of D is 0000 after execution. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of D is 1 after execution. OFF in all other cases.

Example

When CIO 0000.00 is ON in the following example, the status of the bits in CIO 0100 is inverted and the result is copied to D00100.

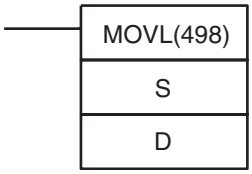


3-7-3 DOUBLE MOVE: MOVL(498)

Purpose

Transfers two words of data to the specified words.

Ladder Symbol



S: First source word

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	MOVL(498)
	Executed Once for Upward Differentiation	@MOVL(498)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

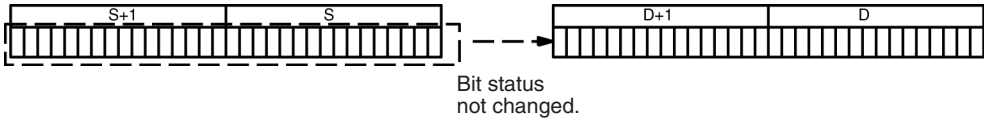
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	---
Data Resisters	---	
Index Registers	IR0 to IR15	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

MOVL(498) transfers S+1 and S to D+1 and D. If S+1 and S are constants, the value can be used for a data setting.

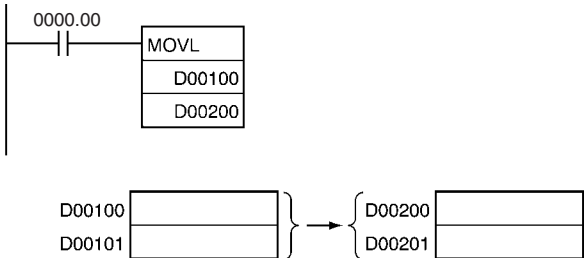


Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the contents of D+1 and D are 0000 0000 after execution. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of D+1 is 1 after execution. OFF in all other cases.

Example

When CIO 0000.00 is ON in the following example, the content of D00101 and D00100 are copied to D00201 and D00200.

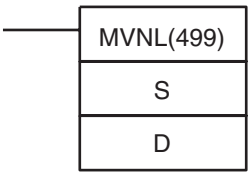


3-7-4 DOUBLE MOVE NOT: MVNL(499)

Purpose

Transfers the complement of two words of data to the specified words.

Ladder Symbol



S: First source word

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	MVNL(499)
	Executed Once for Upward Differentiation	@MVNL(499)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

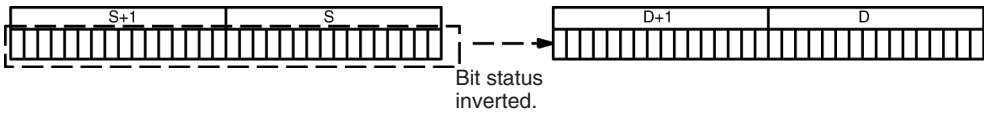
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	---
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

MVNL(499) inverts the bits in S+1 and S and transfers the result to D+1 and D. The contents of S+1 and S are left unchanged.

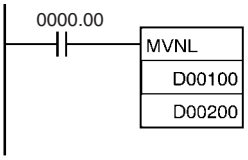


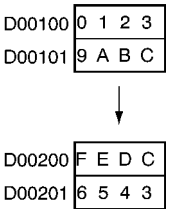
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the contents of D+1 and D are 0000 0000 after execution. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of D+1 is 1 after execution. OFF in all other cases.

Examples

When CIO 0000.00 is ON in the following example, the status of the bits in D00101 and D00100 are inverted and the result is copied to D00201 and D00200. (The original contents of D00101 and D00100 are left unchanged.)



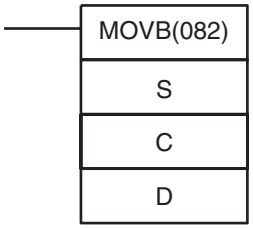


3-7-5 MOVE BIT: MOVB(082)

Purpose

Transfers the specified bit.

Ladder Symbol



S: Source word or data

C: Control word

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	MOVB(082)
	Executed Once for Upward Differentiation	@MOVB(082)
	Executed Once for Downward Differentiation	Not supported

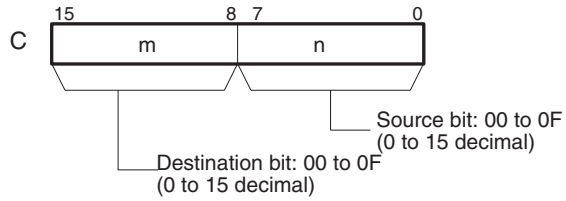
Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C: Control Word

The rightmost two digits of C indicate which bit of S is the source bit and the leftmost two digits of C indicate which bit of D is the destination bit.

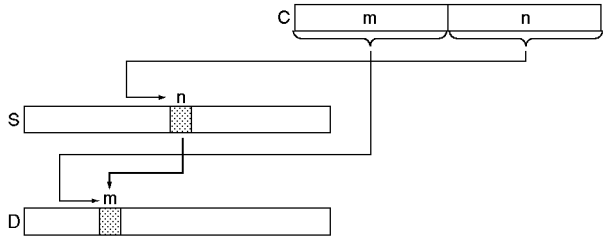


Operand Specifications

Area	S	C	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)	Specified values only	---
Data Resisters	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

MOVB(082) copies the specified bit (n) from S to the specified bit (m) in D. The other bits in the destination word are left unchanged.



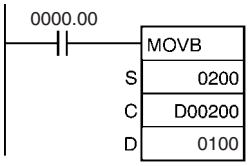
Note The same word can be specified for both S and D to copy a bit within a word.

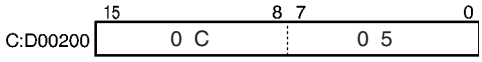
Flags

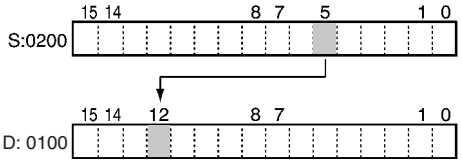
Name	Label	Operation
Error Flag	ER	ON if the rightmost and leftmost two digits of C are not within the specified range of 00 to 0F. OFF in all other cases.

Examples

When CIO 0000.00 is ON in the following example, the 5th bit of the source word (CIO 0200) is copied to the 12th bit of the destination word (CIO 0100) in accordance with the control word's value of 0C05.





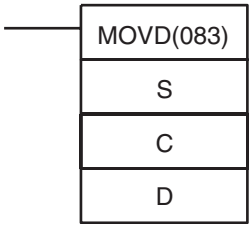


3-7-6 MOVE DIGIT: MOVD(083)

Purpose

Transfers the specified digit or digits. (Each digit is made up of 4 bits.)

Ladder Symbol



S: Source word or data

C: Control word

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	MOVD(083)
	Executed Once for Upward Differentiation	@MOVD(083)
	Executed Once for Downward Differentiation	Not supported

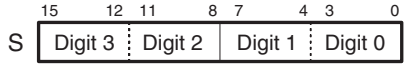
Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

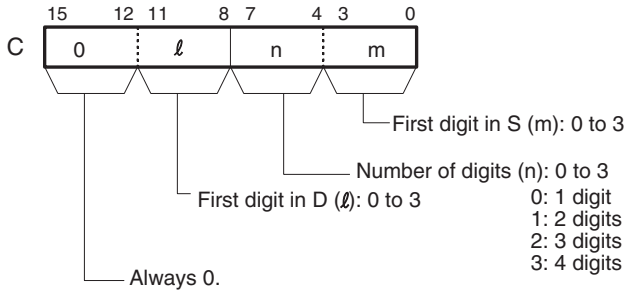
S: Source Word

The source digits are read from right to left, wrapping back to the rightmost digit (digit 0) if necessary.



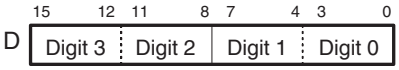
C: Control Word

The first three digits of C indicate the first source digit (m), the number of digits to transfer (n), and the first destination digit (ℓ), as shown in the following diagram.



D: Destination Word

The destination digits are written from right to left, wrapping back to the rightmost digit (digit 0) if necessary.



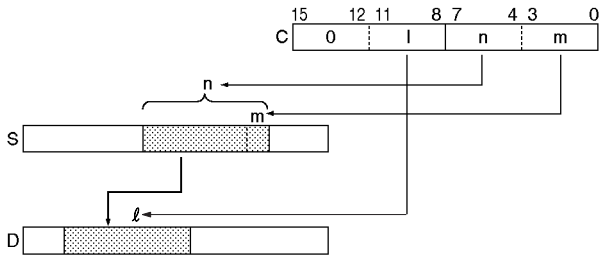
Operand Specifications

Area	S	C	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)	Specified values only	---
Data Resisters	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

MOVD(083) copies the content of n digits from S (beginning at digit m) to D (beginning at digit ℓ). Only the specified digits are changed; the rest are left unchanged.

If the number of digits being read or written exceeds the leftmost digit of S or D, MOVD(083) will wrap to the rightmost digit of the same word.



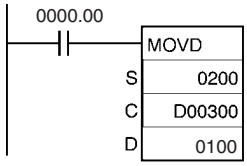
Flags

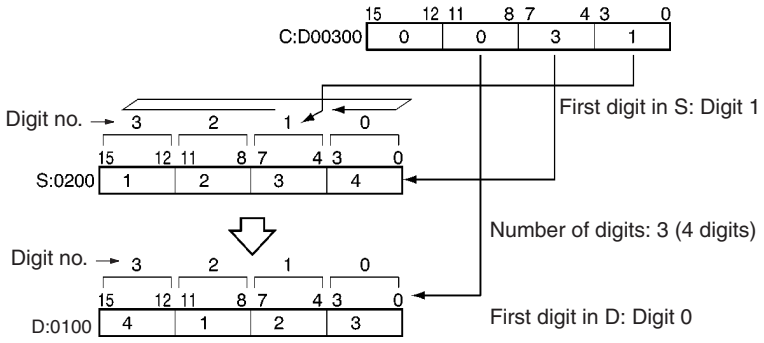
Name	Label	Operation
Error Flag	ER	ON if one of the first three digits of C is not within the specified range of 0 to 3. OFF in all other cases.

Examples

Four-digit Transfer

When CIO 0000.00 is ON in the following example, four digits of data are copied from CIO 0200 to CIO 0100. The transfer begins with the digit 1 of CIO 0200 and digit 0 or CIO 0100, in accordance with the control word's value of 0031.

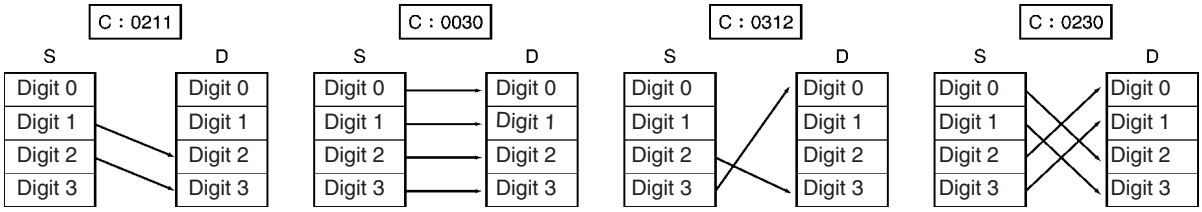




Note After reading the leftmost digit of S (digit 3), MOVD(083) wraps to the rightmost digit (digit 0).

Examples of C

The following diagram shows examples of data transfers for various values of C.

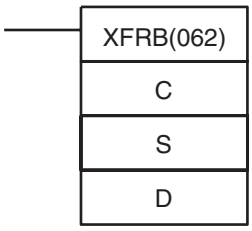


3-7-7 MULTIPLE BIT TRANSFER: XFRB(062)

Purpose

Transfers the specified number of consecutive bits.

Ladder Symbol



C: Control word

S: First source word

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	XFRB(062)
	Executed Once for Upward Differentiation	@XFRB(062)
	Executed Once for Downward Differentiation	Not supported

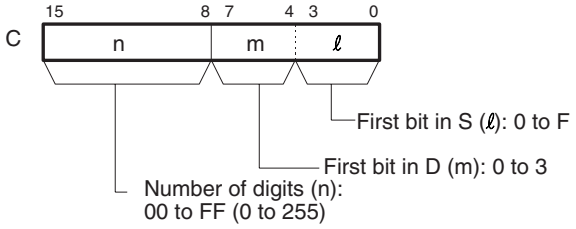
Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

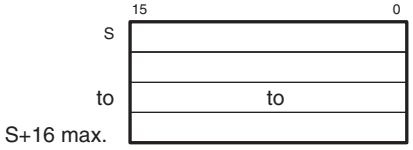
C: Control Word

The first three digits of C indicate the first source digit (m), the number of digits to transfer (n), and the first destination digit (ℓ), as shown in the following diagram.



S: First Source Word

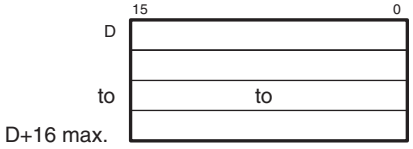
Specifies the first source word. Bits are read from right to left, continuing with consecutive words (up to S+16) when necessary.



Note The source words must be in the same data area.

D: First Destination Word

Specifies the first destination word. Bits are written from right to left, continuing with consecutive words (up to D+16) when necessary.



Note The destination words must be in the same data area.

Operand Specifications

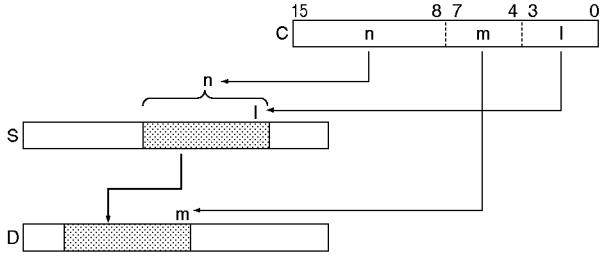
Area	C	S	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		

Area	C	S	D
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	Specified values only	---	---
Data Registers	DR0 to DR15	---	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to 5+(++) ,–(– –) IR0 to , –(– –) IR15		

Description

XFRB(062) transfers up to 255 consecutive bits from the source words (beginning with bit ℓ of S) to the destination words (beginning with bit m of D). Bits in the destination words that are not overwritten by the source bits are left unchanged.

The beginning bits and number of bits are specified in C, as shown in the following diagram.



It is possible for the source words and destination words to overlap. By transferring data overlapping several words, the data can be packed more efficiently in the data area. (This is particularly useful when handling position data for position control.)

Since the source words and destination words can overlap, XFRB(062) can be combined with ANDW(034) to shift m bits by n spaces.

Flags

Name	Label	Operation
Error Flag	ER	OFF

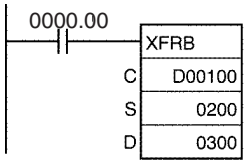
Precautions

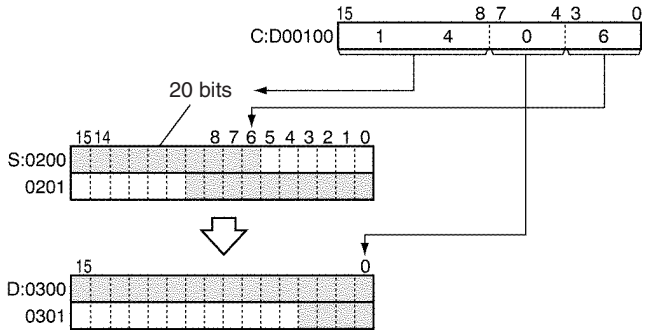
Up to 255 bits of data can be transferred per execution of XFRB(062).

Be sure that the source words and destination words do not exceed the end of the data area.

Examples

When CIO 0000.00 is ON in the following example, the 20 bits beginning with CIO 020006 are copied to the 20 bits beginning with CIO 030000.



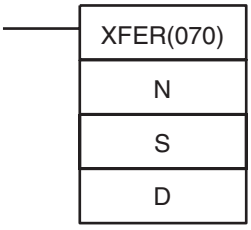


3-7-8 BLOCK TRANSFER: XFER(070)

Purpose

Transfers the specified number of consecutive words.

Ladder Symbol



N: Number of words

S: First source word

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	XFER(070)
	Executed Once for Upward Differentiation	@XFER(070)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

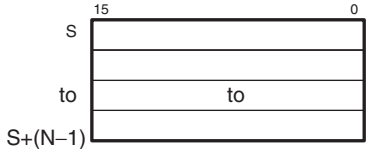
Operands

N: Number of Words

Specifies the number of words to be transferred. The possible range for N is 0000 to FFFF (0 to 65,535 decimal).

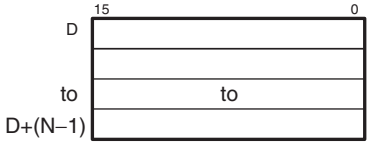
S: First Source Word

Specifies the first source word.



D: First Destination Word

Specifies the first destination word.

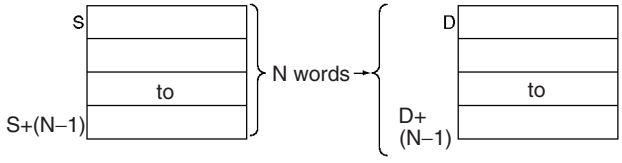


Operand Specifications

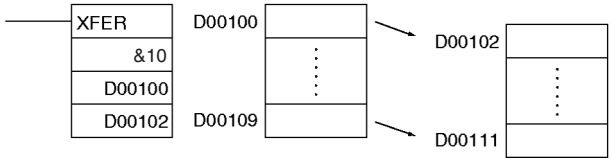
Area	N	S	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary) or &0 to &65535	---	---
Data Resisters	DR0 to DR15	---	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

XFER(070) copies N words beginning with S (S to S+(N–1)) to the N words beginning with D (D to D+(N–1)).



It is possible for the source words and destination words to overlap, so XFER(070) can perform word-shift operations.



Flags

Name	Label	Operation
Error Flag	ER	OFF

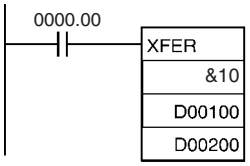
Precautions

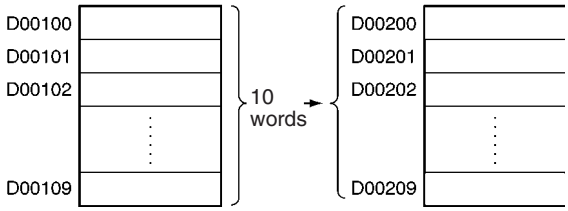
Be sure that the source words (S to S+N–1) and destination words (D to D+N–1) do not exceed the end of the data area.

Some time will be required to complete XFER(070) when a large number of words is being transferred. In this case, the XFER(070) transfer might not be completed if a power interruption occurs during execution of the instruction.

Example

When CIO 0000.00 is ON in the following example, the 10 words D00100 through D00109 are copied to D00200 through D00209.



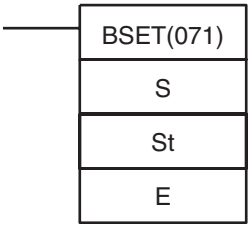


3-7-9 BLOCK SET: BSET(071)

Purpose

Copies the same word to a range of consecutive words.

Ladder Symbol



S: Source word

St: Starting word

E: End word

Variations

Variations		
	Executed Each Cycle for ON Condition	BSET(071)
	Executed Once for Upward Differentiation	@BSET(071)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S: Source Word

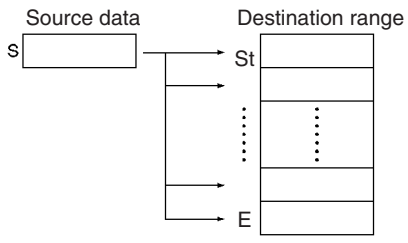
Specifies the source data or the word containing the source data.

St: Starting Word

Specifies the first word in the destination range.

E: End Word

Specifies the last word in the destination range.



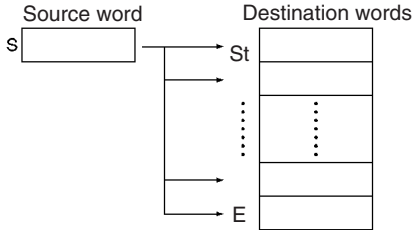
Note St and E must be in the same data area.

Operand Specifications

Area	S	St	E
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959	A448 to A959	
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)	---	
Data Resisters	DR0 to DR15	---	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

BSET(071) copies the same source word (S) to all of the destination words in the range St to E.



Flags

Name	Label	Operation
Error Flag	ER	ON if St is greater than E. OFF in all other cases.

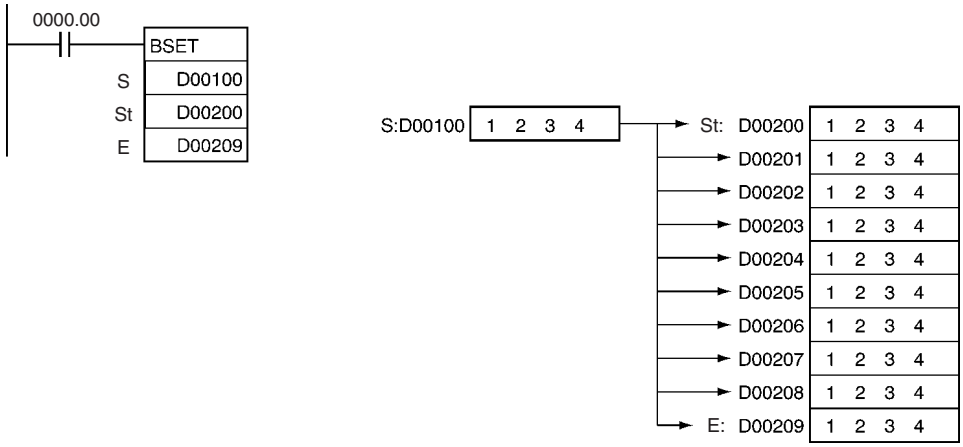
Precautions

Be sure that the starting word (St) and end word (E) are in the same data area and that St ≤ E.

Some time will be required to complete BSET(071) when the source data is being transferred to a large number of words. In this case, the BSET(071) transfer might not be completed if a power interruption occurs during execution of the instruction.

Example

When CIO 0000.00 is ON in the following example, the source data in D00100 is copied to D00200 through D00209.

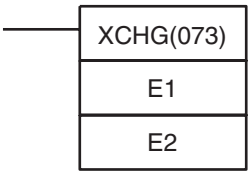


3-7-10 DATA EXCHANGE: XCHG(073)

Purpose

Exchanges the contents of the two specified words.

Ladder Symbol



E1: First exchange word

E2: Second exchange word

Variations

Variations		
	Executed Each Cycle for ON Condition	XCHG(073)
	Executed Once for Upward Differentiation	@XCHG(073)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

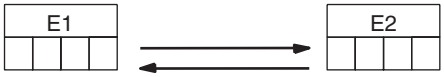
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	E1	E2
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T0255	
Counter Area	C0000 to C0255	
DM Area	D00000 to D32767	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Resisters	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

XCHG(073) exchanges the contents of E1 and E2.

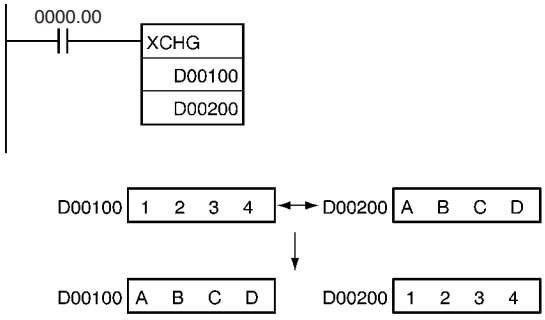


Flags

There are no flags affected by this instruction.

Example

When CIO 0000.00 is ON in the following example, the content of D00100 is exchanged with the content of D00200.

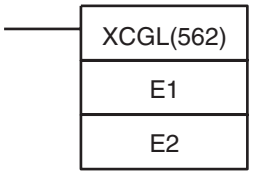


3-7-11 DOUBLE DATA EXCHANGE: XCGL(562)

Purpose

Exchanges the contents of a pair of consecutive words with another pair of consecutive words.

Ladder Symbol



E1: First exchange word

E2: Second exchange word

Variations

Variations		
	Executed Each Cycle for ON Condition	XCGL(562)
	Executed Once for Upward Differentiation	@XCGL(562)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

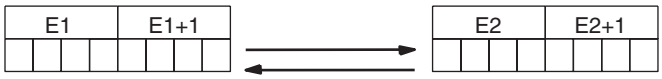
Operand Specifications

Area	E1	E2
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A448 to A958	
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	---
Data Registers	---	

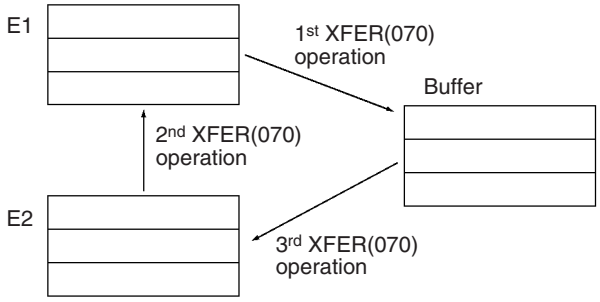
Area	E1	E2
Index Registers	IR0 to IR15	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047, IR0 to –2048 to +2047, IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –) IR0 to , –(– –) IR15	

Description

XCHG(073) exchanges the contents of E1+1 and E1 with the contents of E2+1 and E2.



To exchange 3 or more words, use XFER(070) to transfer the words to a third set of words (a buffer) as shown in the following diagram.

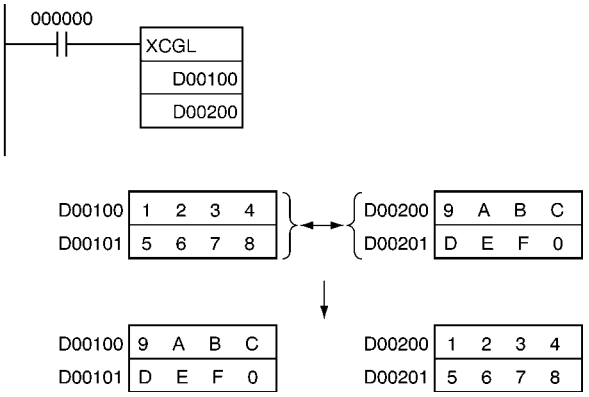


Flags

There are no flags affected by this instruction.

Example

When CIO 000000 is ON in the following example, the contents of D00100 and D00101 are exchanged with the contents of D00200 and D00201.

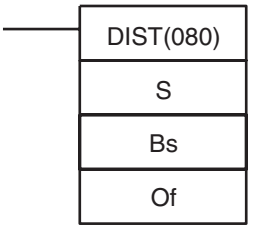


3-7-12 SINGLE WORD DISTRIBUTE: DIST(080)

Purpose

Transfers the source word to a destination word calculated by adding an offset value to the base address.

Ladder Symbol



S: Source word

Bs: Destination base address

Of: Offset

Variations

Variations		
	Executed Each Cycle for ON Condition	DIST(080)
	Executed Once for Upward Differentiation	@DIST(080)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

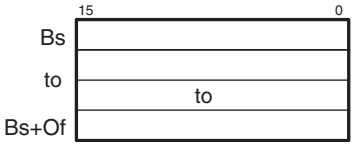
Operands

Bs: Destination Base Address

Specifies the destination base address. The offset is added to this address to calculate the destination word.

Of: Offset

This value is added to the base address to calculate the destination word. The offset can be any value from 0000 to FFFF (0 to 65,535 decimal), but Bs and Bs+Of must be in the same data area.

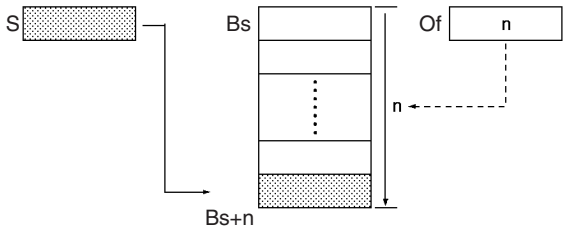


Operand Specifications

Area	S	Bs	Of
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959	A448 to A959	A000 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)	---	#0000 to #FFFF (binary) or &0 to &65535
Data Resisters	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

DIST(080) copies S to the destination word calculated by adding Of to Bs. The same DIST(080) instruction can be used to distribute the source word to various words in the data area by changing the value of Of.



Flags

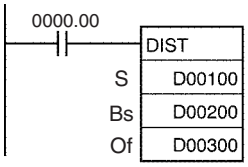
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the source data is 0000. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of the source data is 1. OFF in all other cases.

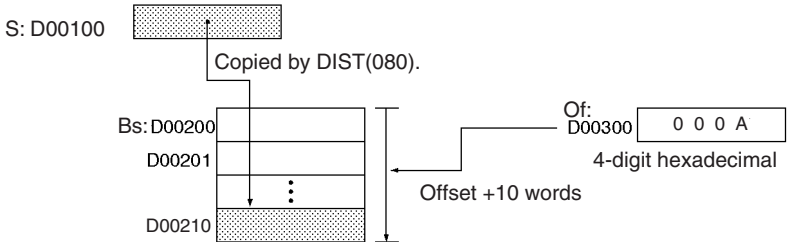
Precautions

Be sure that the offset does not exceed the end of the data area, i.e., Bs and Bs+Of are in the same data area.

Example

When CIO 0000.00 is ON in the following example, the contents of D00100 will be copied to D00210 (D00200 + 10) if the contents of D00300 is 10 (000A hexadecimal). The contents of D00100 can be copied to other words by changing the offset in D00300.



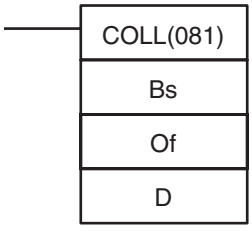


3-7-13 DATA COLLECT: COLL(081)

Purpose

Transfers the source word (calculated by adding an offset value to the base address) to the destination word.

Ladder Symbol



Bs: Source base address

Of: Offset

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	COLL(081)
	Executed Once for Upward Differentiation	@COLL(081)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

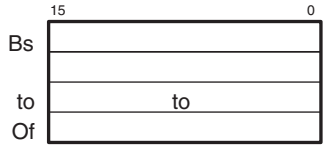
Operands

Bs: Source Base Address

Specifies the source base address. The offset is added to this address to calculate the source word.

Of: Offset

This value is added to the base address to calculate the source word. The offset can be any value from 0000 to FFFF (0 to 65,535 decimal), but Bs and Bs+Of must be in the same data area.

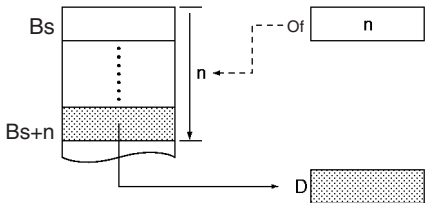


Operand Specifications

Area	Bs	Of	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---	#0000 to #FFFF (binary) or &0 to &65535	---
Data Resisters	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

COLL(081) copies the source word (calculated by adding Of to Bs) to the destination word. The same COLL(081) instruction can be used to collect data from various source words in the data area by changing the value of Of.



Flags

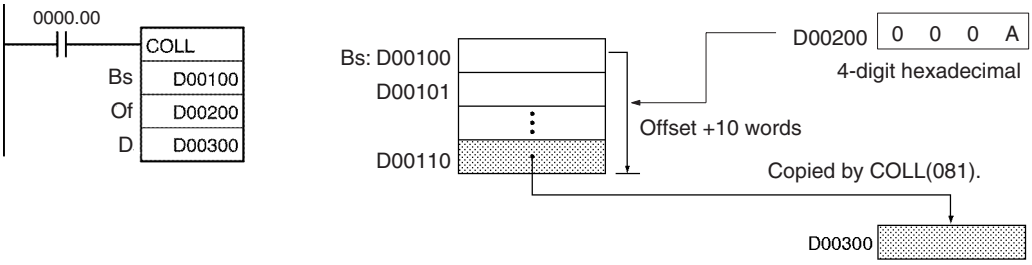
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the source data is 0000. OFF in all other cases.
Negative Flag	N	ON if the leftmost bit of the source data is 1. OFF in all other cases.

Precautions

Be sure that the offset does not exceed the end of the data area, i.e., Bs and Bs+Of are in the same data area.

Example

When CIO 0000.00 is ON in the following example, the contents of D00110 (D00100 + 10) will be copied to D00300 if the content of D00200 is 10 (000A hexadecimal). The contents of other words can be copied to D00300 by changing the offset in D00200.

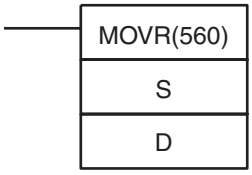


3-7-14 MOVE TO REGISTER: MOVR(560)

Purpose

Sets the PLC memory address of the specified word, bit, or timer/counter Completion Flag in the specified Index Register. (Use MOVRW(561) to set the PLC memory address of a timer/counter PV in an Index Register.)

Ladder Symbol



S: Source (desired word or bit)

D: Destination (Index Register)

Variations

Variations		
	Executed Each Cycle for ON Condition	MOVR(560)
	Executed Once for Upward Differentiation	@MOVR(560)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

D: Destination

The destination must be an Index Register (IR0 to IR15).

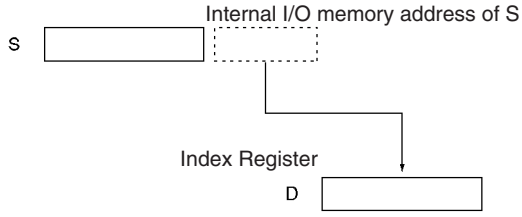
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6143	---
Work Area	W000 to W255	---
Auxiliary Bit Area	A448 to A959	---

Area	S	D
Timer Area	T0000 to T0255 (Completion Flag)	---
Counter Area	C0000 to C0255 (Completion Flag)	---
DM Area	D00000 to D32767	---
Indirect DM addresses in binary	---	
Indirect DM addresses in BCD	---	
Constants	---	
Data Registers	---	
Index Registers	---	IR0 to IR15
Indirect addressing using Index Registers	---	

Description

MOVR(560) finds the PLC memory address (absolute address) of S and writes that address in D (an Index Register).



If a timer or counter is specified in S, MOVR(560) will write the PLC memory address of the timer/counter Completion Flag in D. Use MOVRW(561) to write the PLC memory address of the timer/counter PV in D.

Flags

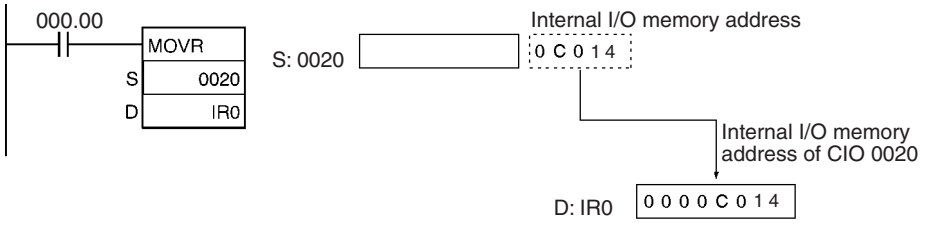
There are no flags affected by this instruction.

Precautions

MOVR(560) cannot set the PLC memory addresses of timer/counter PVs. Use MOVRW(561) to set the PLC memory addresses of timer/counter PVs. If MOVR(560) is executed for a timer/counter, the PLC memory address of the Completion Flag will be set in the index register.

Example

When CIO 0000.00 is ON in the following example, MOVR(560) writes the PLC memory address of CIO 0020 to IR0.

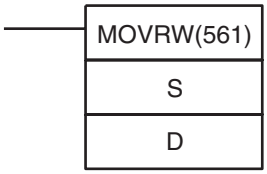


3-7-15 MOVE TIMER/COUNTER PV TO REGISTER: MOVRW(561)

Purpose

Sets the PLC memory address of the specified timer or counter's PV in the specified Index Register. (Use MOVR(560) to set the PLC memory address of a word, bit, or timer/counter Completion Flag in an Index Register.)

Ladder Symbol



S: Source (desired TC number)

D: Destination (Index Register)

Variations

Variations		
	Executed Each Cycle for ON Condition	MOVR(561)
	Executed Once for Upward Differentiation	@MOVR(561)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

D: Destination

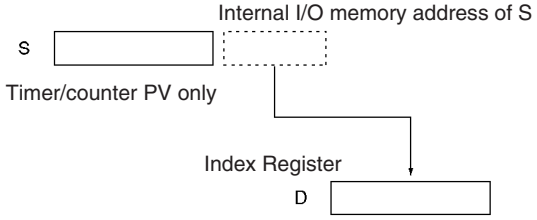
The destination must be an Index Register (IR0 to IR15).

Operand Specifications

Area	S	D
CIO Area	---	
Work Area	---	
Holding Bit Area	---	
Auxiliary Bit Area	---	
Timer Area	T0000 to T0255 (present value)	---
Counter Area	C0000 to C0255 (present value)	---
DM Area	---	
Indirect DM addresses in binary	---	
Indirect DM addresses in BCD	---	
Constants	---	
Data Registers	---	
Index Registers	---	IR0 to IR15
Indirect addressing using Index Registers	---	

Description

MOVRW(561) finds the PLC memory address for the PV of the timer or counter specified in S and writes that address in D (an Index Register).



MOVRW(561) will set the PLC memory address of the timer or counter's PV in D. Use MOVR(560) to set the PLC memory address of the timer or counter Completion Flag.

Flags

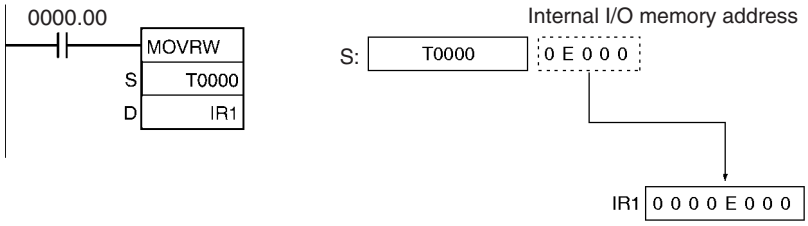
There are no flags affected by this instruction.

Precautions

MOVRW(561) cannot set the PLC memory addresses of data area words, bits, or timer/counter Completion Flags. Use MOVR(560) to set these PLC memory addresses.

Example

When CIO 0000.00 is ON in the following example, MOVRW(561) writes the PLC memory address for the PV of timer T0000 to IR1.



3-8 Data Shift Instructions

This section describes instructions used to shift data within or between words, but in differing amounts and directions.

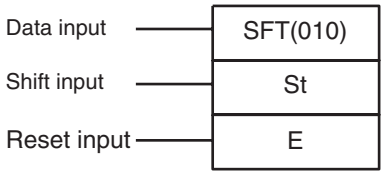
Instruction	Mnemonic	Function code	Page
SHIFT REGISTER	SFT	010	225
REVERSIBLE SHIFT REGISTER	SFTR	084	227
ASYNCHRONOUS SHIFT REGISTER	ASFT	017	230
WORD SHIFT	WSFT	016	232
ARITHMETIC SHIFT LEFT	ASL	025	233
DOUBLE SHIFT LEFT	ASLL	570	235
ARITHMETIC SHIFT RIGHT	ASR	026	236
DOUBLE SHIFT RIGHT	ASRL	571	238
ROTATE LEFT	ROL	027	239
DOUBLE ROTATE LEFT	ROLL	572	241
ROTATE LEFT WITHOUT CARRY	RLNC	574	245
DOUBLE ROTATE LEFT WITHOUT CARRY	RLNL	576	247
ROTATE RIGHT	ROR	028	242
DOUBLE ROTATE RIGHT	RORL	573	244
ROTATE RIGHT WITHOUT CARRY	RRNC	575	248
DOUBLE ROTATE RIGHT WITHOUT CARRY	RRNL	577	250
ONE DIGIT SHIFT LEFT	SLD	074	251
ONE DIGIT SHIFT RIGHT	SRD	075	253
SHIFT N-BITS LEFT	NASL	580	254
DOUBLE SHIFT N-BITS LEFT	NSLL	582	256
SHIFT N-BITS RIGHT	NASR	581	259
DOUBLE SHIFT N-BITS RIGHT	NSRL	583	261

3-8-1 SHIFT REGISTER: SFT(010)

Purpose

Operates a shift register.

Ladder Symbol



St: Starting word

E: End word

Variations

Variations		
	Executed Each Cycle for ON Condition	SFT(010)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	OK

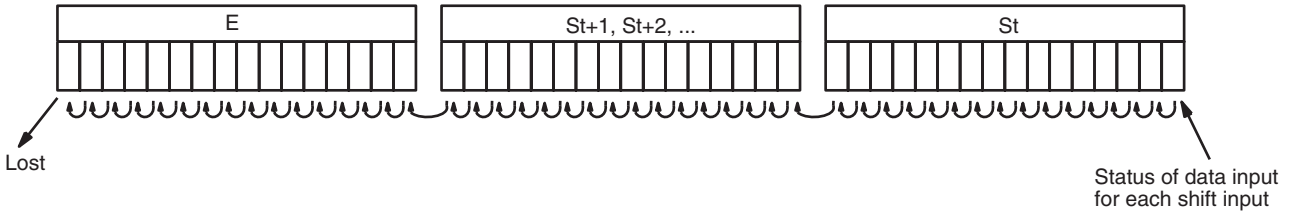
Note St and E must be in the same data area.

Operand Specifications

Area	St	E
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	
Auxiliary Bit Area	A448 to A959	
Timer Area	---	
Counter Area	---	
DM Area	---	
Indirect DM addresses in binary	---	
Indirect DM addresses in BCD	---	
Constants	---	
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15	

Description

When the execution condition on the shift input changes from OFF to ON, all the data from St to E is shifted to the left by one bit (from the rightmost bit to the leftmost bit), and the ON/OFF status of the data input is placed in the rightmost bit.



Flags

Name	Label	Operation
Error Flag	ER	ON if the indirect IR address for St and E is not in the CIO, AR, or WR data areas. OFF in all other cases.

Precautions

The bit data shifted out of the shift register is discarded.

When the reset input turns ON, all bits in the shift register from the rightmost designated word (St) to the leftmost designated word (E) will be reset (i.e., set to 0). The reset input takes priority over other inputs.

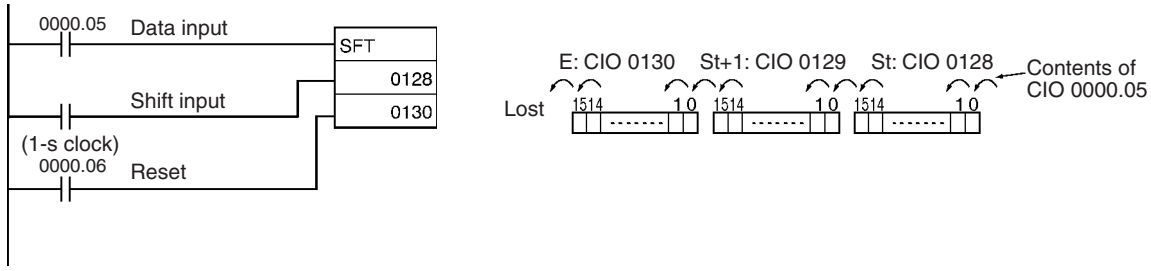
St must be less than or equal to E, but even when St is set to greater than E an error will not occur and one word of data in St will be shifted.

When St and E are designated indirectly using index registers and the actual addresses in I/O memory are not within memory areas for data, an error will occur and the Error Flag will turn ON.

Examples

Shift Register Exceeding 16 Bits

The following example shows a 48-bit shift register using words CIO 0128 to CIO 0130. A 1-s clock pulse is used so that the contents of CIO 0000.05 is input and shifted into a 3-word register between CIO 0128.00 and CIO 0130.15 every second.

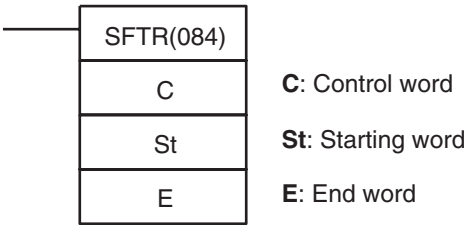


3-8-2 REVERSIBLE SHIFT REGISTER: SFTR(084)

Purpose

Creates a shift register that shifts data to either the right or the left.

Ladder Symbol



Variations

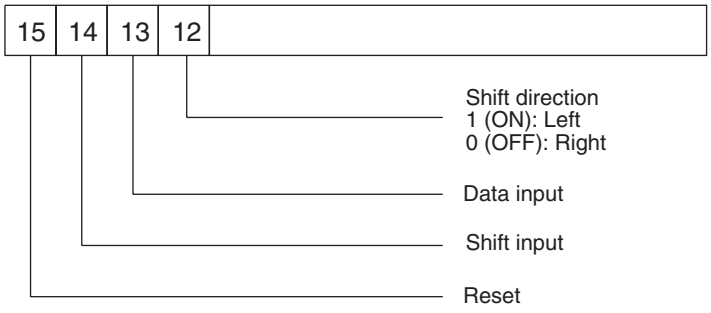
Variations		
	Executed Each Cycle for ON Condition	SFTR(084)
	Executed Once for Upward Differentiation	@SFTR(084)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C: Control Word



Note St and E must be in the same data area.

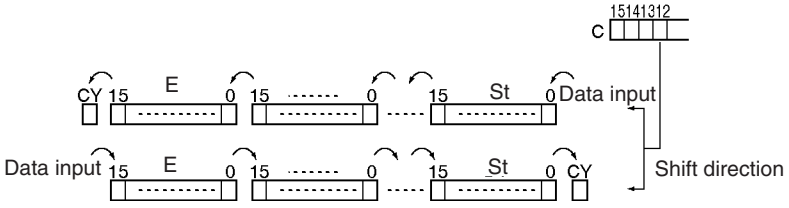
Operand Specifications

Area	C	St	E
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959	A448 to A959	
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		

Area	C	St	E
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---		
Data Resisters	DR0 to DR15	---	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

When the execution condition of the shift input bit (bit 14 of C) changes to ON, all the data from St to E is moved in the designated shift direction (designated by bit 12 of C) by 1 bit, and the ON/OFF status of the data input is placed in the rightmost or leftmost bit. The bit data shifted out of the shift register is placed in the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	ON when St is greater than E. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into it. OFF when 0 is shifted into it. OFF when reset is set to 1.

Precautions

The above shift operations are applicable when the reset bit (bit 15 of C) is set to OFF.

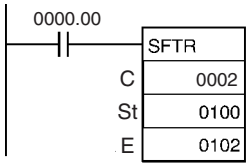
When reset (bit 15 of C) turns ON, all bits in the shift register, from St to E will be reset (i.e., set to 0).

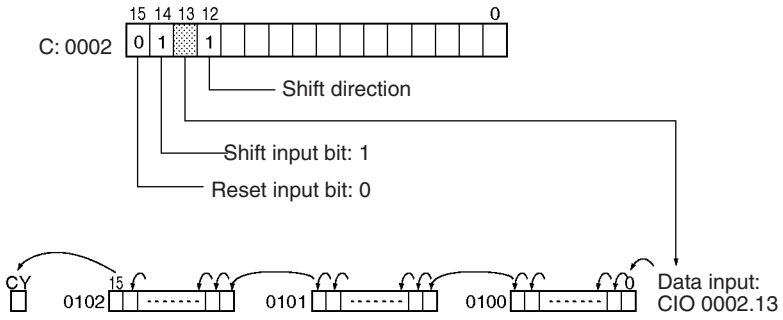
When St is greater than E, an error will be generated and the Error Flag will turn ON.

Examples

Shifting Data

If shift input CIO 0002.14 goes ON when CIO 0000.00 is ON, and the reset bit CIO 0002.15 is OFF, words CIO 0100 through CIO 0102 will shift one bit in the direction designated by CIO 0002.12 (e.g., 1: Left) and the contents of input bit CIO 0002.13 will be input into the rightmost bit, CIO 0100.00. The contents of CIO 0102.15 will be shifted to the Carry Flag (CY).





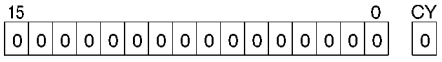
Resetting Data

If CIO 0002.14 is ON when CIO 0000.00 is ON, and the reset bit, CIO 0002.15, is ON, words CIO 0100 through CIO 0102 and the Carry Flag will be reset to OFF.

Controlling Data

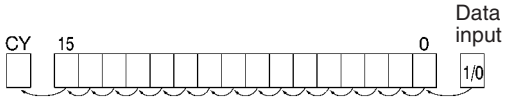
Resetting Data

All bits from St to E and the Carry Flag are set to 0 and no other data can be received when the reset input bit (bit 15 of C) is ON.



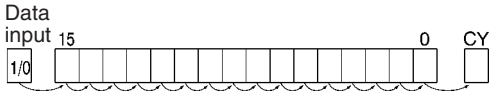
Shifting Data Left (from Rightmost to Leftmost Bit)

When the shift input bit (bit 14 of C) is ON, the contents of the input bit (bit 13 of C) is input to bit 00 of the starting word, and each bit thereafter is shifted one bit to the left. The status of bit 15 of the end word is shifted to the Carry Flag.



Shifting Data Right (from Leftmost to Rightmost Bit

When the shift input bit (bit 14 of C) is ON, the contents of the input bit (bit 13 of C) is input to bit 15 on the end word, and each bit thereafter is shifted one bit to the right. The status of bit 00 of the starting word is shifted to the Carry Flag.

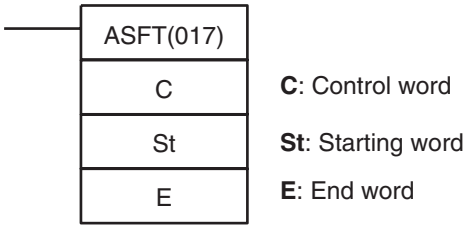


3-8-3 ASYNCHRONOUS SHIFT REGISTER: ASFT(017)

Purpose

Shifts all non-zero word data within the specified word range either towards St or toward E, replacing 0000 hex word data.

Ladder Symbol



Variations

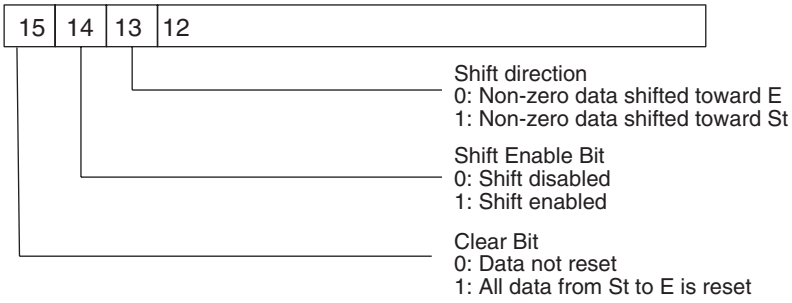
Variations		
	Executed Each Cycle for ON Condition	ASFT(017)
	Executed Once for Upward Differentiation	@ASFT(017)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C: Control Word



Note St and E must be in the same data area.

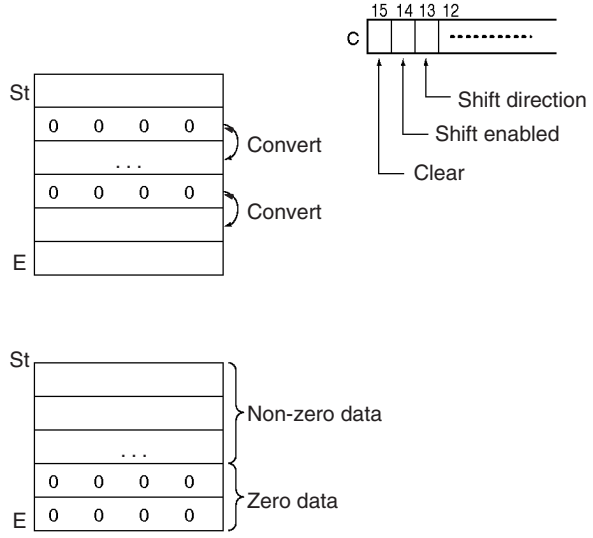
Operand Specifications

Area	C	St	E
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959	A448 to A959	
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---		
Data Resisters	DR0 to DR15	---	

Area	C	St	E
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

When the Shift Enable Bit (bit 14 of C) is ON, all of the words with non-zero content within the range of words between St and E will be shifted one word in the direction determined by the Shift Direction Bit (bit 13 of C) whenever the word in the shift direction contains all zeros. If ASFT(017) is repeated sufficient times, all all-zero words will be replaced by non-zero words. This will result in all the data between St and E being divided into zero and non-zero data.



Flags

Name	Label	Operation
Error Flag	ER	ON when St is greater than E. OFF in all other cases.

Precautions

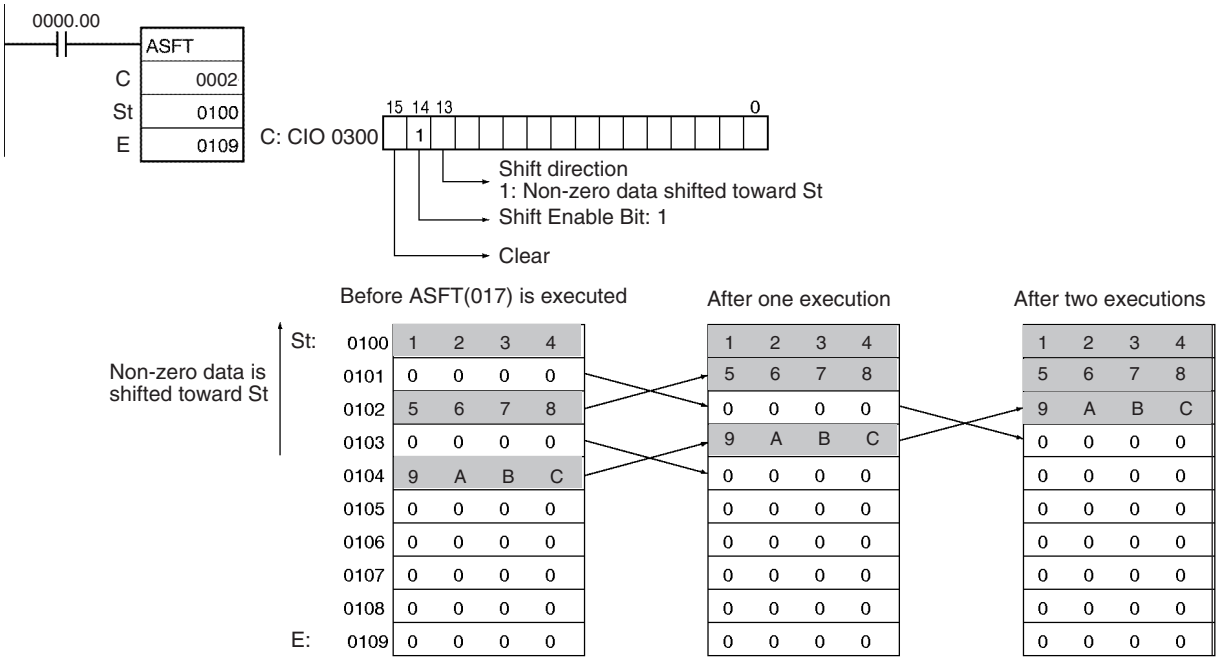
When the Clear Bit (bit 15 of C) goes ON, all bits in the shift register, from St to E, will be reset (i.e., set to 0). The Clear Bit has priority over the Shift Enable Bit (bit 14 of C).

When St is greater than E, an error will be generated and the Error Flag will turn ON.

Examples

Shifting Data:

If the Shift Enable Bit, CIO 0002.14, goes ON when CIO 0000.00 is ON, all words with non-zero data content from CIO 0100 through CIO 0109 will be shifted in the direction designated by the Shift Direction Bit, CIO 0002.13 (e.g., 1: shifted toward St) if the word to the left of the non-zero data is all zeros.

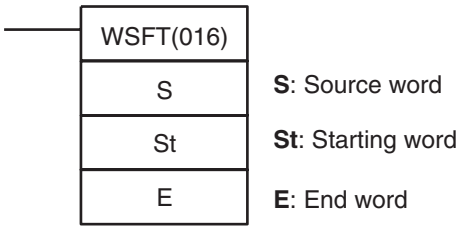


3-8-4 WORD SHIFT: WSFT(016)

Purpose

Shifts data between St and E in word units.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	WSFT(016)
	Executed Once for Upward Differentiation	@WSFT(016)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Note St and E must be in the same data area.

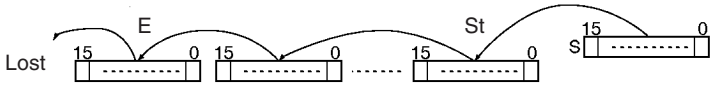
Operand Specifications

Area	S	St	E
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959	A448 to A959	
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		

Area	S	St	E
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)	---	
Data Resisters	DR0 to DR15	---	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

WSFT(016) shifts data from St to E in word units and the data from the source word S is placed into St. The contents of E is lost.



Flags

Name	Label	Operation
Error Flag	ER	ON when St is greater than E. OFF in all other cases.

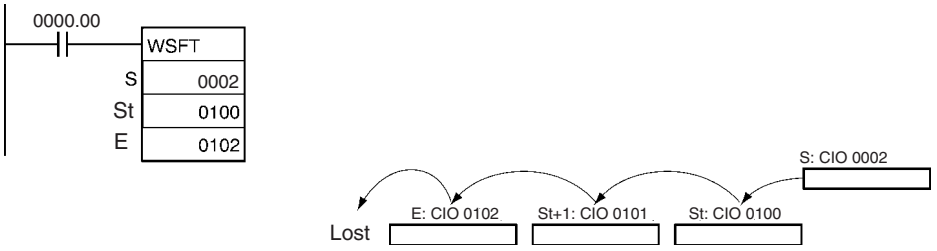
Precautions

When St is greater than E, an error will be generated and the Error Flag will turn ON.

Note When large amounts of data are shifted, the instruction execution time is quite long. Be sure that the power is not cut while WSFT(016) is being executed, causing the shift operation to stop halfway through.

Examples

When CIO 0000.00 is ON, data from CIO 0100 through CIO 0102 will be shifted one word toward E. The contents of CIO 0002 will be stored in CIO 0100 and the contents of CIO 0102 will be lost.

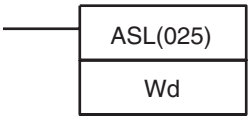


3-8-5 ARITHMETIC SHIFT LEFT: ASL(025)

Purpose

Shifts the contents of Wd one bit to the left.

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	ASL(025)
	Executed Once for Upward Differentiation	@ASL(025)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

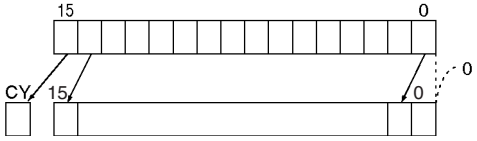
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W255
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
DM Area	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

ASL(025) shifts the contents of Wd one bit to the left (from rightmost bit to leftmost bit). “0” is placed in the rightmost bit and the data from the leftmost bit is shifted into the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0000. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as the result of the shift. OFF in all other cases.

Precautions

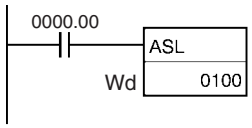
When ASL(025) is executed, the Error Flag will turn OFF.

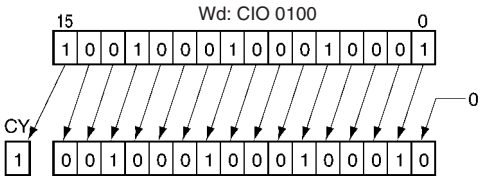
If the contents of Wd is 0000 as the result of the shift, the Equals Flag will turn ON.

If the contents of the leftmost bit in Wd is 1 as the result of the shift, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON, CIO 0100 will be shifted one bit to the left. "0" will be placed in CIO 0100.00 and the contents of CIO 0100.15 will be shifted to the Carry Flag (CY).



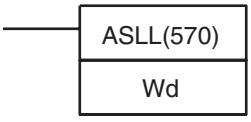


3-8-6 DOUBLE SHIFT LEFT: ASLL(570)

Purpose

Shifts the contents of Wd and Wd +1 one bit to the left.

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	ASLL(570)
	Executed Once for Upward Differentiation	@ASLL(570)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

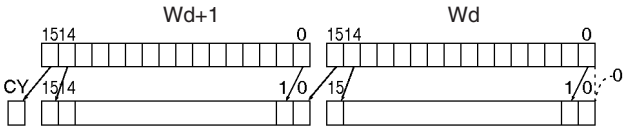
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W254
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T0254
Counter Area	C0000 to C0254
DM Area	D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

ASLL(570) shifts the contents of Wd and Wd +1 one bit to the left (from rightmost bit to leftmost bit). "0" is placed in the rightmost bit of Wd and the contents of the leftmost bit in Wd +1 is shifted into the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0000 0000. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as the result of the shift. OFF in all other cases.

Precautions

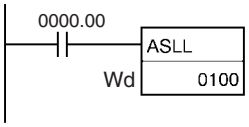
When ASLL(570) is executed, the Error Flag will turn OFF.

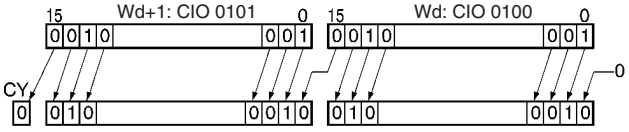
If the contents of Wd and Wd +1 is 0000 0000 as the result of the shift, the Equals Flag will turn ON.

If the contents of the leftmost bit in Wd +1 is 1 as the result of the shift, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON, word CIO 0100 and CIO 0101 will shift one bit to the left. "0" is placed into CIO 0100.00 and the contents of CIO 0100.15 will be shifted to the Carry Flag (CY).



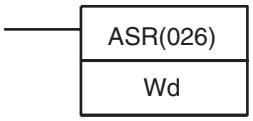


3-8-7 ARITHMETIC SHIFT RIGHT: ASR(026)

Purpose

Shifts the contents of Wd one bit to the right.

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	ASR(026)
	Executed Once for Upward Differentiation	@ASR(026)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

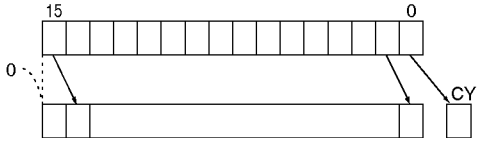
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W255
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
DM Area	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

ASR(026) shifts the contents of Wd one bit to the right (from leftmost bit to rightmost bit). “0” will be placed in the leftmost bit and the contents of the rightmost bit will be shifted into the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0000. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	OFF

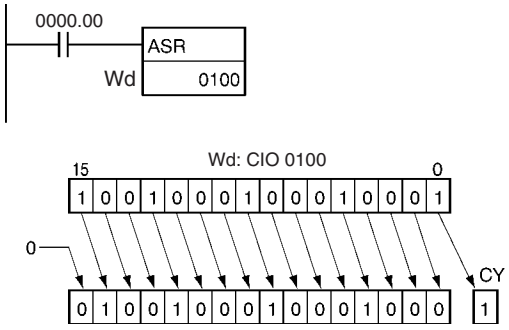
Precautions

When ASR(026) is executed, the Error Flag and the Negative Flag will turn OFF.

If the contents of Wd is 0000 as the result of the shift, the Equals Flag will turn ON.

Examples

When CIO 0000.00 is ON, word CIO 0100 will shift one bit to the right. “0” will be placed in CIO 0100.15 and the contents of CIO 0100.00 will be shifted to the Carry Flag (CY).

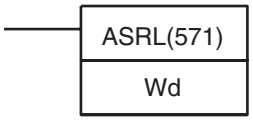


3-8-8 DOUBLE SHIFT RIGHT: ASRL(571)

Purpose

Shifts the contents of Wd and Wd +1 one bit to the right.

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	ASRL(571)
	Executed Once for Upward Differentiation	@ASRL(571)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

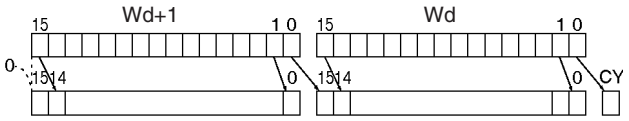
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W254
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T0254
Counter Area	C0000 to C0254
DM Area	D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

ASRL(571) shifts the contents of Wd and Wd +1 one bit to the right (from leftmost bit to rightmost bit). “0” will be placed in the leftmost bit of Wd +1 and the contents of the rightmost bit of Wd will be shifted into the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0000 0000. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	OFF

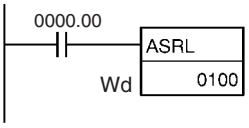
Precautions

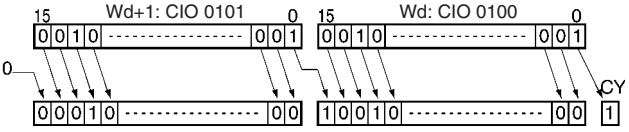
When ASRL (571) is executed, the Error Flag and the Negative Flag will turn OFF.

If the contents of Wd and Wd +1 is 0000 0000 as the result of the shift, the Equals Flag will turn ON.

Examples

When CIO 0000.00 is ON, word CIO 0100 and CIO 0101 will shift one bit to the right. “0” will be placed into CIO 0101.15 and the contents of CIO 0100.00 will be shifted to the Carry Flag (CY).



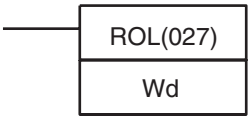


3-8-9 ROTATE LEFT: ROL(027)

Purpose

Shifts all Wd bits one bit to the left including the Carry Flag (CY).

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	ROL(027)
	Executed Once for Upward Differentiation	@ROL(027)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

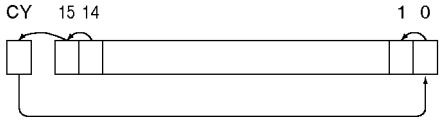
Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W255

Area	Wd
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
DM Area	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

ROL(027) shifts all bits of Wd including the Carry Flag (CY) to the left (from rightmost bit to leftmost bit).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0000. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as the result of the shift. OFF in all other cases.

Precautions

When ROL(027) is executed, the Error Flag will turn OFF.

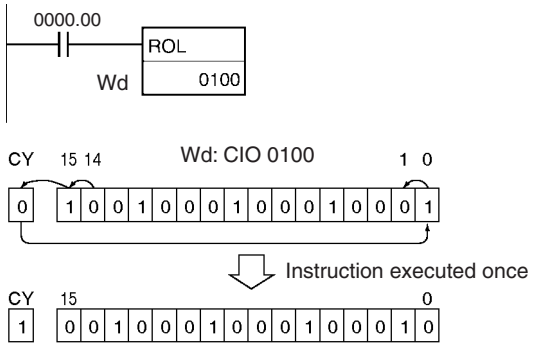
If the contents of Wd is 0000 as the result of the shift, the Equals Flag will turn ON.

If the contents of the leftmost bit in Wd is 1 as the result of the shift, the Negative Flag will turn ON.

Note It is possible to set the Carry Flag contents to 1 or 0 immediately before executing this instruction, by using the Set Carry (STC(040)) or Clear Carry (CLC(041)) instructions.

Examples

When CIO 0000.00 is ON, word CIO 0100 and the Carry Flag (CY) will shift one bit to the left. The contents of CIO 0100.15 will be shifted to the Carry Flag (CY) and the Carry Flag contents will be shifted to CIO 0100.00.

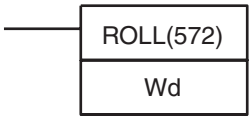


3-8-10 DOUBLE ROTATE LEFT: ROLL(572)

Purpose

Shifts all Wd and Wd +1 bits one bit to the left including the Carry Flag (CY).

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	ROLL(572)
	Executed Once for Upward Differentiation	@ROLL(572)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

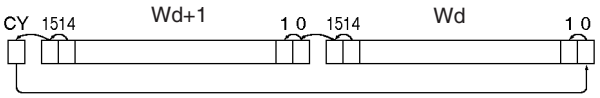
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W254
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T0254
Counter Area	C0000 to C0254
DM Area	D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

ROLL(572) shifts all bits of Wd and Wd +1 including the Carry Flag (CY) to the left (from rightmost bit to leftmost bit).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0000 0000. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as the result of the shift. OFF in all other cases.

Precautions

When ROLL(572) is executed, the Error Flag will turn OFF.

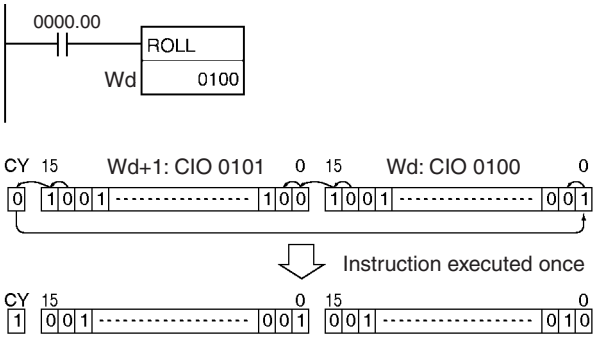
If the contents of Wd and Wd +1 is 0000 0000 as the result of the shift, the Equals Flag will turn ON.

If the contents of the leftmost bit in Wd + 1 is 1 as the result of the shift, the Negative Flag will turn ON.

Note It is possible to set the Carry Flag contents to 1 or 0 immediately before executing this instruction, by using the Set Carry (STC(040)) or Clear Carry (CLC(041)) instructions.

Examples

When CIO 0000.00 is ON, word CIO 0100, CIO 0101 and the Carry Flag (CY) will shift one bit to the left. The contents of CIO 0101.15 will be shifted to the Carry Flag (CY) and the Carry Flag contents will be shifted to CIO 0100.00.

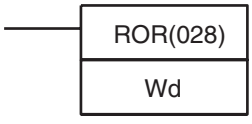


3-8-11 ROTATE RIGHT: ROR(028)

Purpose

Shifts all Wd bits one bit to the right including the Carry Flag (CY).

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	ROR(028)
	Executed Once for Upward Differentiation	@ROR(028)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

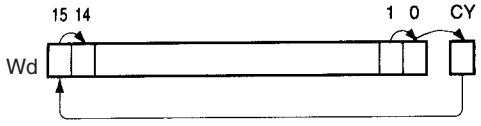
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W255
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
DM Area	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

ROR(028) shifts all bits of Wd including the Carry Flag (CY) to the right (from leftmost bit to rightmost bit).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0000. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as the result of the shift. OFF in all other cases.

Precautions

When ROR(028) is executed, the Error Flag will turn OFF.

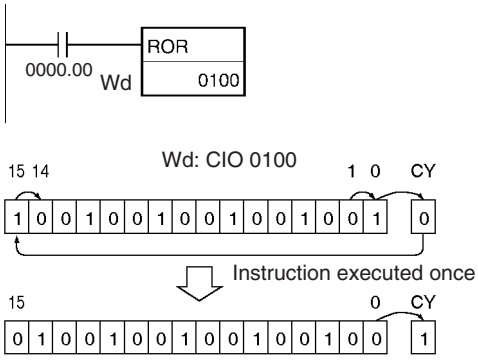
If the contents of Wd is 0000 as the result of the shift, the Equals Flag will turn ON.

If the contents of the leftmost bit in Wd is 1 as the result of the shift, the Negative Flag will turn ON.

Note It is possible to set the Carry Flag contents to 1 or 0 immediately before executing this instruction, by using the Set Carry (STC(040)) or Clear Carry (CLC(041)) instructions.

Examples

When CIO 0000.00 is ON, word CIO 0100 and the Carry Flag (CY) will shift one bit to the right. The contents of CIO 0100.00 will be shifted to the Carry Flag (CY) and the Carry Flag contents will be shifted to CIO 0100.15.

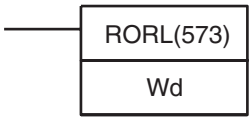


3-8-12 DOUBLE ROTATE RIGHT: RORL(573)

Purpose

Shifts all Wd and Wd +1 bits one bit to the right including the Carry Flag (CY).

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	RORL(573)
	Executed Once for Upward Differentiation	@RORL(573)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

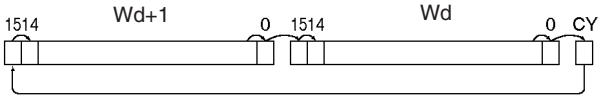
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W254
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T0254
Counter Area	C0000 to C0254
DM Area	D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

RORL(573) shifts all bits of Wd and Wd +1 including the Carry Flag (CY) to the right (from leftmost bit to rightmost bit).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0000 0000. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as the result of the shift. OFF in all other cases.

Precautions

When RORL(573) is executed, the Error Flag will turn OFF.

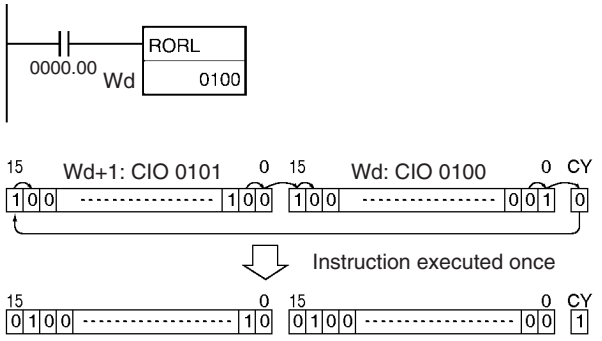
If the contents of Wd and Wd +1 is 0000 0000 as the result of the shift, the Equals Flag will turn ON.

If the contents of the leftmost bit in Wd + 1 is 1 as the result of the shift, the Negative Flag will turn ON.

Note It is possible to set the Carry Flag contents to 1 or 0 immediately before executing this instruction, by using the Set Carry (STC(040)) or Clear Carry (CLC(041)) instructions.

Examples

When CIO 0000.00 is ON, word CIO 0100, CIO 0101 and the Carry Flag (CY) will shift one bit to the right. The contents of CIO 0100.00 will be shifted to the Carry Flag (CY) and the Carry Flag contents will be shifted to CIO 0101.15.

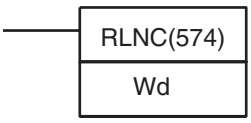


3-8-13 ROTATE LEFT WITHOUT CARRY: RLNC(574)

Purpose

Shifts all Wd bits one bit to the left not including the Carry Flag (CY).

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	RLNC(574)
	Executed Once for Upward Differentiation	@RLNC(574)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

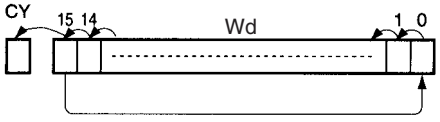
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W255
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
DM Area	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

RLNC(574) shifts all bits of Wd to the left (from rightmost bit to leftmost bit). The contents of the leftmost bit of Wd shifts to the rightmost bit and to the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0000. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as the result of the shift. OFF in all other cases.

Precautions

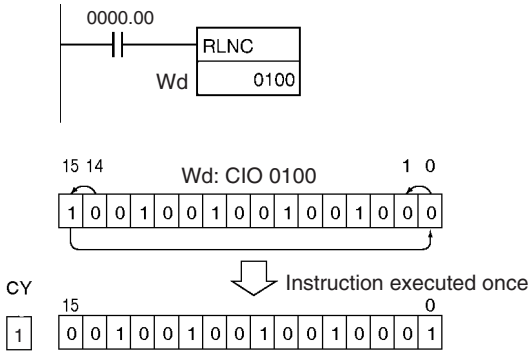
When RLNC(574) is executed, the Error Flag will turn OFF.

If the contents of Wd is 0000 as the result of the shift, the Equals Flag will turn ON.

If the contents of the leftmost bit in Wd is 1 as the result of the shift, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON, word CIO 0100 will shift one bit to the left (excluding the Carry Flag (CY)). The contents of CIO 0100.15 will be shifted to CIO 0100.00.

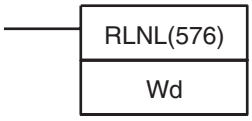


3-8-14 DOUBLE ROTATE LEFT WITHOUT CARRY: RLNL(576)

Purpose

Shifts all Wd and Wd +1 bits one bit to the left not including the Carry Flag (CY).

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	RLNL(576)
	Executed Once for Upward Differentiation	@RLNL(576)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

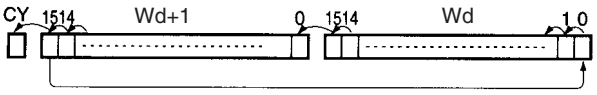
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W254
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T0254
Counter Area	C0000 to C0254
DM Area	D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

RLNL(576) shifts all bits of Wd and Wd +1 to the left (from rightmost bit to leftmost bit). The contents of the leftmost bit of Wd +1 is shifted to the rightmost bit of Wd, and to the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0000 0000. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as the result of the shift. OFF in all other cases.

Precautions

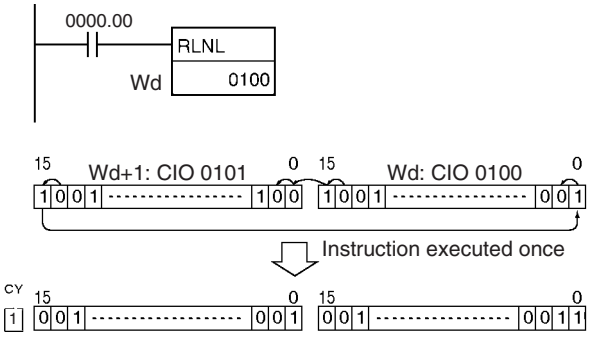
When RLNL(576) is executed, the Error Flag will turn OFF.

If the contents of Wd and Wd +1 is 0000 0000 as the result of the shift, the Equals Flag will turn ON.

If the contents of the leftmost bit in Wd + 1 is 1 as the result of the shift, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON, word CIO 0100 and CIO 0101 will shift one bit to the left (excluding the Carry Flag (CY)). The contents of CIO 0101.15 will be shifted to CIO 0100.00.

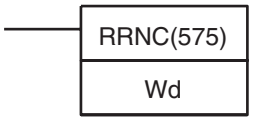


3-8-15 ROTATE RIGHT WITHOUT CARRY: RRNC(575)

Purpose

Shifts all Wd bits one bit to the right not including the Carry Flag (CY). The contents of the rightmost bit of Wd shifts to the leftmost bit and to the Carry Flag (CY).

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	RRNC(575)
	Executed Once for Upward Differentiation	@RRNC(575)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

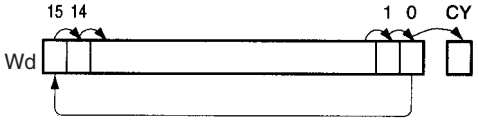
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W255
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
DM Area	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

RRNC(575) shifts all bits of Wd to the right (from leftmost bit to rightmost bit) not including the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0000. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as the result of the shift. OFF in all other cases.

Precautions

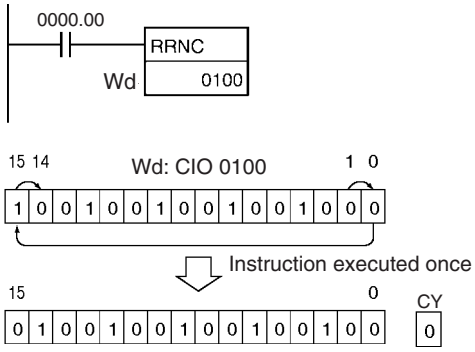
When RRNC(575) is executed, the Error Flag will turn OFF.

If the contents of Wd is 0000 as the result of the shift, the Equals Flag will turn ON.

If the contents of the leftmost bit in Wd is 1 as the result of the shift, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON, word CIO 0100 will shift one bit to the right (excluding the Carry Flag (CY)). The contents of CIO 0100.00 will be shifted to CIO 0100.15.

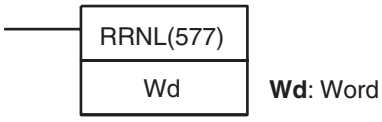


3-8-16 DOUBLE ROTATE RIGHT WITHOUT CARRY: RRNL(577)

Purpose

Shifts all Wd and Wd +1 bits one bit to the right not including the Carry Flag (CY). The contents of the rightmost bit of Wd is shifted to the leftmost bit of Wd +1, and to the Carry Flag (CY).

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	RRNL(577)
	Executed Once for Upward Differentiation	@RRNL(577)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

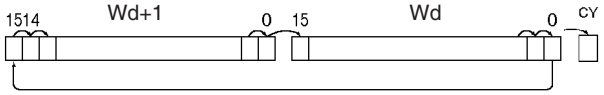
Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W254
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T0254
Counter Area	C0000 to C0254
DM Area	D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	---

Area	Wd
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

RRNL(577) shifts all bits of Wd and Wd +1 to the right (from leftmost bit to rightmost bit) not including the Carry Flag (CY).



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the shift result is 0000 0000. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as the result of the shift. OFF in all other cases.

Precautions

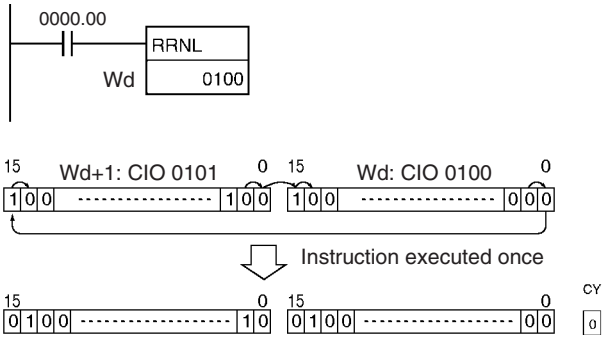
When RRNL(577) is executed, the Error Flag will turn OFF.

If the contents of Wd and Wd +1 is 0000 0000 as the result of the shift, the Equals Flag will turn ON.

If the contents of the leftmost bit in Wd + 1 is 1 as the result of the shift, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON, words CIO 0100 and CIO 0101 will shift one bit to the right, (excluding the Carry Flag (CY)). The contents of CIO 0100.00 will be shifted to CIO 0101.15.

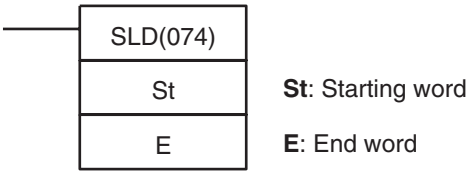


3-8-17 ONE DIGIT SHIFT LEFT: SLD(074)

Purpose

Shifts data by one digit (4 bits) to the left.

Ladder Symbol



St: Starting word

E: End word

Variations

Variations		
	Executed Each Cycle for ON Condition	SLD(074)
	Executed Once for Upward Differentiation	@SLD(074)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

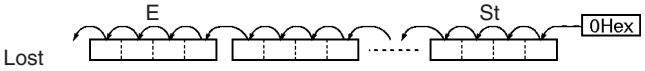
Note St and E must be in the same data area.

Operand Specifications

Area	St	E
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T0255	
Counter Area	C0000 to C0255	
DM Area	D00000 to D32767	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

SLD(074) shifts data between St and E by one digit (4 bits) to the left. “0” is placed in the rightmost digit (bits 3 to 0 of St), and the content of the leftmost digit (bits 15 to 12 of E) is lost.



Flags

Name	Label	Operation
Error Flag	ER	ON when St is greater than E. OFF in all other cases.

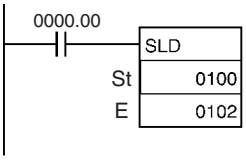
Precautions

When St is greater than E, an error will be generated and the Error Flag will turn ON.

Note When large amounts of data are shifted, the instruction execution time is quite long. Be sure that the power is not cut while SLD(074) is being executed, causing the shift operation to stop halfway through.

Examples

When CIO 0000.00 is ON, words CIO 0100 through CIO 0102 will shift by one digit (4 bits) to the left. A zero will be placed in bits 0 to 3 of word CIO 0100 and the contents of bits 12 to 15 of CIO 0102 will be lost.

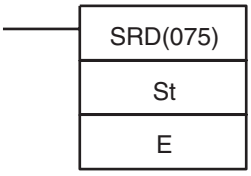


3-8-18 ONE DIGIT SHIFT RIGHT: SRD(075)

Purpose

Shifts data by one digit (4 bits) to the right.

Ladder Symbol



St: Starting word

E: End word

Variations

Variations		
	Executed Each Cycle for ON Condition	SRD(075)
	Executed Once for Upward Differentiation	@SRD(075)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

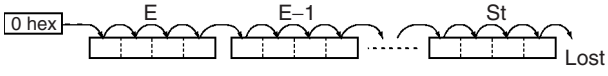
Note St and E must be in the same data area.

Operand Specifications

Area	St	E
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	
Auxiliary Bit Area	A448 to A959	
Timer Area	T0000 to T0255	
Counter Area	C0000 to C0255	
DM Area	D00000 to D32767	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 or ,IR1 –2048 to +2047 ,IR0 or –2048 to +2047 ,IR1 ,IR0+(++) or ,IR1+(++) , –(– –)IR0 or, –(– –)IR1	

Description

SRD(075) shifts data between St and E by one digit (4 bits) to the right. “0” is placed in the leftmost digit (bits 15 to 12 of E), and the content of the rightmost digit (bits 3 to 0 of St) is lost.



Flags

Name	Label	Operation
Error Flag	ER	ON when St is greater than E. OFF in all other cases.

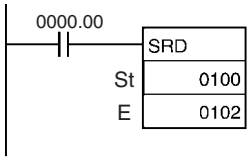
Precautions

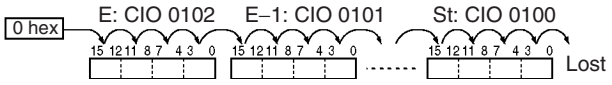
When St is greater than E, an error will be generated and the Error Flag will turn ON.

Note When large amounts of data are shifted, the instruction execution time is quite long. Always take care that the power is not cut while SRD(075) is being executed, causing the shift operation to stop halfway through.

Examples

When CIO 0000.00 is ON, words CIO 0100 through CIO 0102 will shift by one digit (4 bits) to the right. A zero will be placed in bits 12 to 15 of CIO 0102 and the contents of bits 0 to 3 of word CIO 0100 will be lost.



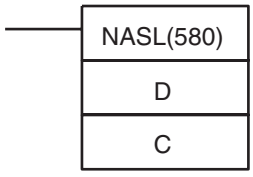


3-8-19 SHIFT N-BITS LEFT: NASL(580)

Purpose

Shifts the specified 16 bits of word data to the left by the specified number of bits.

Ladder Symbol



D: Shift word

C: Control word

Variations

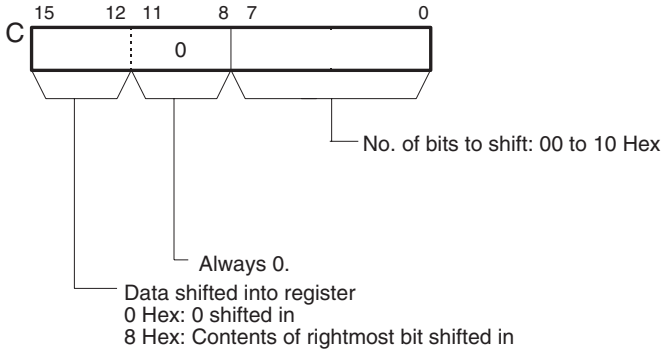
Variations		
	Executed Each Cycle for ON Condition	NASL(580)
	Executed Once for Upward Differentiation	@NASL(580)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C: Control Word

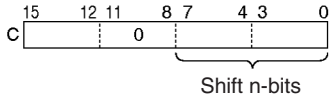


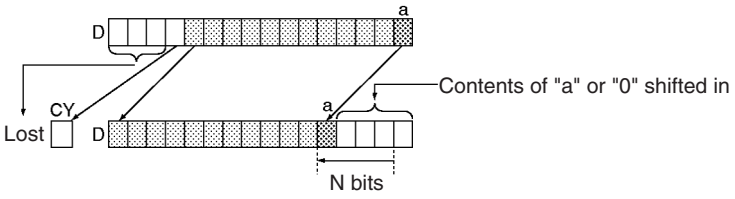
Operand Specifications

Area	D	C
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	
Auxiliary Bit Area	A448 to A959	A000 to A959
Timer Area	T0000 to T0255	
Counter Area	C0000 to C0255	
DM Area	D00000 to D32767	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	Specified values only
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

NASL(580) shifts D (the shift word) by the specified number of binary bits (specified in C) to the left (from the rightmost bit to the leftmost bit). Either zeros or the value of the rightmost bit will be placed into the specified number of bits of the shift word starting from the rightmost bit.





Flags

Name	Label	Operation
Error Flag	ER	ON when the control word C (the number of bits to shift) is not within range. OFF in all other cases.
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

For any bits which are shifted outside the specified word, the contents of the last bit is shifted to the Carry Flag (CY), and all other data is lost.

When the number of bits to shift (specified in C) is “0,” the data will not be shifted. The appropriate flags will turn ON and OFF, however, according to data in the specified word.

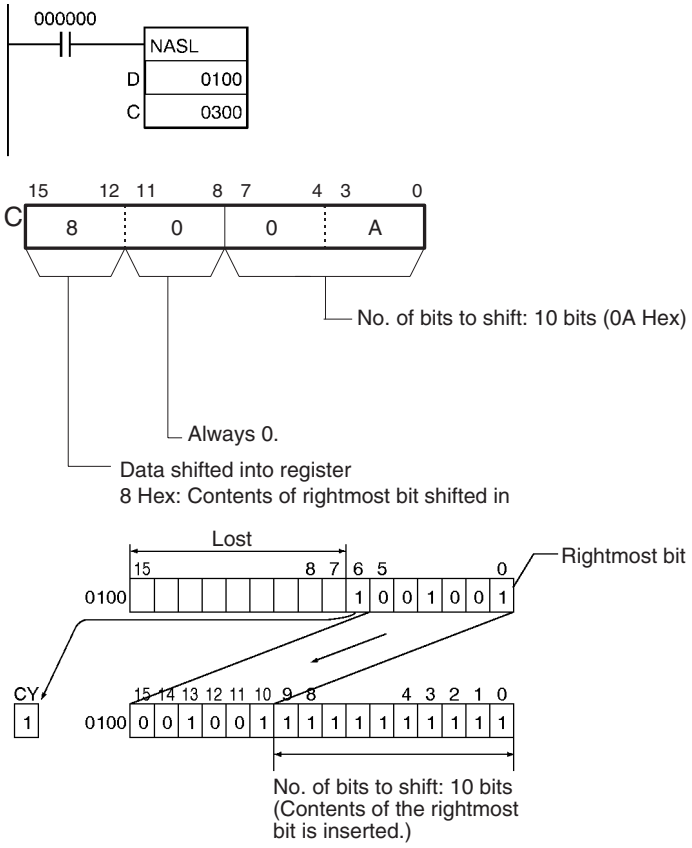
When the contents of the control word C is out of range, an error will be generated and the Error Flag will turn ON.

If as a result of the shift the contents of D is 0000 hex, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of D is 1, the Negative Flag will turn ON.

Examples

When CIO 000000 is ON, The contents of CIO 0100 is shifted 10 bits to the left (from the rightmost bit to the leftmost bit). The number of bits to shift is specified in bits 0 to 7 of word CIO 0300 (control data). The contents of bit 0 of CIO 0100 is copied into bits from which data was shifted and the contents of the rightmost bit which was shifted out of range is shifted into the Carry Flag (CY). All other data is lost.

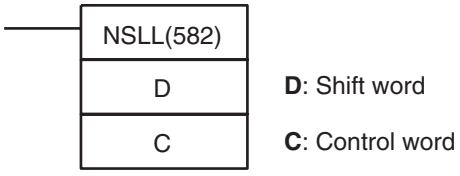


3-8-20 DOUBLE SHIFT N-BITS LEFT: NSLL(582)

Purpose

Shifts the specified 32 bits of word data to the left by the specified number of bits.

Ladder Symbol



Variations

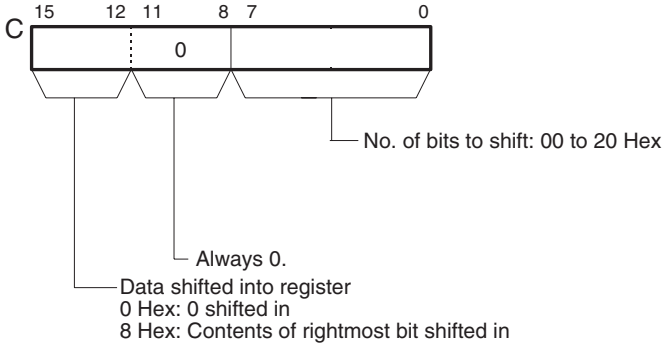
Variations		
	Executed Each Cycle for ON Condition	NSLL(582)
	Executed Once for Upward Differentiation	@NSLL(582)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C: Control Word

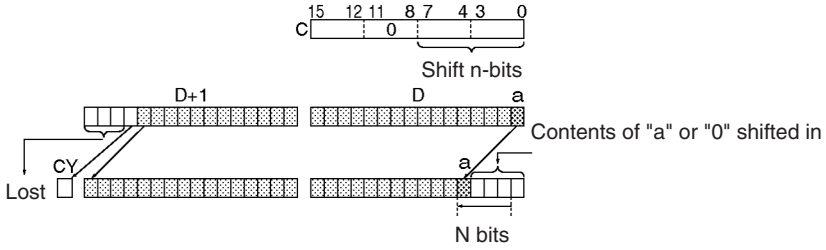


Operand Specifications

Area	D	C
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143
Work Area	W000 to W254	W000 to W255
Auxiliary Bit Area	A448 to A958	A000 to A959
Timer Area	T0000 to T0254	T0000 to T0255
Counter Area	C0000 to C0254	C0000 to C0255
DM Area	D00000 to D32766	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	Specified values only
Data Registers	---	DR0 to DR15
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

NSLL(582) shifts D and D+1 (the shift words) by the specified number of binary bits (specified in C) to the left (from the rightmost bit to the leftmost bit). Either zeros or the value of the rightmost bit will be placed into the specified number of bits of the shift word starting from the rightmost bit.



Flags

Name	Label	Operation
Error Flag	ER	ON when the control word C (the number of bits to shift) is not within range. OFF in all other cases.
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

For any bits which are shifted outside the specified word, the contents of the last bit is shifted to the Carry Flag (CY), and all other data is lost.

When the number of bits to shift (specified in C) is “0,” the data will not be shifted. The appropriate flags will turn ON and OFF, however, according to data in the specified word.

When the contents of the control word C are out of range, an error will be generated and the Error Flag will turn ON.

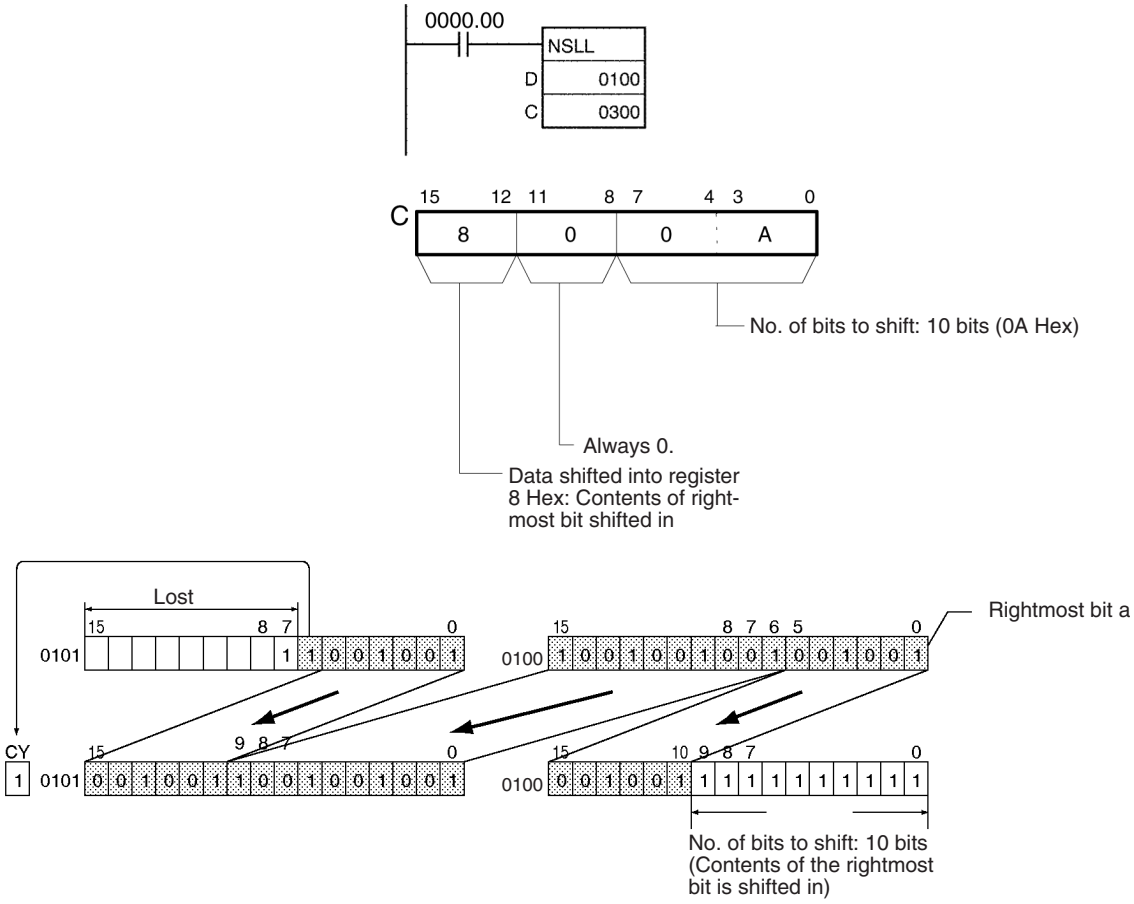
If as a result of the shift the contents of D is 0000, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of D, D +1 is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON, CIO 0100 and CIO 0101 will be shifted to the left (from the rightmost bit to the leftmost bit) by 10 bits. The number of bits to shift is specified in bits 0 to 7 of word CIO 0300 (control data). The contents of bit 0 of CIO 0100 is copied into bits from which data was shifted and the contents

of the rightmost bit which was shifted out of range is shifted into the Carry Flag (CY). All other data is lost.

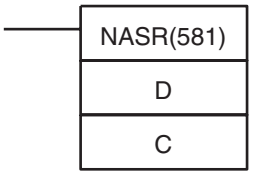


3-8-21 SHIFT N-BITS RIGHT: NASR(581)

Purpose

Shifts the specified 16 bits of word data to the right by the specified number of bits.

Ladder Symbol



D: Shift word

C: Control word

Variations

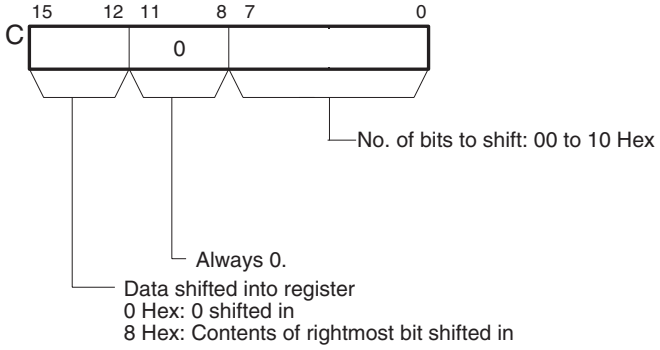
Variations		
	Executed Each Cycle for ON Condition	NASR(581)
	Executed Once for Upward Differentiation	@NASR(581)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C: Control Word

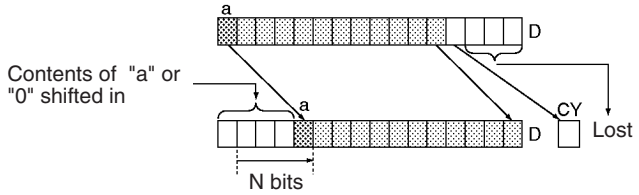


Operand Specifications

Area	D	C
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	
Auxiliary Bit Area	A448 to A959	A000 to A959
Timer Area	T0000 to T0255	
Counter Area	C0000 to C0255	
DM Area	D00000 to D32767	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	Specified values only
Data Registers	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

NASR(581) shifts D (the shift word) by the specified number of binary bits (specified in C) to the right (from the rightmost bit to the leftmost bit). Either zeros or the value of the rightmost bit will be placed into the specified number of bits of the shift word starting from the rightmost bit.



Flags

Name	Label	Operation
Error Flag	ER	ON when the control word C (the number of bits to shift) is not within range. OFF in all other cases.
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.

Name	Label	Operation
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

For any bits which are shifted outside the specified word, the contents of the last bit is shifted to the Carry Flag (CY), and all other data is discarded.

When the number of bits to shift (specified in C) is "0," the data will not be shifted. The appropriate flags will turn ON and OFF, however, according to data in the specified word.

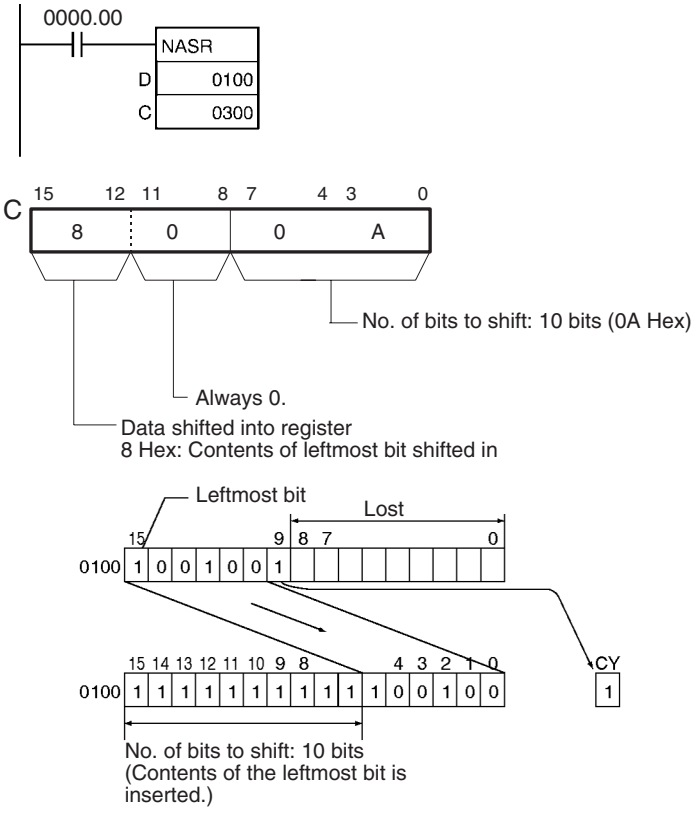
When the contents of the control word C are out of range, an error will be generated and the Error Flag will turn ON.

If as a result of the shift the contents of D is 0000 hex, the Equals Flag will turn ON.

If as a result of the shift the contents of the leftmost bit of D is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON, CIO 0100 will be shifted 10 bits to the right (from the leftmost bit to the rightmost bit). The number of bits to shift is specified in bits 0 to 7 of word CIO 0300. The contents of bit 15 of CIO 0100 is copied into the bits from which data was shifted and the contents of the leftmost bit of data which was shifted out of range, is shifted into the Carry Flag (CY). All other data is lost.

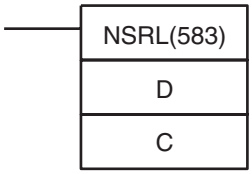


3-8-22 DOUBLE SHIFT N-BITS RIGHT: NSRL(583)

Purpose

Shifts the specified 32 bits of word data to the right by the specified number of bits.

Ladder Symbol



D: Shift word

C: Control word

Variations

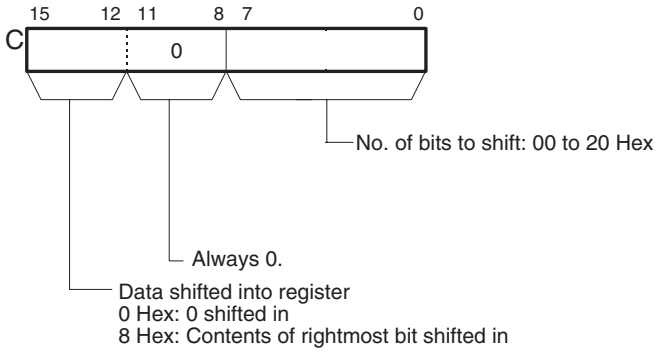
Variations		
	Executed Each Cycle for ON Condition	NSRL(583)
	Executed Once for Upward Differentiation	@NSRL(583)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C: Control Word

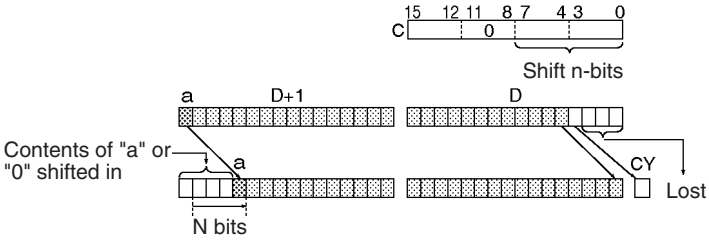


Operand Specifications

Area	D	C
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143
Work Area	W000 to W254	W000 to W255
Auxiliary Bit Area	A448 to A958	A000 to A959
Timer Area	T0000 to T0254	T0000 to T0255
Counter Area	C0000 to C0254	C0000 to C0255
DM Area	D00000 to D32766	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	Specified values only
Data Registers	---	DR0 to DR15
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 -2048 to +2047 ,IR0 to -2048 to +2047 ,IR15 DR0 to DR15,IR0 to IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

NSRL(583) shifts D and D+1 (the shift words) by the specified number of binary bits (specified in C) to the right (from the leftmost bit to the rightmost bit). Either zeros or the value of the rightmost bit will be placed into the specified number of bits of the shift word starting from the rightmost bit.



Flags

Name	Label	Operation
Error Flag	ER	ON when the control word C (the number of bits to shift) is not within range. OFF in all other cases.
Equals Flag	=	ON when the shift result is 0. OFF in all other cases.
Carry Flag	CY	ON when 1 is shifted into the Carry Flag (CY). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit is 1 as a result of the shift. OFF in all other cases.

Precautions

For any bits which are shifted outside the specified word, the contents of the last bit is shifted to the Carry Flag (CY), and all other data is lost.

When the number of bits to shift (specified in C) is “0,” the data will not be shifted. The appropriate flags will turn ON or OFF, however, according to data in the specified word.

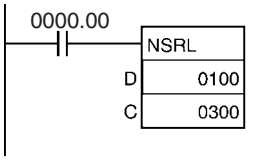
When the contents of the control word C are out of range, an error will be generated and the Error Flag will turn ON.

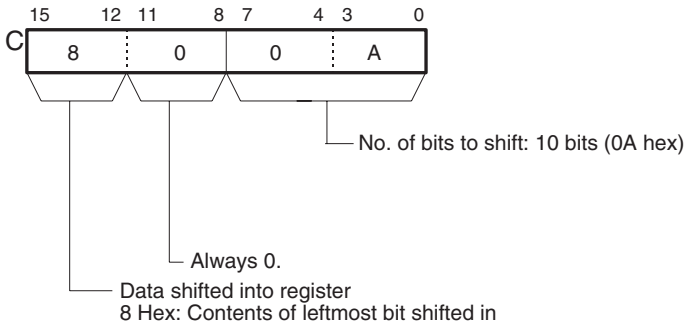
If as a result of the shift the contents of D +1 is 00000000 hex, the Equals Flag will turn ON.

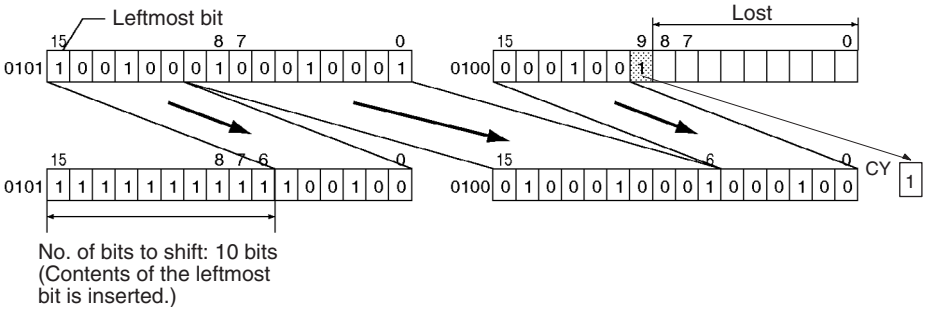
If as a result of the shift the contents of the leftmost bit of D +1 is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON, CIO 0100 and CIO 0101 will be shifted 10 bits to the right (from the leftmost bit to the rightmost bit). The number of bits to shift is specified in bits 0 to 7 of word CIO 0300 (control data). The contents of bit 15 of CIO will be copied into the bits from which data was shifted and the contents of the leftmost bit of data which was shifted out of range will be shifted into the Carry Flag (CY). All other data is lost.






Leftmost bit
Lost
15
8 7
0
0101 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1
0100 0 0 0 1 0 0 1
9
8 7
6
CY
1
0101 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
0100 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0
No. of bits to shift: 10 bits
(Contents of the leftmost
bit is inserted.)

3-9 Increment/Decrement Instructions

This section describes instructions used to increment and decrement binary or BCD values.

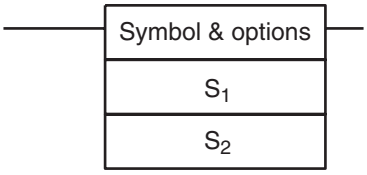
Instruction	Mnemonic	Function code	Page
INCREMENT BINARY	++	590	265
DOUBLE INCREMENT BINARY	++L	591	267
DECREMENT BINARY	– –	592	269
DOUBLE DECREMENT BINARY	– –L	593	271
INCREMENT BCD	++B	594	273
DOUBLE INCREMENT BCD	++BL	595	275
DECREMENT BCD	– –B	596	277
DOUBLE DECREMENT BCD	– –BL	597	279

3-9-1 INCREMENT BINARY: ++(590)

Purpose

Increments the 4-digit hexadecimal content of the specified word by 1.

Ladder Symbol



S1: Comparison data 1

S2: Comparison data 2

Variations

Variations		
	Executed Each Cycle for ON Condition	++(590)
	Executed Once for Upward Differentiation	@++(590)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

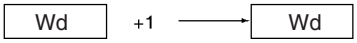
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W255
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
DM Area	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

The ++(590) instruction adds 1 to the binary content of Wd. The specified word will be incremented by 1 every cycle as long as the execution condition of ++(590) is ON. When the up-differentiated variation of this instruction (@++(590)) is used, the specified word is incremented only when the execution condition has gone from OFF to ON.



The Equals Flag will be turned ON if the result is 0000, and the Negative Flag will be turned ON when bit 15 of Wd is ON in the result.

Both the Equals Flag and the Carry Flag will be turned ON when the content of Wd changes from FFFF to 0000.

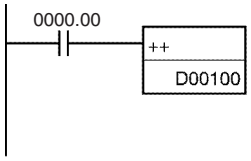
Flags

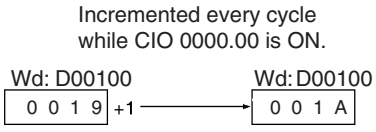
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the content of Wd is 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if the content of Wd changes from FFFF to 0000 during execution. OFF in all other cases.
Negative Flag	N	ON if bit 15 of Wd is ON after execution. OFF in all other cases.

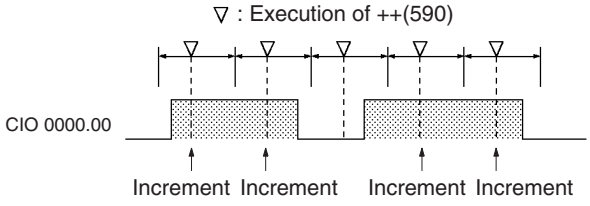
Examples

Operation of ++(590)

In the following example, the content of D00100 will be incremented by 1 every cycle as long as CIO 0000.00 is ON.

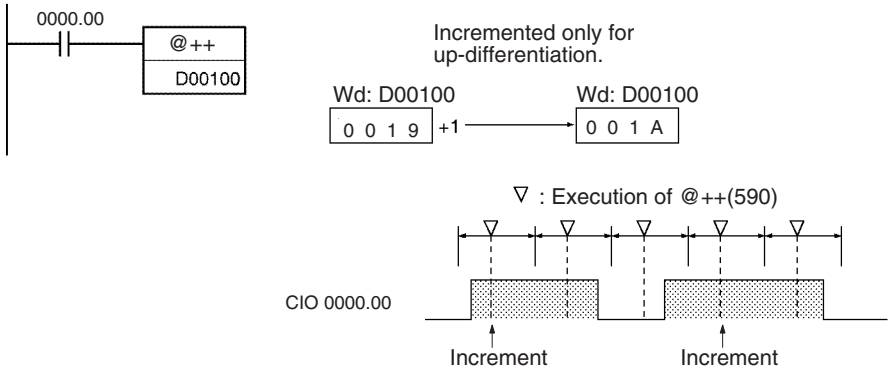






Operation of @++(590)

The up-differentiated variation is used in the following example, so the content of D00100 will be incremented by 1 only when CIO 0000.00 has gone from OFF to ON.

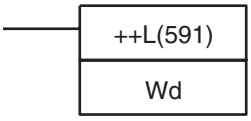


3-9-2 DOUBLE INCREMENT BINARY: ++L(591)

Purpose

Increments the 8-digit hexadecimal content of the specified words by 1.

Ladder Symbol



Wd: First word

Variations

Variations		
	Executed Each Cycle for ON Condition	++L(591)
	Executed Once for Upward Differentiation	@++L(591)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

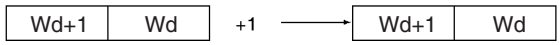
Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W254
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T0254
Counter Area	C0000 to C0254
DM Area	D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	---
Index Registers	IR0 or IR15
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

The ++L(591) instruction adds 1 to the 8-digit hexadecimal content of Wd+1 and Wd. The content of the specified words will be incremented by 1 every cycle as long as the execution condition of ++L(591) is ON. When the up-dif-

ferentiated variation of this instruction (@++L(591)) is used, the content of the specified words is incremented only when the execution condition has gone from OFF to ON.



The Equals Flag will be turned ON if the result is 0000 0000, and the Negative Flag will be turned ON if bit 15 of Wd+1 is ON in the result.

Both the Equals Flag and the Carry Flag will be turned ON when the content of Wd and Wd+1 changes from FFFF FFFF to 0000 0000.

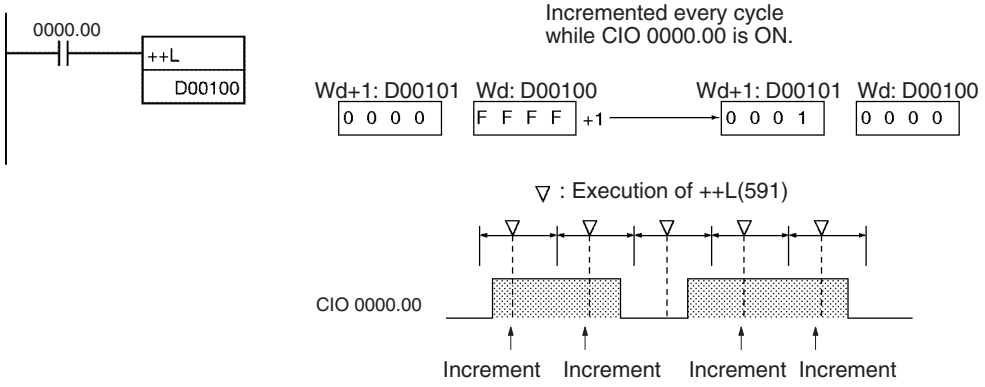
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result is 0000 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if the content of Wd and Wd+1 changes from FFFF FFFF to 0000 0000 during execution. OFF in all other cases.
Negative Flag	N	ON if bit 15 of Wd+1 is ON after execution. OFF in all other cases.

Examples

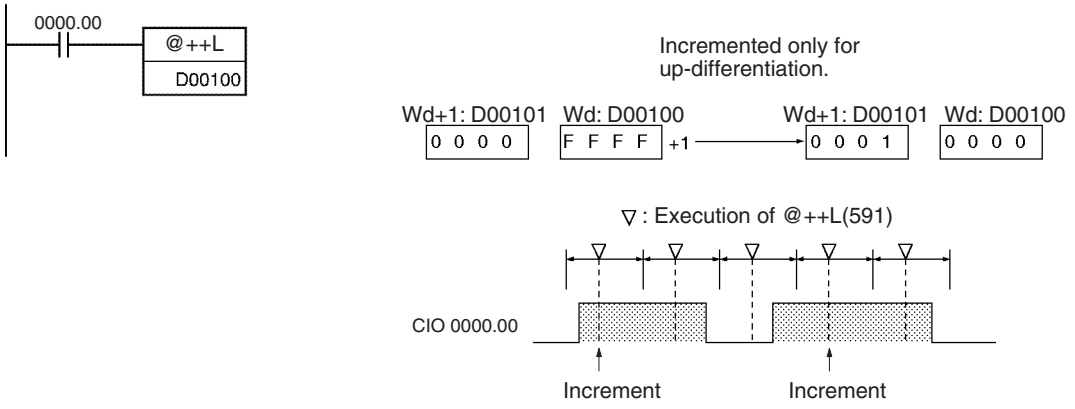
Operation of ++L(591)

In the following example, the 8-digit hexadecimal content of D00101 and D00100 will be incremented by 1 every cycle as long as CIO 0000.00 is ON.



Operation of @++L(591)

The up-differentiated variation is used in the following example, so the content of D00101 and D00100 will be incremented by 1 only when CIO 0000.00 has gone from OFF to ON.



3-9-3 DECREMENT BINARY: – –(592)

Purpose

Decrements the 4-digit hexadecimal content of the specified word by 1.

Ladder Symbol

Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	– – (592)
	Executed Once for Upward Differentiation	@– – (592)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

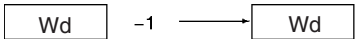
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W255
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
DM Area	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

The – –(592) instruction subtracts 1 from the binary content of Wd. The specified word will be decremented by 1 every cycle as long as the execution condition of – –(592) is ON. When the up-differentiated variation of this instruction (@– –(592)) is used, the specified word is decremented only when the execution condition has gone from OFF to ON.



The Equals Flag will be turned ON if the result is 0000, and the Negative Flag will be turned ON if bit 15 of Wd is ON in the result.

Both the Carry Flag and the Negative Flag will be turned ON when the content of Wd changes from 0000 to FFFF.

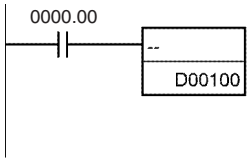
Flags

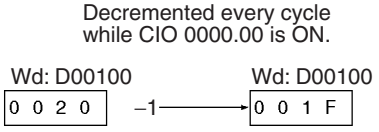
Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the content of Wd is 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if the content of Wd changes from 0000 to FFFF during execution. OFF in all other cases.
Negative Flag	N	ON if bit 15 of Wd is ON after execution. OFF in all other cases.

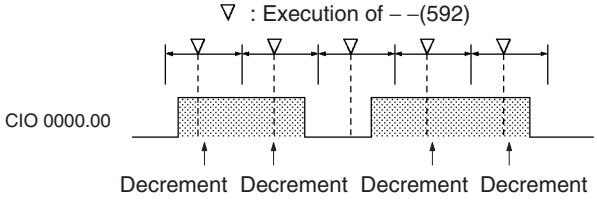
Examples

Operation of – –(592)

In the following example, the content of D00100 will be decremented by 1 every cycle as long as CIO 0000.00 is ON.

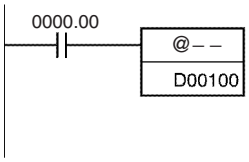


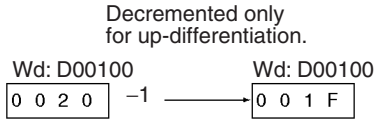


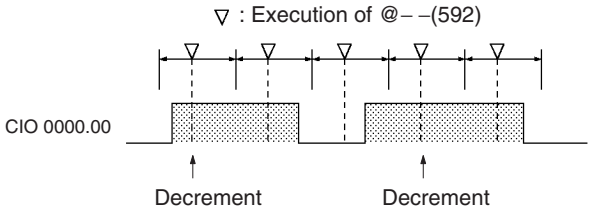


Operation of @– –(592)

The up-differentiated variation is used in the following example, so the content of D00100 will be decremented by 1 only when CIO 0000.00 has gone from OFF to ON.





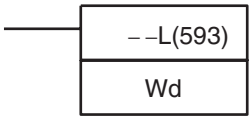


3-9-4 DOUBLE DECREMENT BINARY: – –L(593)

Purpose

Decrements the 8-digit hexadecimal content of the specified words by 1.

Ladder Symbol



Wd: First word

Variations

Variations		
	Executed Each Cycle for ON Condition	– –L(593)
	Executed Once for Upward Differentiation	@– –L(593)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

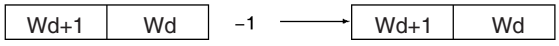
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W254
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T0254
Counter Area	C0000 to C0254
DM Area	D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	---
Index Registers	IR0 or IR15
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

The – –L(593) instruction subtracts 1 from the 8-digit hexadecimal content of Wd+1 and Wd. The content of the specified words will be decremented by 1 every cycle as long as the execution condition of – –L(593) is ON. When the up-differentiated variation of this instruction (@– –L(593)) is used, the content of the specified words is decremented only when the execution condition has gone from OFF to ON.



The Equals Flag will be turned ON if the result is 0000 0000, and the Negative Flag will be turned ON if bit 15 of Wd+1 is ON in the result.

Both the Carry Flag and the Negative Flag will be turned ON when the content changes from 0000 0000 to FFFF FFFF.

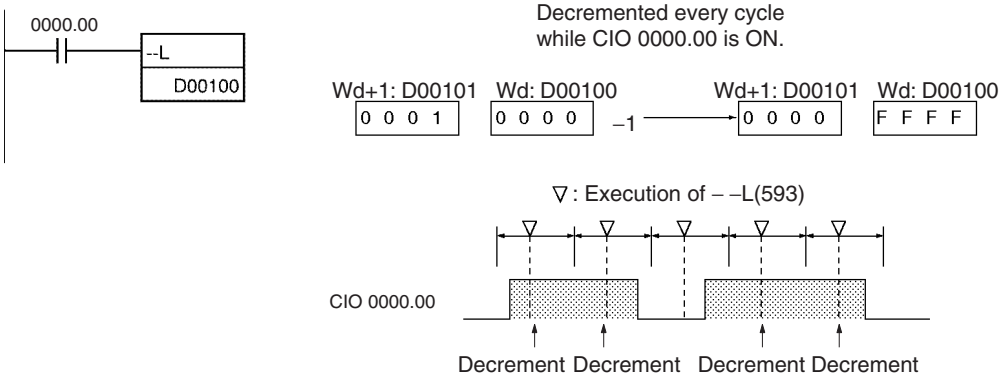
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result is 0000 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if the content of Wd and Wd+1 changes from 0000 0000 to FFFF FFFF during execution. OFF in all other cases.
Negative Flag	N	ON if bit 15 of Wd+1 is ON after execution. OFF in all other cases.

Examples

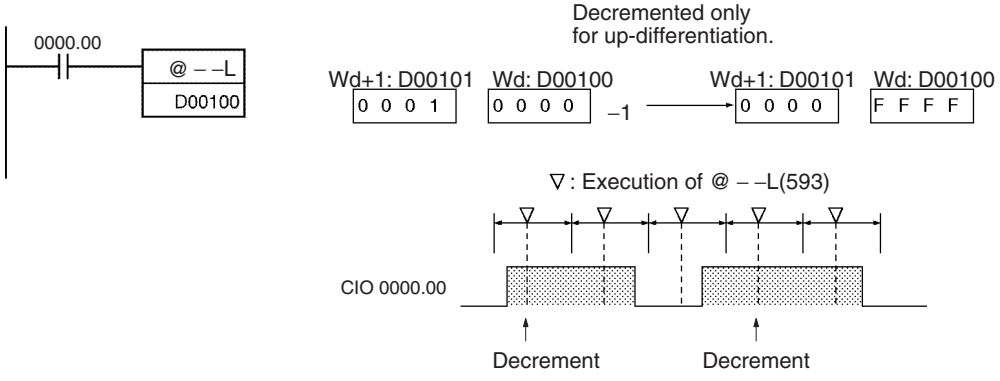
Operation of – –L(593)

In the following example, the 8-digit hexadecimal content of D00101 and D00100 will be decremented by 1 every cycle as long as CIO 0000.00 is ON.



Operation of @– –L(593)

The up-differentiated variation is used in the following example, so the content of D00101 and D00100 will be decremented by 1 only when CIO 0000.00 has gone from OFF to ON.



3-9-5 INCREMENT BCD: ++B(594)

Purpose

Increments the 4-digit BCD content of the specified word by 1.

Ladder Symbol

Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	++B(594)
	Executed Once for Upward Differentiation	@++B(594)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

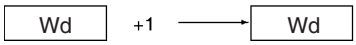
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W255
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
DM Area	D00000 to D32767
Indirect DM addresses in BCD	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

The ++B(594) instruction adds 1 to the BCD content of Wd. The specified word will be incremented by 1 every cycle as long as the execution condition of ++B(594) is ON. When the up-differentiated variation of this instruction (@++B(594)) is used, the specified word is incremented only when the execution condition has gone from OFF to ON.



The Equals Flag will be turned ON if the result is 0000.

Both the Equals Flag and the Carry Flag will be turned ON when the content of Wd changes from 9999 to 0000.

Flags

Name	Label	Operation
Error Flag	ER	ON if the content of Wd is not BCD. OFF in all other cases.
Equals Flag	=	ON if the content of Wd is 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if the content of Wd changes from 9999 to 0000 during execution. OFF in all other cases.

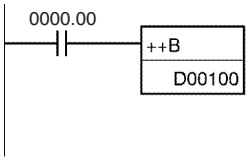
Precautions

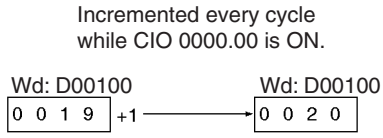
The content of Wd must be BCD. If it is not BCD, an error will occur and the Error Flag will be turned ON.

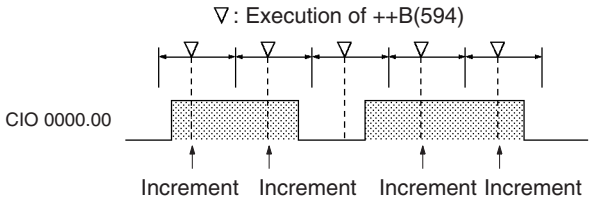
Examples

Operation of ++B(594)

In the following example, the BCD content of D00100 will be incremented by 1 every cycle as long as CIO 0000.00 is ON.

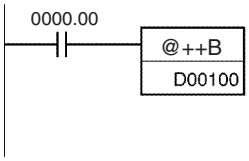


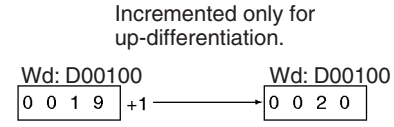


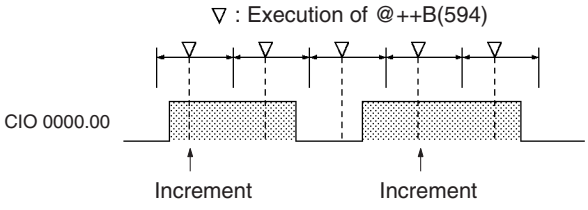


Operation of @++B(594)

The up-differentiated variation is used in the following example, so the content of D00100 will be incremented by 1 only when CIO 0000.00 has gone from OFF to ON.







3-9-6 DOUBLE INCREMENT BCD: ++BL(595)

Purpose

Increments the 8-digit BCD content of the specified words by 1.

Ladder Symbol

Wd: First word

Variations

Variations		
	Executed Each Cycle for ON Condition	++BL(595)
	Executed Once for Upward Differentiation	@++BL(595)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

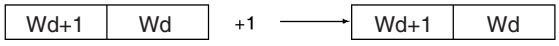
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W254
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T0254
Counter Area	C0000 to C0254
DM Area	D00000 to D32766
Indirect DM addresses in BCD	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

The ++BL(595) instruction adds 1 to the 8-digit BCD content of Wd+1 and Wd. The content of the specified words will be incremented by 1 every cycle as long as the execution condition of ++BL(595) is ON. When the up-differentiated variation of this instruction (@++BL(595)) is used, the content of the specified words is incremented only when the execution condition has gone from OFF to ON.



The Equals Flag will be turned ON if the result is 0000 0000.

Both the Equals Flag and the Carry Flag will be turned ON when the content of Wd and Wd+1 changes from 9999 9999 to 0000 0000.

Flags

Name	Label	Operation
Error Flag	ER	ON if the content of Wd+1 and Wd is not BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0000 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if the content of Wd and Wd+1 changes from 9999 9999 to 0000 0000 during execution. OFF in all other cases.

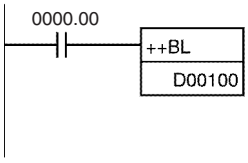
Precautions

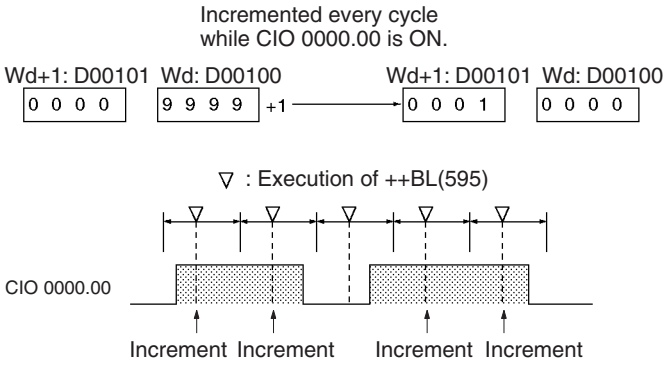
The content of Wd+1 and Wd must be BCD. If it is not BCD, an error will occur and the Error Flag will be turned ON.

Examples

Operation of ++BL(595)

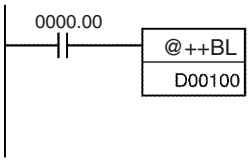
In the following example, the 8-digit BCD content of D00101 and D00100 will be incremented by 1 every cycle as long as CIO 0000.00 is ON.

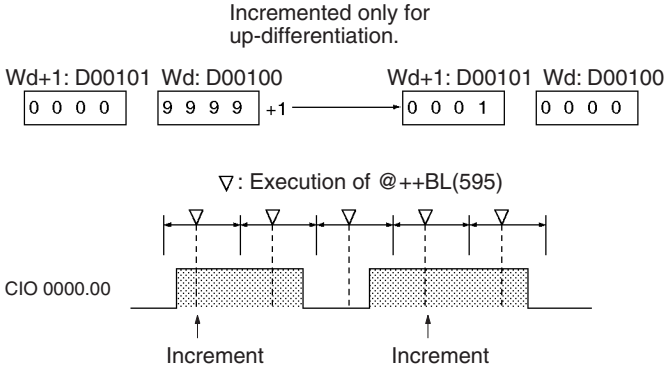




Operation of @++BL(595)

The up-differentiated variation is used in the following example, so the BCD content of D00101 and D00100 will be incremented by 1 only when CIO 0000.00 has gone from OFF to ON.



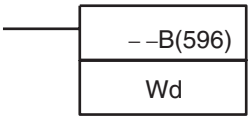


3-9-7 DECREMENT BCD: – –B(596)

Purpose

Decrements the 4-digit BCD content of the specified word by 1.

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	– –B(596)
	Executed Once for Upward Differentiation	@– –B(596)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

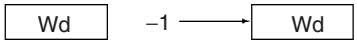
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W255
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
DM Area	D00000 to D32767
Indirect DM addresses in BCD	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	DR0 to DR15
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

The – –B(596) instruction subtracts 1 from the BCD content of Wd. The specified word will be decremented by 1 every cycle as long as the execution condition of – –B(596) is ON. When the up-differentiated variation of this instruction (@– –B(596)) is used, the specified word is decremented only when the execution condition has gone from OFF to ON.



The Equals Flag will be turned ON if the result is 0000 and the Carry Flag will be turned ON when the content of Wd changes from 0000 to 9999.

Flags

Name	Label	Operation
Error Flag	ER	ON if the content of Wd is not BCD. OFF in all other cases.
Equals Flag	=	ON if the content of Wd is 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if the content of Wd changes from 0000 to 9999 during execution. OFF in all other cases.

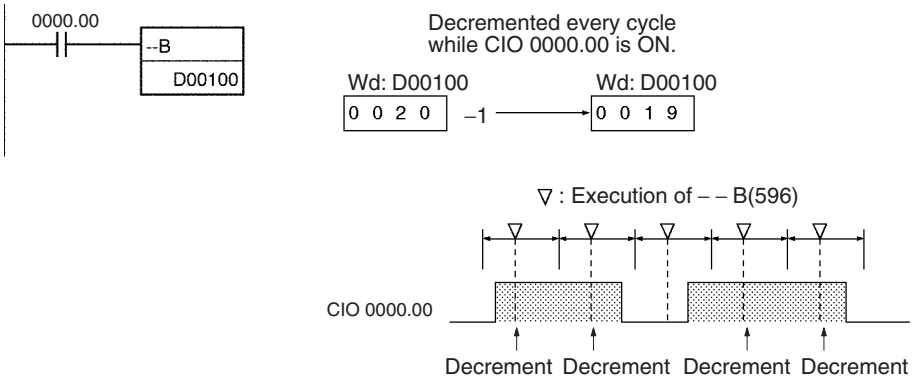
Precautions

The content of Wd must be BCD. If it is not BCD, an error will occur and the Error Flag will be turned ON.

Examples

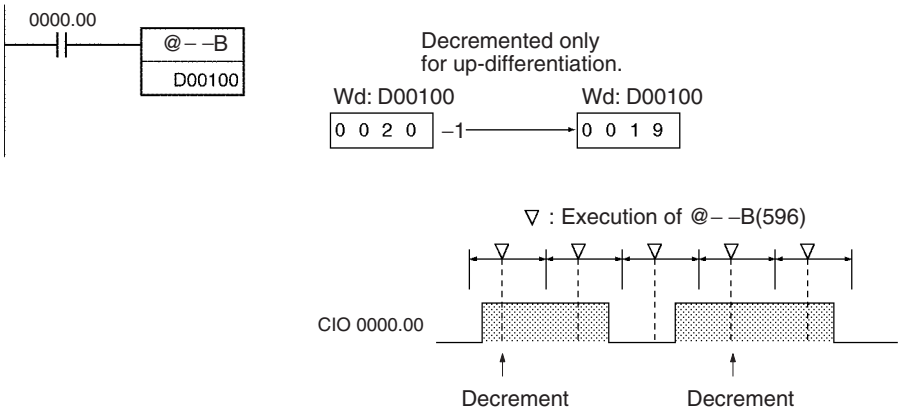
Operation of – –B(596)

In the following example, the BCD content of D00100 will be decremented by 1 every cycle as long as CIO 0000.00 is ON.



Operation of @– –B(596)

The up-differentiated variation is used in the following example, so the BCD content of D00100 will be decremented by 1 only when CIO 0000.00 has gone from OFF to ON.

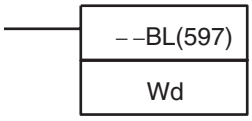


3-9-8 DOUBLE DECREMENT BCD: – –BL(597)

Purpose

Decrements the 8-digit BCD content of the specified words by 1.

Ladder Symbol



Wd: First word

Variations

Variations		
	Executed Each Cycle for ON Condition	– –BL(597)
	Executed Once for Upward Differentiation	@– –BL(597)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

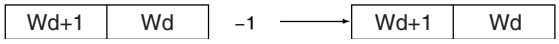
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W254
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T0254
Counter Area	C0000 to C0254
DM Area	D00000 to D32766
Indirect DM addresses in BCD	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

The – –BL(597) instruction subtracts 1 from the 8-digit BCD content of Wd+1 and Wd. The content of the specified words will be decremented by 1 every cycle as long as the execution condition of – –BL(597) is ON. When the up-differentiated variation of this instruction (@– –BL(597)) is used, the content of the specified words is decremented only when the execution condition has gone from OFF to ON.



The Equals Flag will be turned ON if the result is 0000 0000 and the Carry Flag will be turned ON when the content of Wd and Wd+1 changes from 0000 0000 to 9999 9999.

Flags

Name	Label	Operation
Error Flag	ER	ON if the content of Wd+1 and Wd is not BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0000 0000 after execution. OFF in all other cases.
Carry Flag	CY	ON if the content of Wd and Wd+1 changes from 0000 0000 to 9999 9999 during execution. OFF in all other cases.

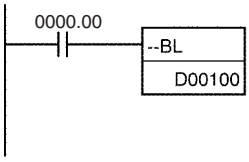
Precautions

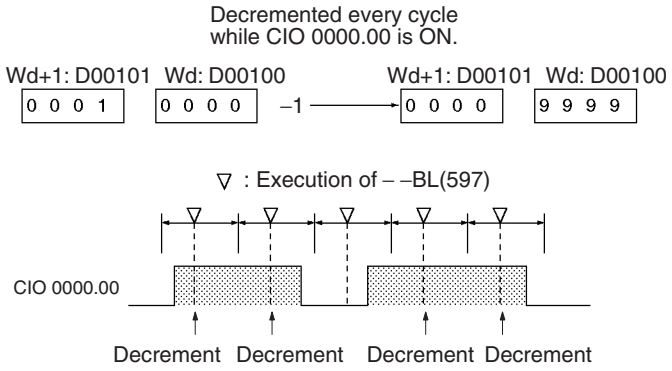
The content of Wd+1 and Wd must be BCD. If it is not BCD, an error will occur and the Error Flag will be turned ON.

Examples

Operation of – –BL(597)

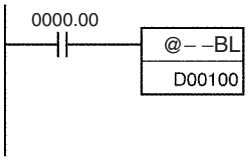
In the following example, the 8-digit BCD content of D00101 and D00100 will be decremented by 1 every cycle as long as CIO 0000.00 is ON.

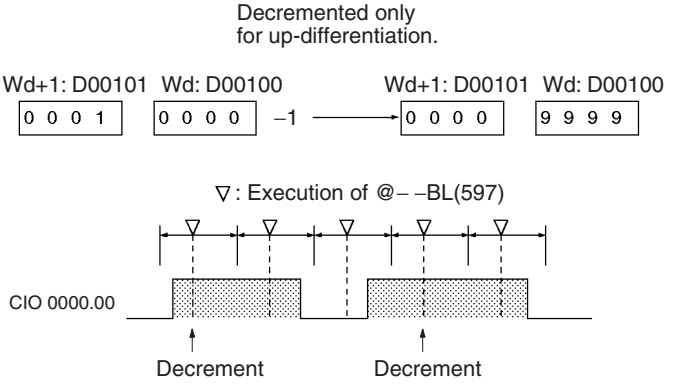




Operation of @– –BL(597)

The up-differentiated variation is used in the following example, so the BCD content of D00101 and D00100 will be decremented by 1 only when CIO 0000.00 has gone from OFF to ON.





3-10 Symbol Math Instructions

This section describes the Symbol Math Instructions, which perform arithmetic operations on BCD or binary data.

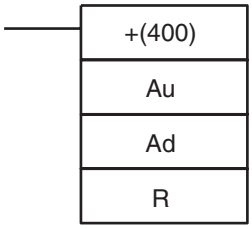
Instruction	Mnemonic	Function code	Page
SIGNED BINARY ADD WITHOUT CARRY	+	400	282
DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+L	401	283
SIGNED BINARY ADD WITH CARRY	+C	402	285
DOUBLE SIGNED BINARY ADD WITH CARRY	+CL	403	287
BCD ADD WITHOUT CARRY	+B	404	289
DOUBLE BCD ADD WITHOUT CARRY	+BL	405	290
BCD ADD WITH CARRY	+BC	406	292
DOUBLE BCD ADD WITH CARRY	+BCL	407	293
SIGNED BINARY SUBTRACT WITHOUT CARRY	–	410	295
DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	–L	411	296
SIGNED BINARY SUBTRACT WITH CARRY	–C	412	300
DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	–CL	413	302
BCD SUBTRACT WITHOUT CARRY	–B	414	304
DOUBLE BCD SUBTRACT WITHOUT CARRY	–BL	415	306
BCD SUBTRACT WITH CARRY	–BC	416	309
DOUBLE BCD SUBTRACT WITH CARRY	–BCL	417	310
SIGNED BINARY MULTIPLY	*	420	312
DOUBLE SIGNED BINARY MULTIPLY	*L	421	314
UNSIGNED BINARY MULTIPLY	*U	422	315
DOUBLE UNSIGNED BINARY MULTIPLY	*UL	423	317
BCD MULTIPLY	*B	424	318
DOUBLE BCD MULTIPLY	*BL	425	320
SIGNED BINARY DIVIDE	/	430	321
DOUBLE SIGNED BINARY DIVIDE	/L	431	323
UNSIGNED BINARY DIVIDE	/U	432	324
DOUBLE UNSIGNED BINARY DIVIDE	/UL	433	326
BCD DIVIDE	/B	434	328
DOUBLE BCD DIVIDE	/BL	435	329

3-10-1 SIGNED BINARY ADD WITHOUT CARRY: +(400)

Purpose

Adds 4-digit (single-word) hexadecimal data and/or constants.

Ladder Symbol



Au: Augend word

Ad: Addend word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	+(400)
	Executed Once for Upward Differentiation	@+(400)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

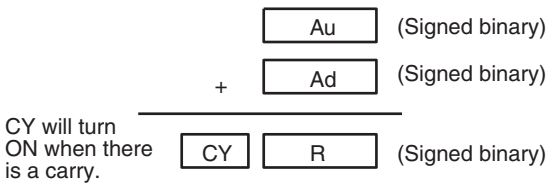
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)		---
Data Resisters	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

+(400) adds the binary values in Au and Ad and outputs the result to R.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.
Overflow Flag	OF	ON when the result of adding two positive numbers is in the range 8000 to FFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the result of adding two negative numbers is in the range 0000 to 7FFF hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

When +(400) is executed, the Error Flag will turn OFF.

If as a result of the addition, the content of R is 0000 hex, the Equals Flag will turn ON.

If the addition results in a carry, the Carry Flag will turn ON.

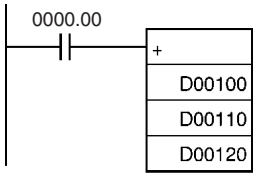
If the result of adding two positive numbers is negative (in the range 8000 to FFFF hex), the Overflow Flag will turn ON.

If the result of adding two negative numbers is positive (in the range 0000 to 7FFF hex), the Underflow Flag will turn ON.

If as a result of the addition, the content of the leftmost bit of R is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00100 and D00110 will be added as 4-digit signed binary values and the result will be output to D00120.

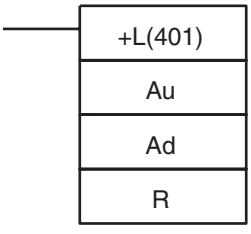


3-10-2 DOUBLE SIGNED BINARY ADD WITHOUT CARRY: +L(401)

Purpose

Adds 8-digit (double-word) hexadecimal data and/or constants.

Ladder Symbol



Au: 1st augend word

Ad: 1st addend word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	+L(401)
	Executed Once for Upward Differentiation	@+L(401)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

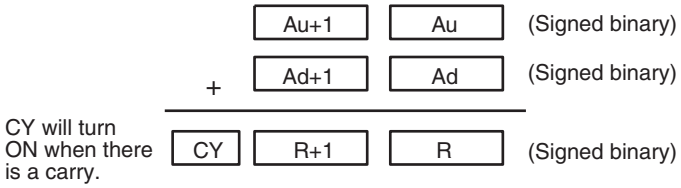
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)		---
Data Resisters	---		
Index Registers	IR0 or IR15		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

+L(401) adds the binary values in Au and Au+1 and Ad and Ad+1 and outputs the result to R and R+1.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000 0000. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.
Overflow Flag	OF	ON when the result of adding two positive numbers is in the range 8000 0000 to FFFF FFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the result of adding two negative numbers is in the range 0000 0000 to 7FFF FFFF hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

When +L(401) is executed, the Error Flag will turn OFF.

If as a result of the addition, the content of R, R+1 is 0000 0000 hex, the Equals Flag will turn ON.

If the addition results in a carry, the Carry Flag will turn ON.

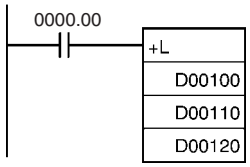
If the result of adding two positive numbers is negative (in the range 8000 0000 to FFFF FFFF hex), the Overflow Flag will turn ON.

If the result of adding two negative numbers is positive (in the range 0000 0000 to 7FFF FFFF hex), the Underflow Flag will turn ON.

If as a result of the addition, the content of the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON, D00100 and D00110 and D00111 and D00110 will be added as 8-digit signed binary values and the result will be output to D00120 and D00121.

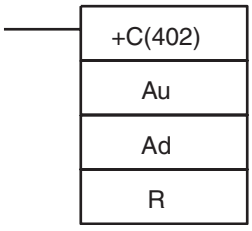


3-10-3 SIGNED BINARY ADD WITH CARRY: +C(402)

Purpose

Adds 4-digit (single-word) hexadecimal data and/or constants with the Carry Flag (CY).

Ladder Symbol



Au: Augend word

Ad: Addend word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	+C(402)
	Executed Once for Upward Differentiation	@+C(402)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

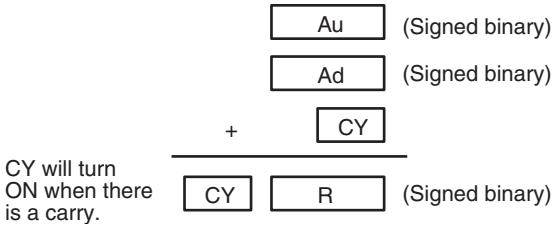
Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		

Area	Au	Ad	R
Constants	#0000 to #FFFF (binary)		---
Data Resisters	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

+C(402) adds the binary values in Au, Ad, and CY and outputs the result to R.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the addition result is 0000. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.
Overflow Flag	OF	ON when the result of adding two positive numbers and CY is in the range 8000 to FFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the result of adding two negative numbers and CY is in the range 0000 to 7FFF hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

When +C(402) is executed, the Error Flag will turn OFF.

If as a result of the addition, the content of R is 0000 hex, the Equals Flag will turn ON.

If the addition results in a carry, the Carry Flag will turn ON.

If the result of adding two positive numbers and CY is negative (in the range 8000 to FFFF hex), the Overflow Flag will turn ON.

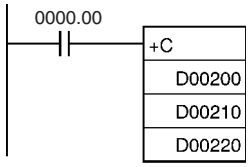
If the result of adding two negative numbers and CY is positive (in the range 0000 to 7FFF hex), the Underflow Flag will turn ON.

If as a result of the addition, the content of the leftmost bit of R is 1, the Negative Flag will turn ON.

Note To clear the Carry Flag (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 0000.00 is ON, D00200, D00210, and CY will be added as 4-digit signed binary values and the result will be output to D00220.

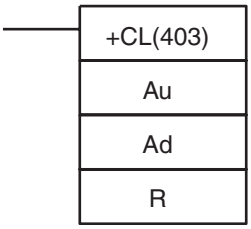


3-10-4 DOUBLE SIGNED BINARY ADD WITH CARRY: +CL(403)

Purpose

Adds 8-digit (double-word) hexadecimal data and/or constants with the Carry Flag (CY).

Ladder Symbol



Au: 1st augend word

Ad: 1st addend word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	+CL(403)
	Executed Once for Upward Differentiation	@+CL(403)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

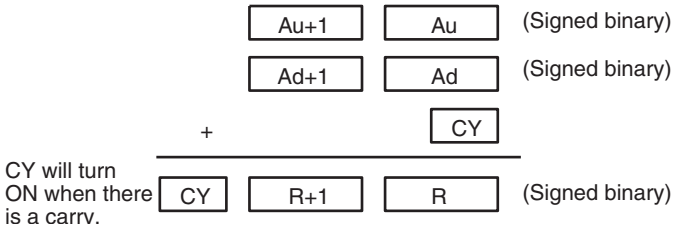
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)		---
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

+CL(403) adds the binary values in Au and Au+1, Ad and Ad+1, and CY and outputs the result to R and R+1.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000 0000. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.
Overflow Flag	OF	ON when the result of adding two positive numbers and CY is in the range 8000 0000 to FFFF FFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the result of adding two negative numbers and CY is in the range 0000 0000 to 7FFF FFFF hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

When +CL(403) is executed, the Error Flag will turn OFF.

If as a result of the addition, the content of R, R+1 is 0000 0000 hex, the Equals Flag will turn ON.

If the addition results in a carry, the Carry Flag will turn ON.

If the result of adding two positive numbers and CY is negative (in the range 8000 0000 to FFFF FFFF hex), the Overflow Flag will turn ON.

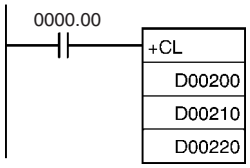
If the result of adding two negative numbers and CY is positive (in the range 0000 0000 to 7FFF FFFF hex), the Underflow Flag will turn ON.

If as a result of the addition, the content of the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Note To clear the Carry Flag (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 0000.00 is ON, D00201, D00200, D00211, D00210, and CY will be added as 8-digit signed binary values, and the result will be output to D00221 and D00220.

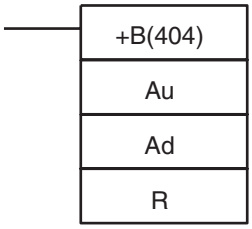


3-10-5 BCD ADD WITHOUT CARRY: +B(404)

Purpose

Adds 4-digit (single-word) BCD data and/or constants.

Ladder Symbol



Au: Augend word

Ad: Addend word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	+B(404)
	Executed Once for Upward Differentiation	@+B(404)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

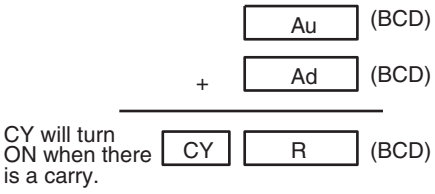
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #9999 (BCD)		---
Data Resisters	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

+B(404) adds the BCD values in Au and Ad and outputs the result to R.



Flags

Name	Label	Operation
Error Flag	ER	ON when Au is not BCD. ON when Ad is not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0000. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.

Precautions

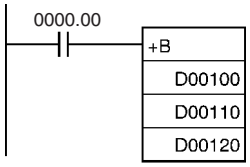
If Au or Ad is not BCD, an error is generated and the Error Flag will turn ON.

If as a result of the addition, the content of R is 0000 hex, the Equals Flag will turn ON.

If an addition results in a carry, the Carry Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00100 and D00110 will be added as 4-digit BCD values, and the result will be output to D00120.

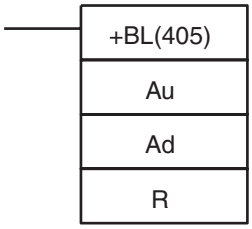


3-10-6 DOUBLE BCD ADD WITHOUT CARRY: +BL(405)

Purpose

Adds 8-digit (double-word) BCD data and/or constants.

Ladder Symbol



Au: 1st augend word

Ad: 1st addend word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	+BL(405)
	Executed Once for Upward Differentiation	@+BL(405)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

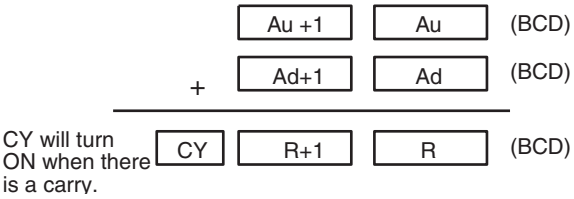
Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		

Area	Au	Ad	R
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #9999 9999 (BCD)		---
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

+BL(405) adds the BCD values in Au and Au+1 and Ad and Ad+1 and outputs the result to R, R+1.



Flags

Name	Label	Operation
Error Flag	ER	ON when Au, Au +1 is not BCD. ON when Ad, Ad +1 is not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0000 0000. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.

Precautions

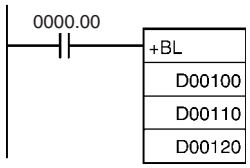
If Au, Au +1 or Ad, Ad +1 are not BCD, an error is generated and the Error Flag will turn ON.

If as a result of the addition, the content of R, R +1 is 0000 0000 hex, the Equals Flag will turn ON.

If an addition results in a carry, the Carry Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00101 and D00100 and D00111 and D00110 will be added as 8-digit BCD values, and the result will be output to D00121 and D00120.

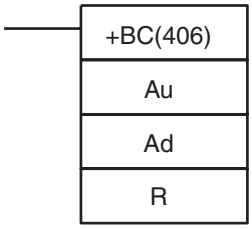


3-10-7 BCD ADD WITH CARRY: +BC(406)

Purpose

Adds 4-digit (single-word) BCD data and/or constants with the Carry Flag (CY).

Ladder Symbol



Au: Augend word

Ad: Addend word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	+BC(406)
	Executed Once for Upward Differentiation	@+BC(406)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

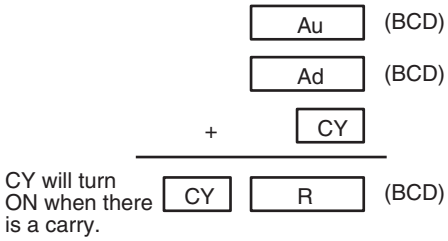
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #9999 (BCD)		---
Data Resisters	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

+BC(406) adds BCD values in Au, Ad, and CY and outputs the result to R.



Flags

Name	Label	Operation
Error Flag	ER	ON when Au is not BCD. ON when Ad is not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0000. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.

Precautions

If Au or Ad is not BCD, an error is generated and the Error Flag will turn ON.

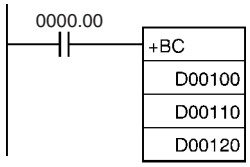
If as a result of the addition, the content of R is 0000 hex, the Equals Flag will turn ON.

If an addition results in a carry, the Carry Flag will turn ON.

Note To clear the Carry Flag (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 0000.00 is ON in the following example, D00100, D00110, and CY will be added as 4-digit BCD values, and the result will be output to D00120.

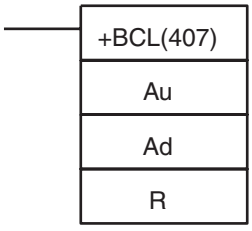


3-10-8 DOUBLE BCD ADD WITH CARRY: +BCL(407)

Purpose

Adds 8-digit (double-word) BCD data and/or constants with the Carry Flag (CY).

Ladder Symbol



Au: 1st augend word

Ad: 1st addend word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	+BCL(407)
	Executed Once for Upward Differentiation	@+BCL(407)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

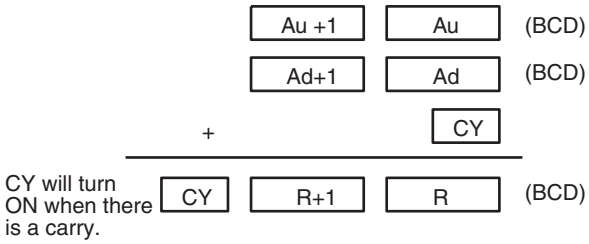
Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		

Area	Au	Ad	R
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #9999 9999 (BCD)		---
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

+BCL(407) adds the BCD values in Au and Au+1, Ad and Ad+1, and CY and outputs the result to R, R+1.



Flags

Name	Label	Operation
Error Flag	ER	ON when Au, Au +1 is not BCD. ON when Ad, Ad +1 is not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0000 0000. OFF in all other cases.
Carry Flag	CY	ON when the addition results in a carry. OFF in all other cases.

Precautions

If Au, Au +1 or Ad, Ad +1 are not BCD, an error is generated and the Error Flag will turn ON.

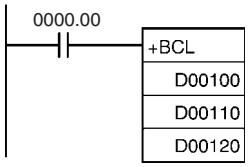
If as a result of the addition, the content of R, R +1 is 0000 0000 hex, the Equals Flag will turn ON.

If an addition results in a carry, the Carry Flag will turn ON.

Note To clear the Carry Flag (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 0000.00 is ON in the following example, D00101, D00100, D00111, D00110, and CY will be added as 8-digit BCD values, and the result will be output to D00121 and D00120.

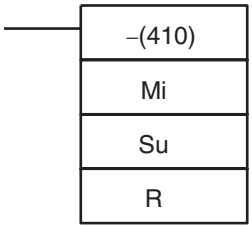


3-10-9 SIGNED BINARY SUBTRACT WITHOUT CARRY: –(410)

Purpose

Subtracts 4-digit (single-word) hexadecimal data and/or constants.

Ladder Symbol



Mi: Minuend word

Su: Subtrahend word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	–(410)
	Executed Once for Upward Differentiation	@–(410)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

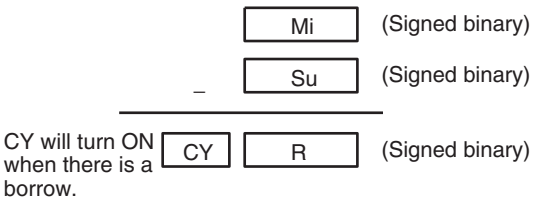
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)		---
Data Resisters	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

–(410) subtracts the binary values in Su from Mi and outputs the result to R. When the result is negative, it is output to R as a 2's complement. (Refer to *3-10-10 DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY: –L(411)* for an example of handling 2's complements.)



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000. OFF in all other cases.
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.
Overflow Flag	OF	ON when the result of subtracting a negative number from a positive number is in the range 8000 to FFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the result of subtracting a positive number from a negative number is in the range 0000 to 7FFF hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

When –(410) is executed, the Error Flag will turn OFF.

If as a result of the subtraction, the content of R is 0000 hex, the Equals Flag will turn ON.

If the subtraction results in a borrow, the Carry Flag will turn ON.

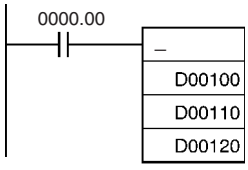
If the result of subtracting a negative number from a positive number is negative (in the range 8000 to FFFF hex), the Overflow Flag will turn ON.

If the result of subtracting a positive number from a negative number is positive (in the range 0000 to 7FFF hex), the Underflow Flag will turn ON.

If as a result of the subtraction, the content of the leftmost bit of R is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00110 will be subtracted from D00100 as 4-digit signed binary values and the result will be output to D00120.

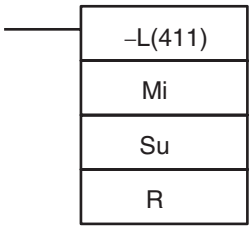


3-10-10 DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY: –L(411)

Purpose

Subtracts 8-digit (double-word) hexadecimal data and/or constants.

Ladder Symbol



Mi: 1st minuend word

Su: 1st subtrahend word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	–L(411)
	Executed Once for Upward Differentiation	@–L(411)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

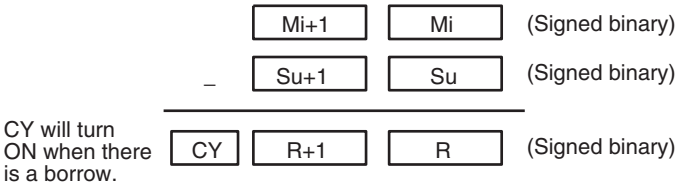
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)		---
Data Resisters	---		
Index Registers	IR0 or IR15		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

–L(411) subtracts the binary values in Su and Su+1 from Mi and Mi+1 and outputs the result to R, R+1. When the result is negative, it is output to R and R+1 as a 2's complement.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000 0000. OFF in all other cases.
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.
Overflow Flag	OF	ON when the result of subtracting a negative number from a positive number is in the range 8000 0000 to FFFF FFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the result of subtracting a positive number from a negative number is in the range 0000 0000 to 7FFF FFFF hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

When –L(411) is executed, the Error Flag will turn OFF.

If as a result of the subtraction, the content of R, R+1 is 0000 0000 hex, the Equals Flag will turn ON.

If the subtraction results in a borrow, the Carry Flag will turn ON.

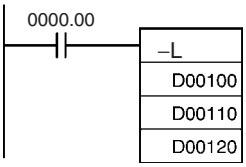
If the result of subtracting a negative number from a positive number is negative (in the range 8000 0000 to FFFF FFFF hex), the Overflow Flag will turn ON.

If the result of subtracting a positive number from a negative number is positive (in the range 0000 0000 to 7FFF FFFF hex), the Underflow Flag will turn ON.

If as a result of the subtraction, the content of the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00111 and D00110 will be subtracted from D00101 and D00100 as 8-digit signed binary values and the result will be output to D00121 and D00120.



If the result of a subtraction is a negative number (Mi<Su for –(410) or Mi+1, Mi <Su+1, Su for –L(411)), the result is output as the 2's complement and the Carry Flag (CY) will turn ON to indicate that the result of the subtraction is negative. To convert the 2's complement to the true number, an instruction which subtracts the result from 0000 or 0000 0000 is necessary using the Carry Flag (CY) as an execution condition.

Note 2's Complement

A 2's complement is the value obtained by subtracting each binary digit from 1 and adding one to the result. For example, the 2's complement for the binary number 1101 is calculated as follows: 1111 (F hexadecimal) – 1101 (D hexadecimal) + 1 (1 hexadecimal) = 0011 (3 hexadecimal). The 2's complement for 3039 (hexadecimal) is calculated as follows: FFFF (hexadecimal) – 3039 (hexadecimal) + 0001 (hexadecimal) = CFC7 (hexadecimal). Therefore, in case of 4-digit hexadecimal value, the 2's complement can be calculated as follows: FFFF (hexadecimal) – a (hexadecimal) + 0001 (hexadecimal) = b (hexadecimal). To obtain the true number "a" from the 2's complement b (hexadecimal): a (hexadecimal) = 10000 (hexadecimal) – b (hexadecimal). For example, to obtain the true number "a" from the 2's complement CFC7 (hexadecimal): 10000 (hexadecimal) – CFC7 = 3039.

Example 1

	Signed data	Unsigned data
FFFF hex →	–1	65535
–) 0001 hex →	–) +1	–) 1
FFFE hex →	–2 Note 1	65534 Note 2
Negative Flag ON		
Carry Flag OFF		

Example 2

	Signed data	Unsigned data
FFFD hex →	–3	65533
–)FFFF hex →	–) –1	–) 65535
FFFE hex →	–2 Note 3	65534 Note 4
Negative Flag ON		
Carry Flag ON		

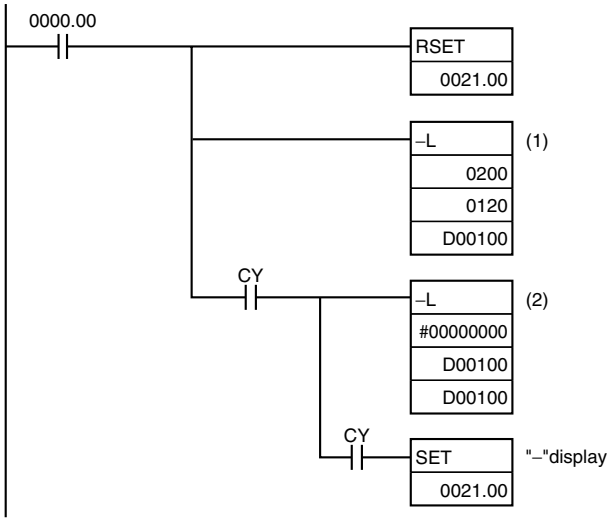
Note

1. Since the Negative Flag is ON, the result (FFFE hex) is a negative value (2's complement) and is thus –2.
2. Since the Carry Flag is OFF, the result (FFFE hex) is an unsigned positive value of 65534.
3. Since the Negative Flag is ON, the result (FFFE hex) is a negative value (2's complement) and is thus –2.
4. Since the Carry Flag is ON, the result (FFFE hex) is a negative value (2's complement) and becomes –2 when converted to a true value.

Program Example

20F55A10 – B8A360E3 = –97AE06D3.

In this example, the eight-digit binary value in CIO 0121 and CIO 0120 is subtracted from the value in CIO 0201 and CIO 0200, and the result is output in eight-digit binary to D00101 and D00100. If the result is negative, the instruction at (2) will be executed, and the actual result will then be output to D00101 and D00100.



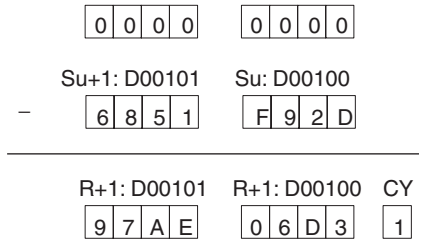
Subtraction at 1

	Mi+1: CIO 0201	Mi: CIO 0200
	2 0 F 5	5 A 1 0
	Su+1: CIO 0121	Su: CIO 0120
–	B 8 A 3	6 0 E 3

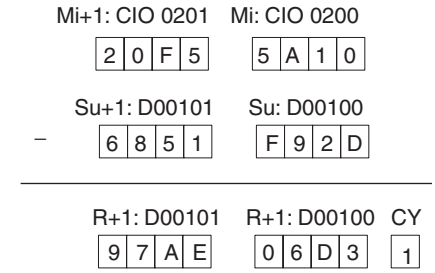
R+1: D00101	R+1: D00100	CY
6 8 5 1	F 9 2 D	1

The Carry Flag (CY) is ON, so the result is subtracted from 0000 0000 to obtain the actual number.

Subtraction at 2



Final Subtraction Result



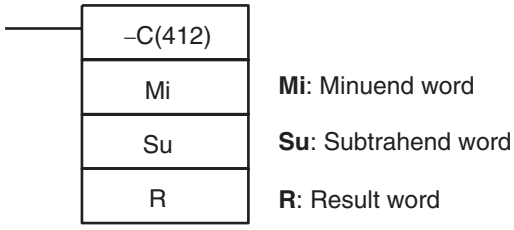
The Carry Flag (CY) is turned ON, so the actual number is –97AE06D3. Because the content of D00101 and D00100 is negative, CY is used to turn ON CIO 0021.00 to indicate this.

3-10-11 SIGNED BINARY SUBTRACT WITH CARRY: –C(412)

Purpose

Subtracts 4-digit (single-word) hexadecimal data and/or constants with the Carry Flag (CY).

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	–C(412)
	Executed Once for Upward Differentiation	@–C(412)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

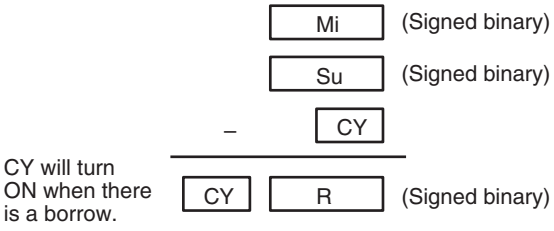
Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		

Area	Mi	Su	R
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)		---
Data Resisters	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

–C(412) subtracts the binary values in Su and CY from Mi, and outputs the result to R. When the result is negative, it is output to R as a 2's complement. (Refer to *3-10-12 DOUBLE SIGNED BINARY SUBTRACT WITH CARRY: –CL(413)* for an example of handling 2's complements.)



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the subtraction result is 0000. OFF in all other cases.
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.
Overflow Flag	OF	ON when the result of subtracting a negative number and CY from a positive number is in the range 8000 to FFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the result of subtracting a positive number and CY from a negative number is in the range 0000 to 7FFF hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

When –C(412) is executed, the Error Flag will turn OFF.

If as a result of the subtraction, the content of R is 0000 hex, the Equals Flag will turn ON.

If the subtraction results in a borrow, the Carry Flag will turn ON.

If the result of subtracting a negative number and CY from a positive number is negative (in the range 8000 to FFFF hex), the Overflow Flag will turn ON.

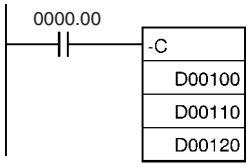
If the result of subtracting a positive number and CY from a negative number is positive (in the range 0000 to 7FFF hex), the Underflow Flag will turn ON.

If as a result of the subtraction, the content of the leftmost bit of R is 1, the Negative Flag will turn ON.

Note To clear the Carry Flag (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 0000.00 is ON in the following example, D00110 and CY will be subtracted from D00100 as 4-digit signed binary values and the result will be output to D00120.

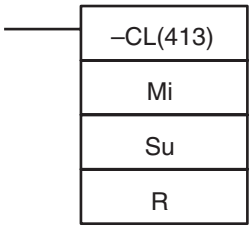


3-10-12 DOUBLE SIGNED BINARY SUBTRACT WITH CARRY: –CL(413)

Purpose

Subtracts 8-digit (double-word) hexadecimal data and/or constants with the Carry Flag (CY).

Ladder Symbol



Mi: 1st minuend word

Su: 1st subtrahend word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	–CL(413)
	Executed Once for Upward Differentiation	@–CL(413)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

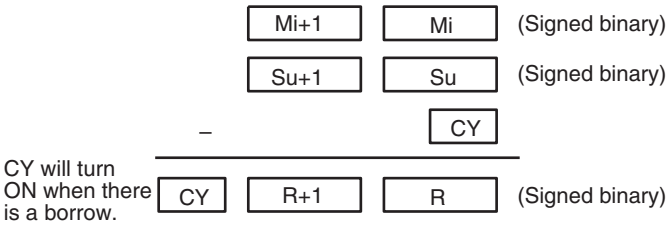
Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)		---
Data Resisters	---		

Area	Mi	Su	R
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

–CL(413) subtracts the binary values in Su and Su+1 and CY from Mi and Mi+1, and outputs the result to R, R+1. When the result is negative, it is output to R, R+1 as a 2's complement.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000 0000. OFF in all other cases.
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.
Overflow Flag	OF	ON when the result of subtracting a negative number and CY from a positive number is in the range 8000 0000 to FFFF FFFF hex. OFF in all other cases.
Underflow Flag	UF	ON when the result of subtracting a positive number and CY from a negative number is in the range 0000 0000 to 7FFF FFFF hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

When –CL(413) is executed, the Error Flag will turn OFF.

If as a result of the subtraction, the content of R, R+1 is 0000 0000 hex, the Equals Flag will turn ON.

If the subtraction results in a borrow, the Carry Flag will turn ON.

If the result of subtracting a negative number and CY from a positive number is negative (in the range 8000 0000 to FFFF FFFF hex), the Overflow Flag will turn ON.

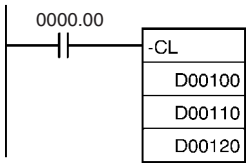
If the result of subtracting a positive number and CY from a negative number is positive (in the range 0000 0000 to 7FFF FFFF hex), the Underflow Flag will turn ON.

If as a result of the subtraction, the content of the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Note To clear the Carry Flag (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 0000.00 is ON in the following example, D00111, D00110 and CY will be subtracted from D00101 and D00100 as 8-digit signed binary values, and the result will be output to D00121 and D00120.



If the result of the subtraction is a negative number (Mi<Su for –C(412) or Mi+1, Mi <Su+1, Su for –CL(413)), the result is output as a 2's complement. The Carry Flag (CY) will turn ON. To convert the 2's complement to the true number, a program which subtracts the result from 0000 or 0000 0000 is necessary, using the Carry Flag (CY) as the execution condition. The Carry Flag turning ON thus indicates that the result of the subtraction is negative.

Note 2's Complement

A 2's complement is the value obtained by subtracting each binary digit from 1 and adding one to the result.

Example: The 2's complement for the binary number 1101 is as follows:

1111 (F hex) – 1101 (D hex) + 1 (1 hex) = 0011 (3 hex).

Example: The 2's complement for the 4-digit hexadecimal number 3039 is as follows:

FFFF hex – 3039 hex + 0001 hex = CFC7 hex.

Accordingly, the 2's complement for the 4-digit hexadecimal number "a" is as follows:

FFFF hex – a hex + 0001 hex = b hex.

And to obtain the true number "a" hex from the 2's complement "b" hex:

a hex = 10000 hex – b hex.

Example: To obtain the true number from the 2's complement CFC7 hex:

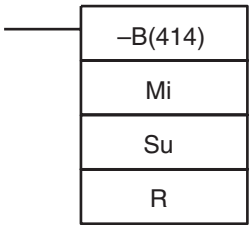
10000 hex – CFC7 hex = 3039 hex.

3-10-13 BCD SUBTRACT WITHOUT CARRY: –B(414)

Purpose

Subtracts 4-digit (single-word) BCD data and/or constants.

Ladder Symbol



Mi: Minuend word

Su: Subtrahend word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	–B(414)
	Executed Once for Upward Differentiation	@–B(414)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

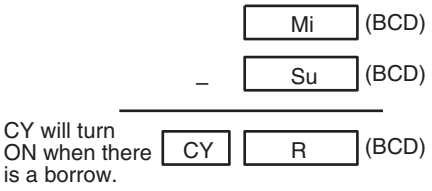
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #9999 (BCD)		---
Data Resisters	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

–B(414) subtracts the BCD values in Su from Mi and outputs the result to R. If the result of the subtraction is negative, the result is output to R as a 10's complement. (Refer to *3-10-14 DOUBLE BCD SUBTRACT WITHOUT CARRY: –BL(415)* for an example of handling 10's complements.)



Flags

Name	Label	Operation
Error Flag	ER	ON when Mi is not BCD. ON when Su is not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0000. OFF in all other cases.
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.

Precautions

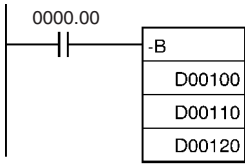
If Mi and/or Su are not BCD, an error is generated and the Error Flag will turn ON.

If as a result of the subtraction, the content of R is 0000 hex, the Equals Flag will turn ON.

If a subtraction results in a borrow, the Carry Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00110 is subtracted from D00100 as 4-digit BCD values, and the result will be output to D00120.

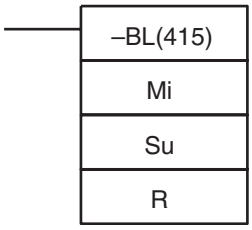


3-10-14 DOUBLE BCD SUBTRACT WITHOUT CARRY: –BL(415)

Purpose

Subtracts 8-digit (double-word) BCD data and/or constants.

Ladder Symbol



Mi: 1st minuend word

Su: 1st subtrahend word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	–BL(415)
	Executed Once for Upward Differentiation	@–BL(415)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

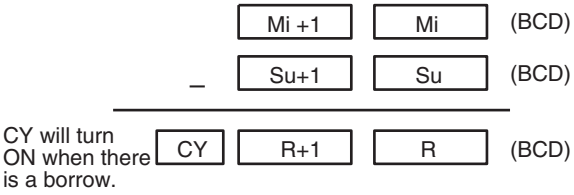
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #9999 9999 (BCD)		---
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

–BL(415) subtracts the BCD values in Su and Su+1 from Mi and Mi+1 and outputs the result to R, R+1. If the result is negative, it is output to R, R+1 as a 10's complement.



Flags

Name	Label	Operation
Error Flag	ER	ON when Mi and/or Mi +1 are not BCD. ON when Su and/or Su +1 are not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0000 0000. OFF in all other cases.
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.

Precautions

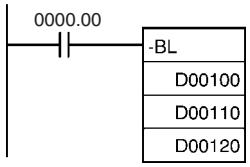
If Mi, Mi +1 and/or Su, Su +1 are not BCD, an error is generated and the Error Flag will turn ON.

If as a result of the subtraction, the content of R, R +1 is 0000 0000 hex, the Equals Flag will turn ON.

If a subtraction results in a borrow, the Carry Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00111 and D00110 will be subtracted from D00101 and D00100 as 8-digit BCD values, and the result will be output to D00121 and D00120.



If the result of the subtraction is a negative number (Mi<Su for –B(414) or Mi+1, Mi <Su+1, Su for –BL(415)), the result is output as a 10's complement. The Carry Flag (CY) will turn ON. To convert the 10's complement to the true number, a program which subtracts the result from 0 is necessary, using the Carry Flag (CY) as the execution condition. The Carry Flag turning ON thus indicates that the result of the subtraction is negative.

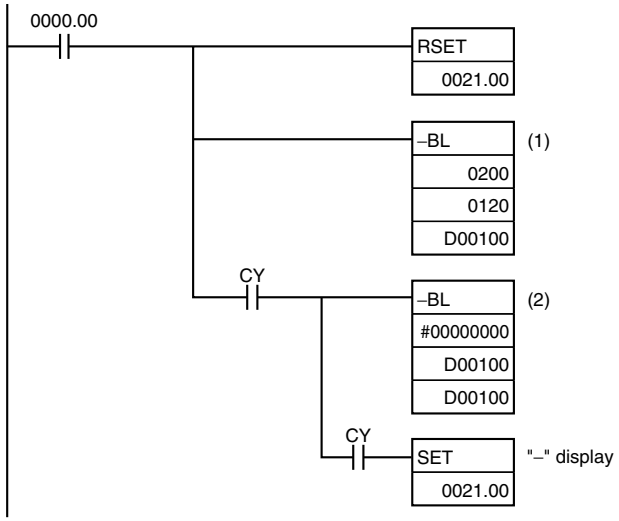
Note 10's Complement

A 10's complement is the value obtained by subtracting each digit from 9 and adding one to the result. For example, the 10's complement for 7556 is calculated as follows: 9999 – 7556 + 1 = 2444. For a four digit number, the 10's complement of A is 9999 – A + 1 = B. To obtain the true number from the 10's complement B: A = 10000 – B. For example, to obtain the true number from the 10's complement 2444: 10000 – 2444 = 7556.

Program Example

9,583,960 – 17,072,641 = –7,488,681.

In this example, the eight-digit BCD content of CIO 0121 and CIO 0120 is subtracted from the content of CIO 0201 and CIO 0200, and the result is output in eight-digit BCD to D00101 and D00100. The result is negative, so the instruction at (2) will be executed, and the true number will then be output to D00101 and D00100.



Subtraction at 1

Mi+1: CIO 0201 Mi: CIO 0200

0 9 5 8 | 3 9 6 0

Su+1: CIO 0121 Su: CIO 0120

– 1 7 0 7 | 2 6 4 1

09583960 + (100000000 – 17072641)

R+1: D00101 R+1: D00100 CY

→ 9 2 5 1 | 1 3 1 9 | 1

The Carry Flag (CY) is ON, so the result is subtracted from 0000 0000.

Subtraction at 2

0 0 0 0 | 0 0 0 0

Su+1: D00101 Su: D00100

– 9 2 5 1 | 1 3 1 9

00000000 + (100000000 – 92511319)

R+1: D00101 R+1: D00100 CY

→ 0 7 4 8 | 8 6 8 1 | 1

Final Subtraction Result

Mi+1: CIO 0201 Mi: CIO 0200

0 9 5 8 | 3 9 6 0

Su+1: CIO 0121 Su: CIO 0120

– 1 7 0 7 | 2 6 4 1

CY R+1: D00101 R+1: D00100

1 | 0 7 4 8 | 8 6 8 1

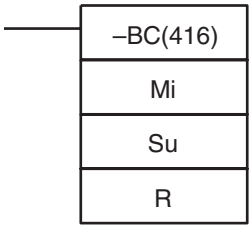
The Carry Flag (CY) will be turned ON, so the actual number is –7,488,681. Because the content of D00101 and D00100 is negative, CY is used to turn ON CIO 0021.00 to indicate this.

3-10-15 BCD SUBTRACT WITH CARRY: –BC(416)

Purpose

Subtracts 4-digit (single-word) BCD data and/or constants with the Carry Flag (CY).

Ladder Symbol



Mi: Minuend word

Su: Subtrahend word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	–BC(416)
	Executed Once for Upward Differentiation	@–BC(416)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

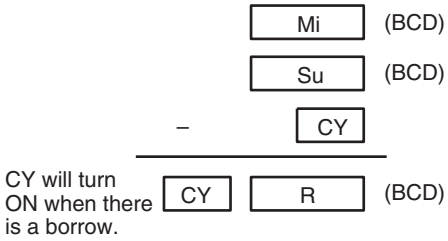
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #9999 (BCD)		---
Data Resisters	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

–BC(416) subtracts BCD values in Su and CY from Mi and outputs the result to R. If the result is negative, it is output to R as a 10's complement. (Refer to *3-10-16 DOUBLE BCD SUBTRACT WITH CARRY: –BCL(417)* for an example of handling 10's complements.)



Flags

Name	Label	Operation
Error Flag	ER	ON when Mi is not BCD. ON when Su is not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0000. OFF in all other cases.
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.

Precautions

If Mi and/or Su are not BCD, an error is generated and the Error Flag will turn ON.

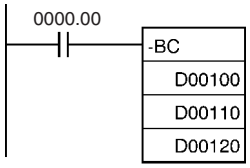
If as a result of the subtraction, the content of R is 0000 hex, the Equals Flag will turn ON.

If a subtraction results in a borrow, the Carry Flag will turn ON.

Note To clear the Carry Flag (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 0000.00 is ON in the following example, D00110 and CY will be subtracted from D00100 as 4-digit BCD values, and the result will be output to D00120.

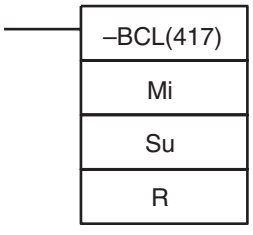


3-10-16 DOUBLE BCD SUBTRACT WITH CARRY: –BCL(417)

Purpose

Subtracts 8-digit (double-word) BCD data and/or constants with the Carry Flag (CY).

Ladder Symbol



Mi: 1st minuend word

Su: 1st subtrahend word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	–BCL(417)
	Executed Once for Upward Differentiation	@–BCL(417)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

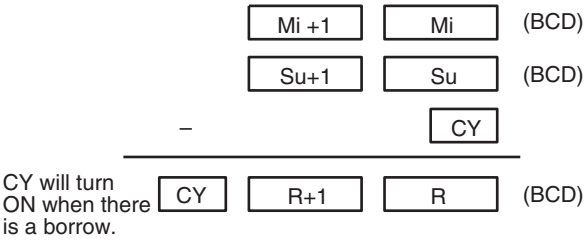
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #9999 9999 (BCD)		---
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

–BCL(417)subtracts the BCD values in Su, Su+1, and CY from Mi and Mi+1 and outputs the result to R, R+1. If the result is negative, it is output to R, R+1 as a 10's complement.



Flags

Name	Label	Operation
Error Flag	ER	ON when Mi and/or Mi +1 are not BCD. ON when Su and/or Su +1 are not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0000 0000. OFF in all other cases.
Carry Flag	CY	ON when the subtraction results in a borrow. OFF in all other cases.

Precautions

If Mi, Mi +1 and/or Su, Su +1 are not BCD, an error is generated and the Error Flag will turn ON.

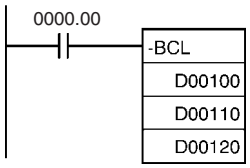
If as a result of the subtraction, the content of R, R +1 is 0000 0000 hex, the Equals Flag will turn ON.

If an subtraction results in a borrow, the Carry Flag will turn ON.

Note To clear the Carry Flag (CY), execute the Clear Carry (CLC(041)) instruction.

Examples

When CIO 0000.00 is ON in the following example, D00111, D00110, and CY will be subtracted from D00101 and D00100 as 8-digit BCD values, and the result will be output to D00121 and D00120.



If the result of the subtraction is a negative number (Mi<Su for –BC(416) or Mi+1, Mi <Su+1, Su for –BCL(417)), the result is output as a 10's complement. The Carry Flag (CY) will turn ON. To convert the 10's complement to the true number, a program which subtracts the result from 0 is necessary, using the Carry Flag (CY) as the execution condition. The Carry Flag turning ON thus indicates that the result of the subtraction is negative.

Note 10's Complement

A 10's complement is the value obtained by subtracting each digit from 9 and adding one to the result. For example, the 10's complement for 7556 is calculated as follows: 9999 – 7556 + 1 = 2444. For a four digit number, the 10's complement of A is 9999 – A + 1 = B. To obtain the true number from the 10's complement B: A = 10000 – B. For example, to obtain the true number from the 10's complement 2444: 10000 – 2444 = 7556.

3-10-17 SIGNED BINARY MULTIPLY: *(420)

Purpose

Multiplies 4-digit signed hexadecimal data and/or constants.

Ladder Symbol

Md: Multiplicand word

Mr: Multiplier word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	*(420)
	Executed Once for Upward Differentiation	@*(420)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

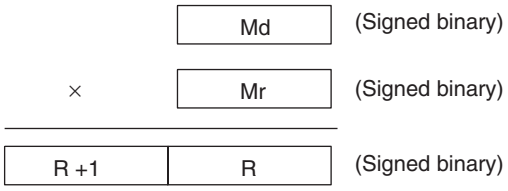
Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6142
Work Area	W000 to W255		W000 to W254
Auxiliary Bit Area	A000 to A959		A448 to A958

Area	Md	Mr	R
Timer Area	T0000 to T0255		T0000 to T0254
Counter Area	C0000 to C0255		C0000 to C0254
DM Area	D00000 to D32767		D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)		---
Data Resisters	DR0 to DR15		---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

*(420) multiplies the signed binary values in Md and Mr and outputs the result to R, R+1.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000 0000. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

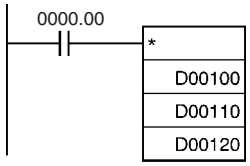
When *(420) is executed, the Error Flag will turn OFF.

If as a result of the multiplication, the content of R, R+1 is 0000 0000 hex, the Equals Flag will turn ON.

If as a result of the multiplication, the content of the leftmost bit of R+1 and R is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00100 and D00110 will be multiplied as 4-digit signed hexadecimal values and the result will be output to D00120.

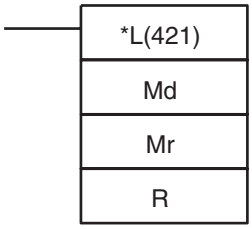


3-10-18 DOUBLE SIGNED BINARY MULTIPLY: *L(421)

Purpose

Multiplies 8-digit signed hexadecimal data and/or constants.

Ladder Symbol



Md: 1st multiplicand word

Mr: 1st multiplier word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	*L(421)
	Executed Once for Upward Differentiation	@*L(421)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

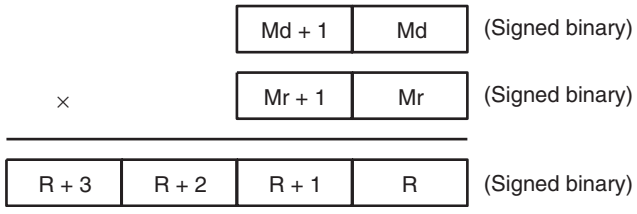
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0000 to CIO 6142		CIO 0000 to CIO 6140
Work Area	W000 to W254		W000 to W252
Auxiliary Bit Area	A000 to A958		A448 to A956
Timer Area	T0000 to T0254		T0000 to T0252
Counter Area	C0000 to C0254		C0000 to C0252
DM Area	D00000 to D32766		D00000 to D32764
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)		---
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

*L(421) multiplies the signed binary values in Md and Md+1 and in Mr and Mr+1 and outputs the result to R, R+1, R+2, and R+3.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0 (all 16 bits). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

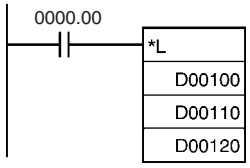
When *L(421) is executed, the Error Flag will turn OFF.

If as a result of the multiplication, the content of R, R+1, R+2, R+3 is 0000 0000 0000 0000 hex, the Equals Flag will turn ON.

If as a result of the multiplication, the content of the leftmost bit of R+3 is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00101, D00100, D00111, and D00110 will be multiplied as 8-digit signed hexadecimal values and the result will be output to D00123, D00122, D00121, and D00120.

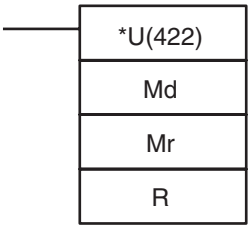


3-10-19 UNSIGNED BINARY MULTIPLY: *U(422)

Purpose

Multiplies 4-digit unsigned hexadecimal data and/or constants.

Ladder Symbol



Md: Multiplicand word

Mr: Multiplier word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	*U(422)
	Executed Once for Upward Differentiation	@*U(422)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6142
Work Area	W000 to W255		W000 to W254
Auxiliary Bit Area	A000 to A959		A448 to A958
Timer Area	T0000 to T0255		T0000 to T0254
Counter Area	C0000 to C0255		C0000 to C0254

Area	Md	Mr	R
DM Area	D00000 to D32767		D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)		---
Data Resisters	DR0 to DR15		---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

*U(420) multiplies the unsigned binary values in Md and Mr and outputs the result to R, R+1.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000 0000. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

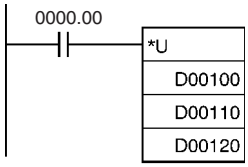
When *U(422) is executed, the Error Flag will turn OFF.

If as a result of the multiplication, the content of R, R+1 is 0000 0000 hex, the Equals Flag will turn ON.

If as a result of the multiplication, the content of the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00100 and D00110 will be multiplied as 4-digit unsigned binary values and the result will be output to D00121 and D00120.

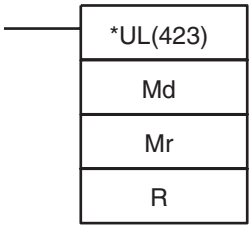


3-10-20 DOUBLE UNSIGNED BINARY MULTIPLY: *UL(423)

Purpose

Multiplies 8-digit unsigned hexadecimal data and/or constants.

Ladder Symbol



Md: 1st multiplicand word

Mr: 1st multiplier word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	*UL(423)
	Executed Once for Upward Differentiation	@*UL(423)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

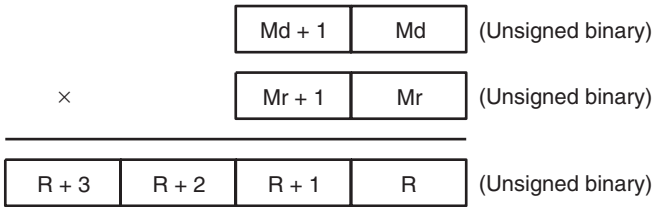
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0000 to CIO 6142		CIO 0000 to CIO 6140
Work Area	W000 to W254		W000 to W252
Auxiliary Bit Area	A000 to A958		A448 to A956
Timer Area	T0000 to T0254		T0000 to T0252
Counter Area	C0000 to C0254		C0000 to C0252
DM Area	D00000 to D32766		D00000 to D32764
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)		---
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

*UL(423) multiplies the unsigned binary values in Md and Md+1 and in Mr and Mr+1 and outputs the result to R, R+1, R+2, and R+3.



Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0 (all 16 bits). OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the result is 1. OFF in all other cases.

Precautions

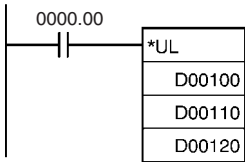
When *UL(423) is executed, the Error Flag will turn OFF.

If as a result of the multiplication, the content of R, R+1, R+2, R+3 is 0000 0000 0000 0000 hex, the Equals Flag will turn ON.

If as a result of the multiplication, the content of the leftmost bit of R+3 is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00101, D00100, D00111, and D00110 will be multiplied as 8-digit unsigned binary values and the result will be output to D00123, D00122, D00121, and D00120.



3-10-21 BCD MULTIPLY: *B(424)

Purpose

Multiplies 4-digit (single-word) BCD data and/or constants.

Ladder Symbol

Md: Multiplicand word

Mr: Multiplier word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	*B(424)
	Executed Once for Upward Differentiation	@*B(424)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6142
Work Area	W000 to W255		W000 to W254
Auxiliary Bit Area	A000 to A959		A448 to A958
Timer Area	T0000 to T0255		T0000 to T0254
Counter Area	C0000 to C0255		C0000 to C0254

Area	Md	Mr	R
DM Area	D00000 to D32767		D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #9999 (BCD)		---
Data Resisters	DR0 to DR15		---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

*B(424) multiplies the BCD content of Md and Mr and outputs the result to R, R+1.

Flags

Name	Label	Operation
Error Flag	ER	ON when Md is not BCD. ON when Mr is not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0000 0000. OFF in all other cases.

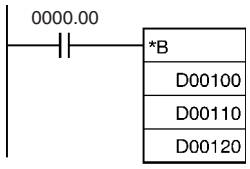
Precautions

If Md and/or Mr are not BCD, an error will be generated and the Error Flag will turn ON.

If as a result of the multiplication, the content of R, R+1 is 0000 0000 hex, the Equals Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00100 and D00110 will be multiplied as 4-digit BCD values and the result will be output to D00121 and D00120.



3-10-22 DOUBLE BCD MULTIPLY: *BL(425)

Purpose

Multiplies 8-digit (double-word) BCD data and/or constants.

Ladder Symbol

Md: 1st multiplicand word

Mr: 1st multiplier word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	*BL(425)
	Executed Once for Upward Differentiation	@*BL(425)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

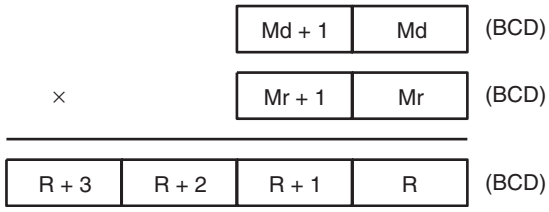
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0000 to CIO 6142		CIO 0000 to CIO 6140
Work Area	W000 to W254		W000 to W252
Auxiliary Bit Area	A000 to A958		A448 to A956
Timer Area	T0000 to T0254		T0000 to T0252
Counter Area	C0000 to C0254		C0000 to C0252
DM Area	D00000 to D32766		D00000 to D32764
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #9999 9999 (BCD)		---
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

*BL(425) multiplies BCD values in Md and Md+1 and in Mr and Mr+1 and outputs the result to R, R+1, R+2, and R+3.



Flags

Name	Label	Operation
Error Flag	ER	ON when Md and/or Md+1 are not BCD. ON when Mr and/or Mr +1 are not BCD. OFF in all other cases.
Equals Flag	=	ON when the result is 0 (all 16 bits). OFF in all other cases.

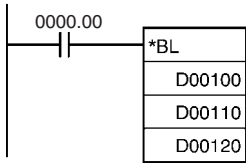
Precautions

If Md, Md+1 and/or Mr, Mr+1 are not BCD, an error will be generated and the Error Flag will turn ON.

If as a result of the multiplication, the content of R, R+1, R+2, R+3 is 0000 0000 0000 0000 hex, the Equals Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00101, D00100, D00111, and D00110 will be multiplied as 8-digit BCD values and the result will be output to D00123, D00122, D00121 and D00120.

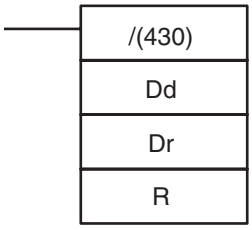


3-10-23 SIGNED BINARY DIVIDE: /(430)

Purpose

Divides 4-digit (single-word) signed hexadecimal data and/or constants.

Ladder Symbol



Dd: Dividend word

Dr: Divisor word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	/(430)
	Executed Once for Upward Differentiation	@/(430)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

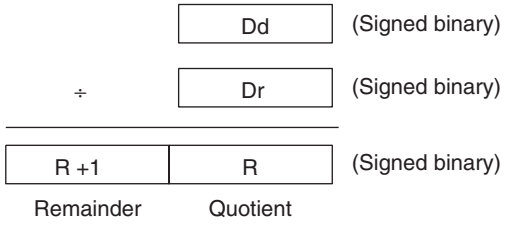
Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6142
Work Area	W000 to W255		W000 to W254
Auxiliary Bit Area	A000 to A959		A448 to A958
Timer Area	T0000 to T0255		T0000 to T0254
Counter Area	C0000 to C0255		C0000 to C0254
DM Area	D00000 to D32767		D00000 to D32766

Area	Dd	Dr	R
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)	#0001 to #FFFF (binary)	---
Data Resisters	DR0 to DR15		---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

/(430) divides the signed binary (16 bit) values in Dd by those in Dr and outputs the result to R, R+1. The quotient is placed in R and the remainder in R+1.



Flags

Name	Label	Operation
Error Flag	ER	ON when Dr is 0000. OFF in all other cases.
Equals Flag	=	ON when as a result of the division, R is 0000. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the R is 1. OFF in all other cases.

Precautions

Dividing 8000 hex by FFFF hex will produce an inconsistent result.

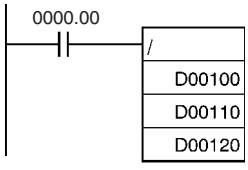
When the content of Dr is 0000 hex, an error will be generated and the Error Flag will turn ON.

If as a result of the division, the content of R is 0000 hex, the Equals Flag will turn ON.

If as a result of the division, the content of the leftmost bit of R is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00100 will be divided by D00110 as 4-digit signed binary values and the quotient will be output to D00120 and the remainder to D00121.

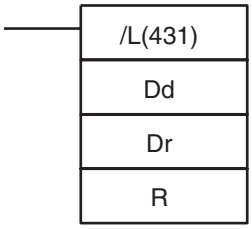


3-10-24 DOUBLE SIGNED BINARY DIVIDE: /L(431)

Purpose

Divides 8-digit (double-word) signed hexadecimal data and/or constants.

Ladder Symbol



Dd: 1st dividend word

Dr: 1st divisor word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	/L(431)
	Executed Once for Upward Differentiation	@/L(431)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

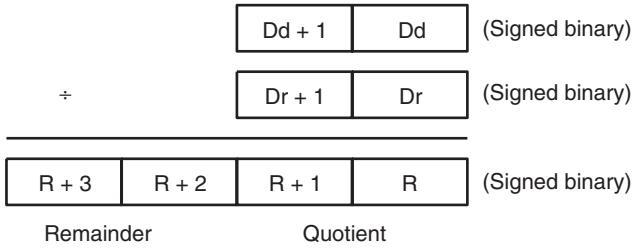
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0000 to CIO 6142		CIO 0000 to CIO 6140
Work Area	W000 to W254		W000 to W252
Auxiliary Bit Area	A000 to A958		A448 to A956
Timer Area	T0000 to T0254		T0000 to T0252
Counter Area	C0000 to C0254		C0000 to C0252
DM Area	D00000 to D32766		D00000 to D32764
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)	#0000 0001 to #FFFF FFFF (binary)	---
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

/L(431) divides the signed binary values in Dd and Dd+1 by those in Dr and Dr+1 and outputs the result to R, R+1, R+2, and R+3. The quotient is output to R and R+1 and the remainder is output to R+2 and R+3.



Flags

Name	Label	Operation
Error Flag	ER	ON when Dr and Dr+1 is 0000 0000. OFF in all other cases.
Equals Flag	=	ON when as a result of the division, R+1, R is 0000 0000. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the R+1 is 1. OFF in all other cases.

Precautions

Dividing 8000 0000 hex by FFFF FFFF hex will produce an inconsistent result.

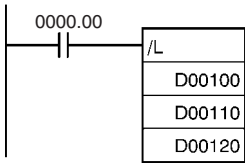
When the content of Dr+1 and Dr is 0000 0000, the Error Flag will turn ON.

If as a result of the division, the content of R+1, R is 0000 0000 hex, the Equals Flag will turn ON.

If as a result of the division, the content of the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00101 and D00100 are divided by D00111 and D00110 as 8-digit signed hexadecimal values and the quotient will be output to D00121 and D00120 and the remainder to D00123 and D00122.

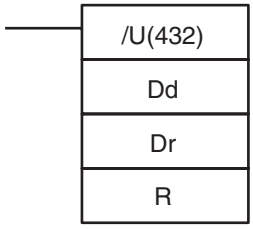


3-10-25 UNSIGNED BINARY DIVIDE: /U(432)

Purpose

Divides 4-digit (single-word) unsigned hexadecimal data and/or constants.

Ladder Symbol



Dd: Dividend word

Dr: Divisor word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	/U(432)
	Executed Once for Upward Differentiation	@/U(432)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

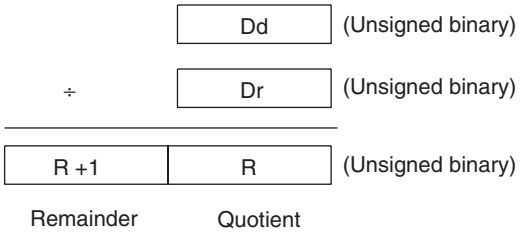
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6142
Work Area	W000 to W255		W000 to W254
Auxiliary Bit Area	A000 to A959		A448 to A958
Timer Area	T0000 to T0255		T0000 to T0254
Counter Area	C0000 to C0255		C0000 to C0254
DM Area	D00000 to D32767		D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)	#0001 to #FFFF (binary)	---
Data Resisters	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

/U(432) divides the unsigned binary values in Dd by those in Dr and outputs the quotient to R and the remainder to R+1.



Flags

Name	Label	Operation
Error Flag	ER	ON when Dr is 0000. OFF in all other cases.
Equals Flag	=	ON when as a result of the division, R is 0000 hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the R is 1. OFF in all other cases.

Precautions

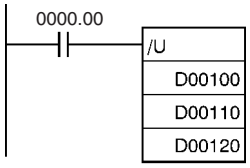
When the content of Dr is 0000 hex, the Error Flag will turn ON.

If as a result of the division, the content of R is 0000 hex, the Equals Flag will turn ON.

If as a result of the division, the content of the leftmost bit of R is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00100 will be divided by D00110 as 4-digit unsigned binary values and the quotient will be output to D00120 and the remainder will be output to D00121.

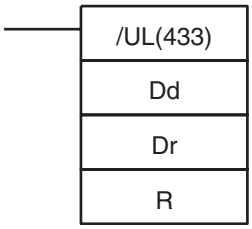


3-10-26 DOUBLE UNSIGNED BINARY DIVIDE: /UL(433)

Purpose

Divides 8-digit (double-word) unsigned hexadecimal data and/or constants.

Ladder Symbol



Dd: 1st dividend word

Dr: 1st divisor word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	/UL(433)
	Executed Once for Upward Differentiation	@/UL(433)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

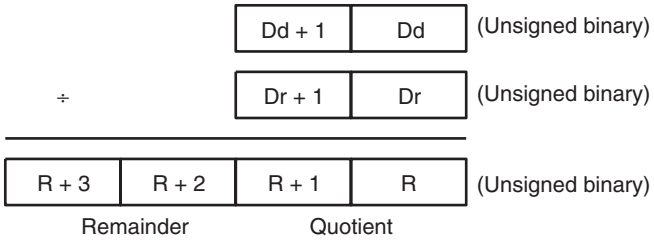
Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0000 to CIO 6142		CIO 0000 to CIO 6140
Work Area	W000 to W254		W000 to W252
Auxiliary Bit Area	A000 to A958		A448 to A956
Timer Area	T0000 to T0254		T0000 to T0252
Counter Area	C0000 to C0254		C0000 to C0252
DM Area	D00000 to D32766		D00000 to D32764
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)	#0000 0001 to #FFFF FFFF (binary)	---
Data Resisters	---		

Area	Dd	Dr	R
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

/UL(433) divides the unsigned binary values in Dd and Dd+1 by those in Dr and Dr+1 and outputs the quotient to R, R+1 and the remainder to R+2, and R+3.



Flags

Name	Label	Operation
Error Flag	ER	ON when Dr and Dr+1 is 0000 0000 hex. OFF in all other cases.
Equals Flag	=	ON when as a result of the division R+1, R is 0000 0000 hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of the R+1 is 1. OFF in all other cases.

Precautions

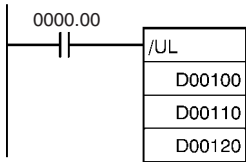
When the content of Dr, Dr+1 is 0000 0000 hex, the Error Flag will turn ON.

If as a result of the division, the content of R, R+1, is 0000 0000 hex, the Equals Flag will turn ON.

If as a result of the division, the content of the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00100 and D00101 will be divided by D00111 and D00110 as 8-digit unsigned hexadecimal values and the quotient will be output to D00121 and D00120 and the remainder to D00123 and D00122.

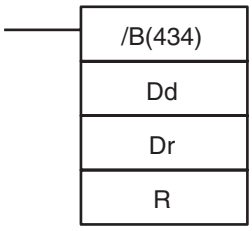


3-10-27 BCD DIVIDE: /B(434)

Purpose

Divides 4-digit (single-word) BCD data and/or constants.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	/B(434)
	Executed Once for Upward Differentiation	@/B(434)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

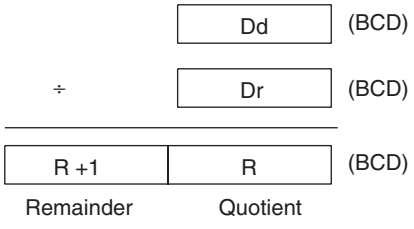
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0000 to CIO 6143		CIO 0000 to CIO 6142
Work Area	W000 to W255		W000 to W254
Auxiliary Bit Area	A000 to A959		A448 to A958
Timer Area	T0000 to T0255		T0000 to T0254
Counter Area	C0000 to C0255		C0000 to C0254
DM Area	D00000 to D32767		D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #9999 (BCD)	#0001 to #9999 (BCD)	---
Data Resisters	DR0 to DR15		---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

/B(434) divides the BCD content of Dd by that of Dr and outputs the quotient to R and the remainder to R+1.



Flags

Name	Label	Operation
Error Flag	ER	ON when Dd is not BCD. ON when Dr is not BCD. ON when Dr is 0000. OFF in all other cases.
Equals Flag	=	ON when R is 0000 hex. OFF in all other cases.

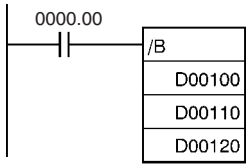
Precautions

If Dd or Dr is not BCD or if Dr is 0000, an error will be generated and the Error Flag will turn ON.

If as a result of the division, the content of R is 0000 hex, the Equals Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00100 will be divided by D00110 as 4-digit BCD values and the quotient will be output to D00120 and the remainder to D00121.

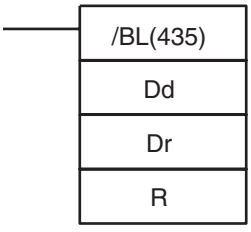


3-10-28 DOUBLE BCD DIVIDE: /BL(435)

Purpose

Divides 8-digit (double-word) BCD data and/or constants.

Ladder Symbol



Dd: 1st dividend word

Dr: 1st divisor word

R: 1st result word

Variations

Variations		
	Executed Each Cycle for ON Condition	/BL(435)
	Executed Once for Upward Differentiation	@/BL(435)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

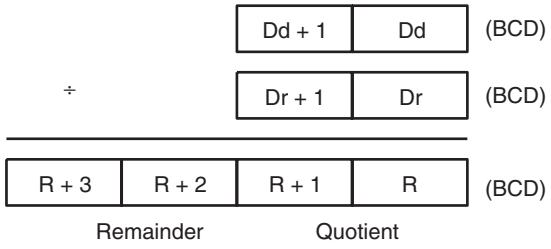
Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0000 to CIO 6142		CIO 0000 to CIO 6140
Work Area	W000 to W254		W000 to W252
Auxiliary Bit Area	A000 to A958		A448 to A956
Timer Area	T0000 to T0254		T0000 to T0252
Counter Area	C0000 to C0254		C0000 to C0252

Area	Dd	Dr	R
DM Area	D00000 to D32766		D00000 to D32764
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #9999 9999 (BCD)	#0000 0001 to #9999 9999 (BCD)	---
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

/BL(435) divides BCD values in Dd and Dd+1 by those in Dr and Dr+1 and outputs the quotient to R, R+1 and the remainder to R+2, R+3.



Flags

Name	Label	Operation
Error Flag	ER	ON when Dd, Dd+1 is not BCD. ON when Dr, Dr +1 is not BCD. ON when the content of Dr+1 and Dr is 0000 0000 hex. OFF in all other cases.
Equals Flag	=	ON when as the result of a division R+1 and R is 0000 0000. OFF in all other cases.

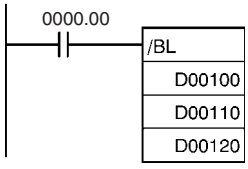
Precautions

If Dd, Dd+1 and/or Dr, Dr+1 are not BCD or the content of Dr, Dr+1 is 0000 0000 hex, an error will be generated and the Error Flag will turn ON.

If as a result of the division, the content of R, R+1 is 0000 0000 hex, the Equals Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, D00101 and D00100 will be divided by D00111 and D00110 as 8-digit BCD values and the quotient will be output to D00121 and D00120 and the remainder to D00123 and D00122.



3-11 Conversion Instructions

This section describes instructions used for data conversion.

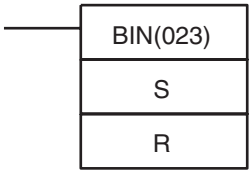
Instruction	Mnemonic	Function code	Page
BCD-TO-BINARY	BIN	023	331
DOUBLE BCD-TO-DOUBLE BINARY	BINL	058	333
BINARY-TO-BCD	BCD	024	334
DOUBLE BINARY-TO-DOUBLE BCD	BCDL	059	336
2'S COMPLEMENT	NEG	160	338
DOUBLE 2'S COMPLEMENT	NEGL	161	339
ASCII CONVERT	ASC	086	341
ASCII TO HEX	HEX	162	345
16-BIT TO 32-BIT SIGNED BINARY	SIGN	600	349

3-11-1 BCD-TO-BINARY: BIN(023)

Purpose

Converts BCD data to binary data.

Ladder Symbol



S: Source word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	BIN(023)
	Executed Once for Upward Differentiation	@BIN(023)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

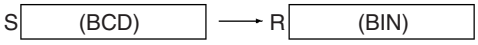
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	
Auxiliary Bit Area	A000 to A959	A448 to A959
Timer Area	T0000 to T0255	
Counter Area	C0000 to C0255	
DM Area	D00000 to D32767	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Resisters	DR0 to DR15	

Area	S	R
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

BIN(023) converts the BCD data in S to binary data and writes the result to R.

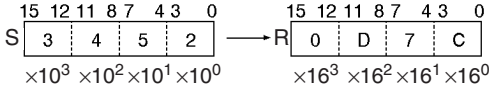


Flags

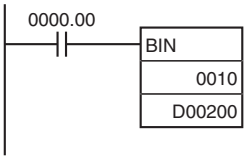
Name	Label	Operation
Error Flag	ER	ON if the content of S is not BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.
Negative Flag	N	OFF

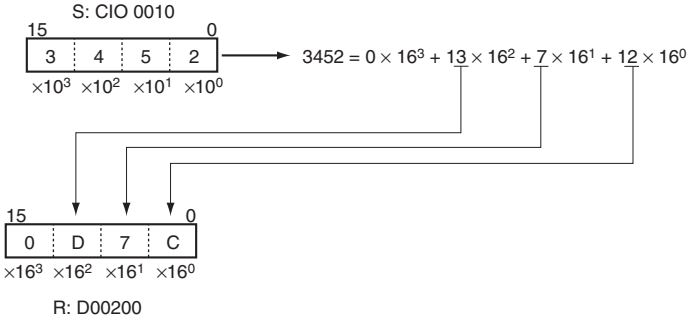
Example

The following diagram shows an example BCD-to-binary conversion.



When CIO 0000.00 is ON in the following example, the 4-digit BCD value in CIO 0010 is converted to hexadecimal and stored in D00200.



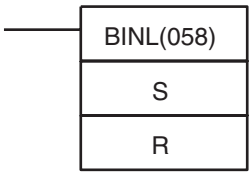


3-11-2 DOUBLE BCD-TO-DOUBLE BINARY: BINL(058)

Purpose

Converts 8-digit BCD data to 8-digit hexadecimal (32-bit binary) data.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	BINL(058)
	Executed Once for Upward Differentiation	@BINL(058)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

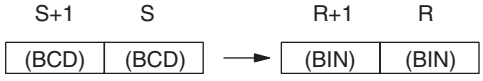
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

BINL(058) converts the 8-digit BCD data in S and S+1 to 8-digit hexadecimal (32-bit binary) data and writes the result to R and R+1.

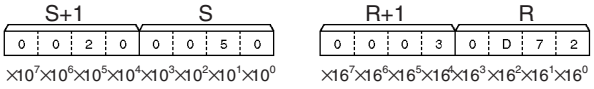


Flags

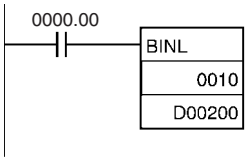
Name	Label	Operation
Error Flag	ER	ON if the content of S+1, S is not BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0000 0000 hex. OFF in all other cases.
Negative Flag	N	OFF

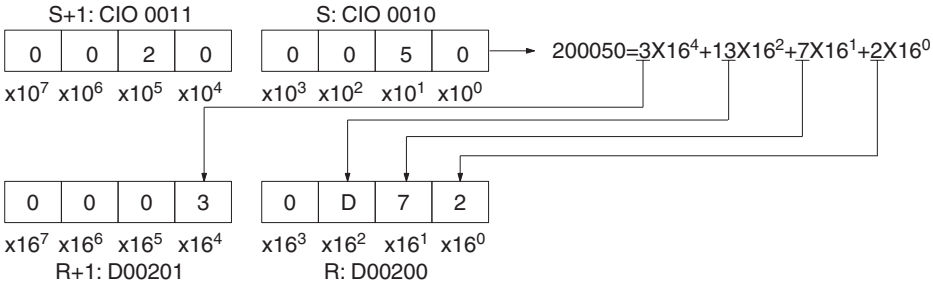
Examples

The following diagram shows an example of 8-digit BCD-to-binary conversion.



When CIO 0000.00 is ON in the following example, the 8-digit BCD value in CIO 0010 and CIO 0011 is converted to hexadecimal and stored in D00200 and D00201.



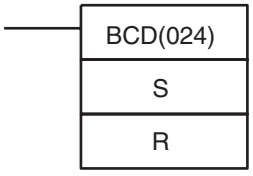


3-11-3 BINARY-TO-BCD: BCD(024)

Purpose

Converts a word of binary data to a word of BCD data.

Ladder Symbol



S: Source word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	BCD(024)
	Executed Once for Upward Differentiation	@BCD(024)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S: Source Word

S must be between 0000 and 270F hexadecimal (0000 and 9999 BCD).

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	
Auxiliary Bit Area	A000 to A959	A448 to A959
Timer Area	T0000 to T0255	
Counter Area	C0000 to C0255	
DM Area	D00000 to D32767	

Area	S	R
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Resisters	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

BCD(024) converts the binary data in S to BCD data and writes the result to R.

Flags

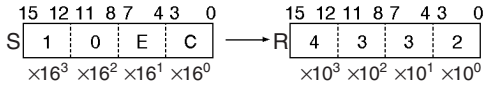
Name	Label	Operation
Error Flag	ER	ON if the content of S is not between 0000 to 270F hex (0 to 9999 decimal). OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.

Precautions

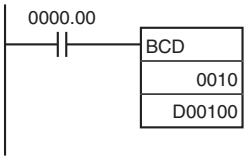
The content of S must be between 0000 to 270F hex (0000 to 9999 decimal).

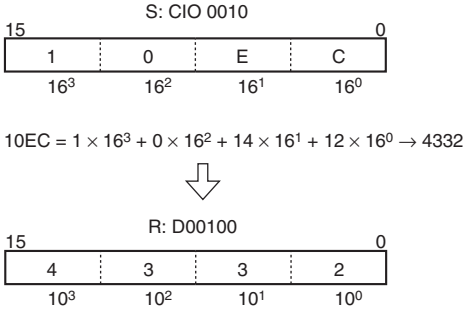
Example

The following diagram shows an example BCD-to-binary conversion.



When CIO 0000.00 is ON in the following example, the 4-digit hexadecimal value in CIO 0010 is converted to BCD and stored in D00100.



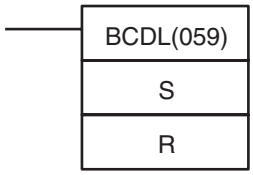


3-11-4 DOUBLE BINARY-TO-DOUBLE BCD: BCDL(059)

Purpose

Converts 8-digit hexadecimal (32-bit binary) data to 8-digit BCD data.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	BCDL(059)
	Executed Once for Upward Differentiation	@BCDL(059)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S: First Source Word

The content of S+1 and S must be between 0000 0000 and 05F5 E0FF hexadecimal (0000 0000 and 9999 9999 BCD).

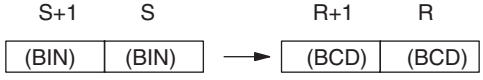
Note S and S+1, as well as D and D+1 must be in the same data area.

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

BCDL(059) converts the 8-digit hexadecimal (32-bit binary) data in S and S+1 to 8-digit BCD data and writes the result to R and R+1.



Flags

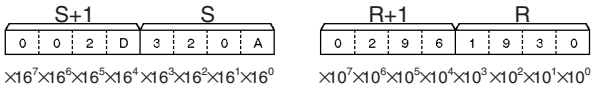
Name	Label	Operation
Error Flag	ER	ON if the content of S and S+1 is not between 0000 0000 to 05F5 E0FF hexadecimal (0000 0000 to 9999 9999 decimal). OFF in all other cases.
Equals Flag	=	ON if the result is 0000 0000. OFF in all other cases.

Precautions

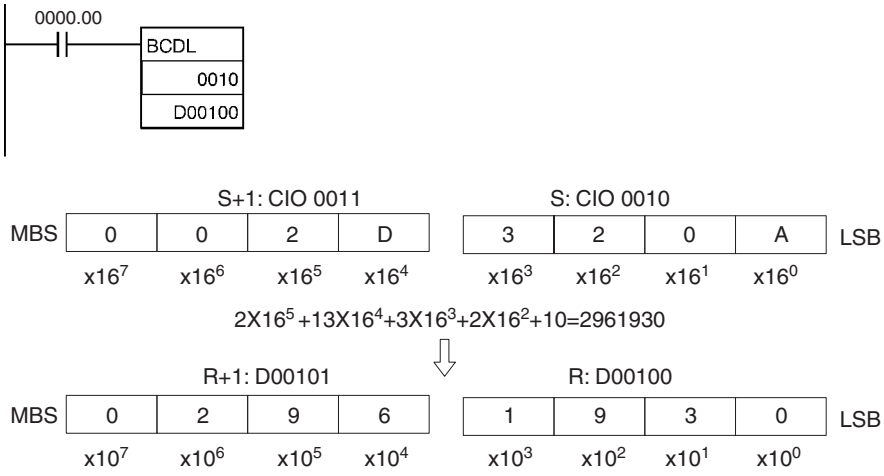
The content of S+1 and S must be between 0000 0000 to 05F5 E0FF hex (0000 0000 to 9999 9999 decimal).

Examples

The following diagram shows an example of 8-digit BCD-to-binary conversion.



When CIO 0000.00 is ON in the following example, the hexadecimal value in CIO 0011 and CIO 0010 is converted to a BCD value and stored in D00101 and D00100.

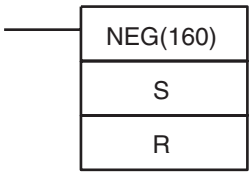


3-11-5 2'S COMPLEMENT: NEG(160)

Purpose

Calculates the 2's complement of a word of hexadecimal data.

Ladder Symbol



S: Source word

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	NEG(160)
	Executed Once for Upward Differentiation	@NEG(160)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

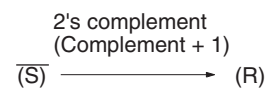
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6143	
Work Area	W000 to W255	
Auxiliary Bit Area	A000 to A959	A448 to A959
Timer Area	T0000 to T0255	
Counter Area	C0000 to C0255	
DM Area	D00000 to D32767	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 to #FFFF (binary)	---
Data Resisters	DR0 to DR15	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

NEG(160) calculates the 2's complement of S and writes the result to R. The 2's complement calculation basically reverses the status of the bits in S and adds 1.



Note This operation (reversing the status of the bits and adding 1) is equivalent to subtracting the content of S from 0000.

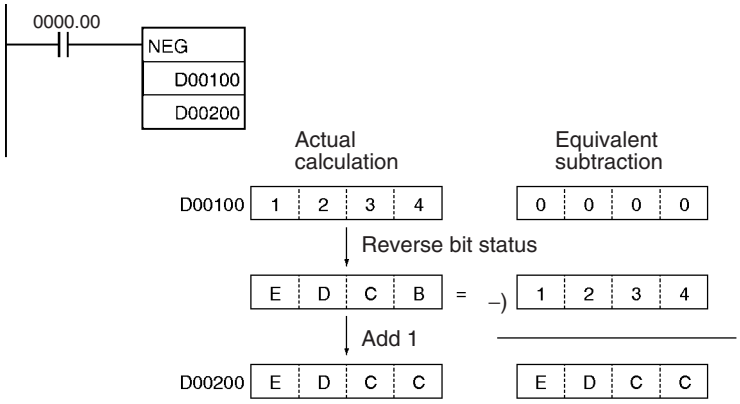
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result is 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 of the result is ON. OFF in all other cases.

Note The result for 8000 hex will be 8000 hex.

Example

When CIO 0000.00 is ON in the following example, NEG(160) calculates the 2's complement of the content of D00100 and writes the result to D00200.

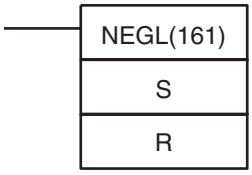


3-11-6 DOUBLE 2'S COMPLEMENT: NEGL(161)

Purpose

Calculates the 2's complement of two words of hexadecimal data.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	NEGL(161)
	Executed Once for Upward Differentiation	@NEGL(161)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

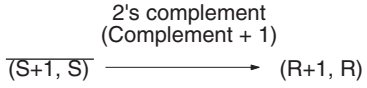
Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	

Area	S	R
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	---
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Note S and S+1 as well as R and R+1 must be in the same data area

Description

NEGL(161) calculates the 2's complement of S+1 and S and writes the result to R+1 and R. The 2's complement calculation basically reverses the status of the bits in S+1 and S and adds 1.



Note This operation (reversing the status of the bits and adding 1) is equivalent to subtracting the content of S+1 and S from 0000 0000.

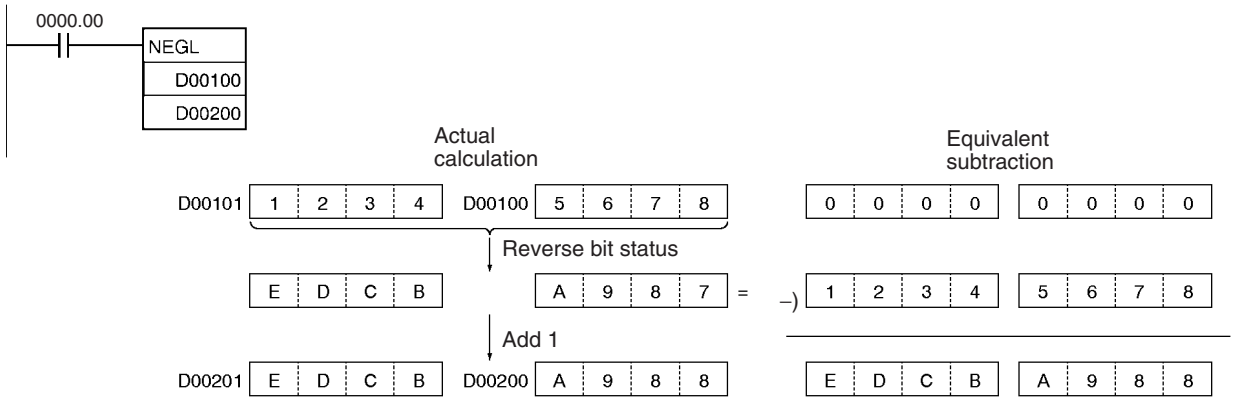
Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result is 0000 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 of R+1 is ON. OFF in all other cases.

Note The result for 8000 0000 hex will be 8000 0000 hex.

Example

When CIO 0000.00 is ON in the following example, NEGL(161) calculates the 2's complement of the content of D00101 and D00100 and writes the result to D00201 and D00200.

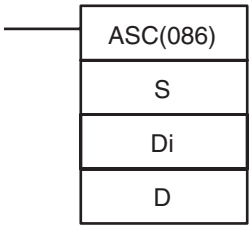


3-11-7 ASCII CONVERT: ASC(086)

Purpose

Converts 4-bit hexadecimal digits in the source word into their 8-bit ASCII equivalents. This instruction can be used only in the Coordinator Module.

Ladder Symbol



S: Source word

Di: Digit designator

D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	ASC(086)
	Executed Once for Upward Differentiation	@ASC(086)
	Executed Once for Downward Differentiation	Not supported

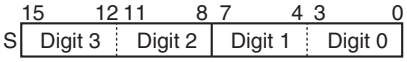
Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S: Source Word

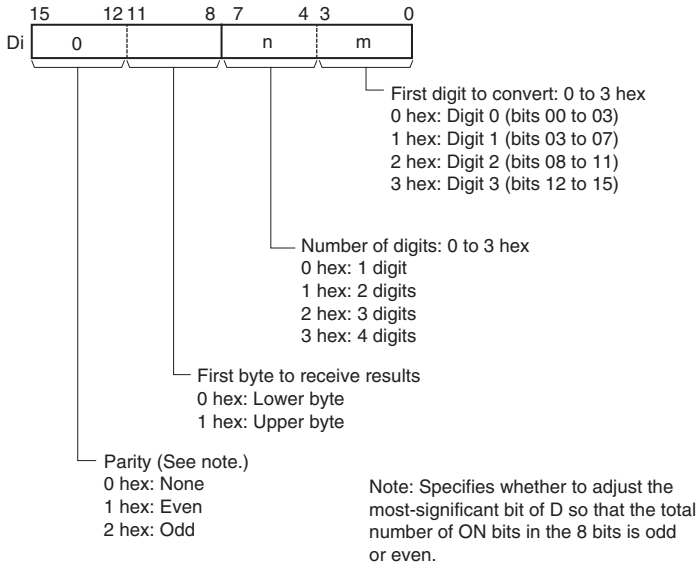
Up to four digits in the source word can be converted. The digits are numbered 0 to 3, right to left.

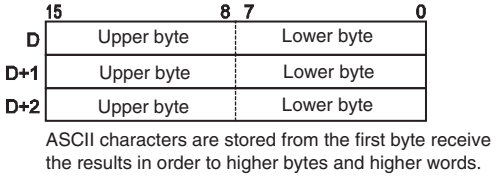


All data starting from the first digit to be converted is taken as hexadecimal data and converted to ASCII. Digit 0 follows digit 3.

Di: Digit Designator

The digit designator specifies various parameters for the conversion, as shown in the following diagram.





ASCII characters are stored from the first byte receive the results in order to higher bytes and higher words.

Note: D and D+2 must be in the same area.

D: First destination word

The converted ASCII data is written to the destination word(s) beginning with the specified byte in D. Three destination words (D to D+3) will be required if 4 digits are being converted and the leftmost byte is selected as the first byte in D. The destination words must be in the same data area.

Any bytes in the destination word(s) that are not overwritten with ASCII data will be left unchanged.

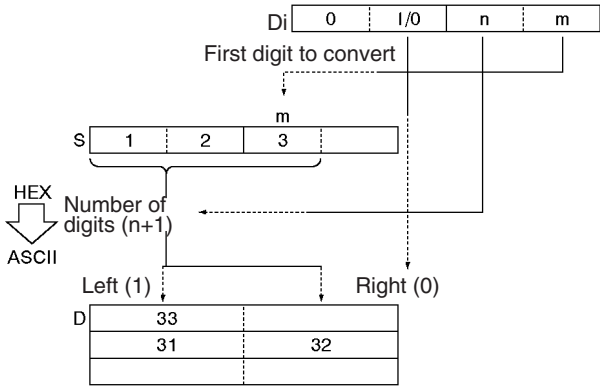
Operand Specifications

Area	S	Di	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---	Specified values only	---
Data Resisters	DR0 to DR15		---

Area	S	Di	D
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

ASC(086) treats the contents of S as 4 hexadecimal digits, converts the designated digit(s) of S into their 8-bit ASCII equivalents, and writes this data into the destination word(s) beginning with the specified byte in D.



Parity

It is possible to specify the parity of the ASCII data for use in error control during data transmissions. The leftmost bit of each ASCII character will be automatically adjusted for even, odd, or no parity.

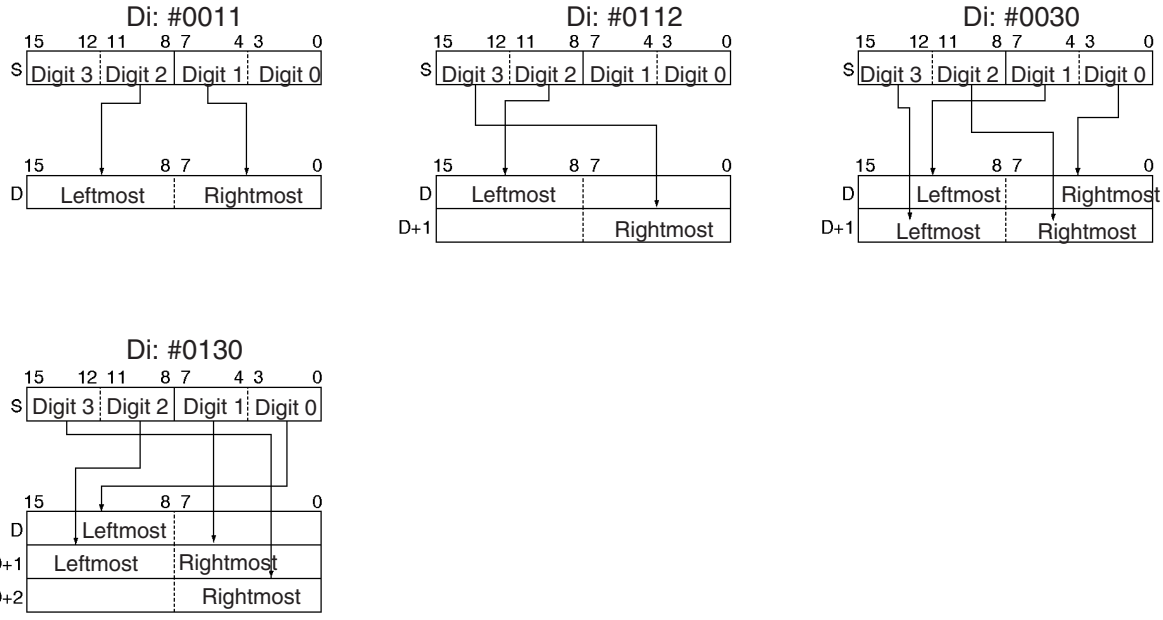
When no parity (0) is designated, the leftmost bit will always be zero. When even parity (1) is designated, the leftmost bit will be adjusted so that the total number of ON bits is even. When odd parity (2) is designated, the leftmost bit of each ASCII character will be adjusted so that there is an odd number of ON bits. The status of the parity bit does not affect the meaning of the ASCII code.

Examples of even parity:
When adjusted for even parity, ASCII “31” (00110001) will be “B1” (10110001: parity bit turned ON to create an even number of ON bits); ASCII “36” (00110110) will be “36” (00110110: parity bit remains OFF because the number of ON bits is already even).

Examples of odd parity:
When adjusted for odd parity, ASCII “36” (00110110) will be “B6” (10110110: parity bit turned ON to create an odd number of ON bits); ASCII “46” (01000110) will be “46” (01000110: parity bit remains OFF because the number of ON bits is already odd).

Examples of Di

When two or more digits are being converted, ASC(086) will read the bytes in S from right to left and will wrap around to the rightmost byte if necessary. The following diagram shows some example values for Di and the conversions that they produce.

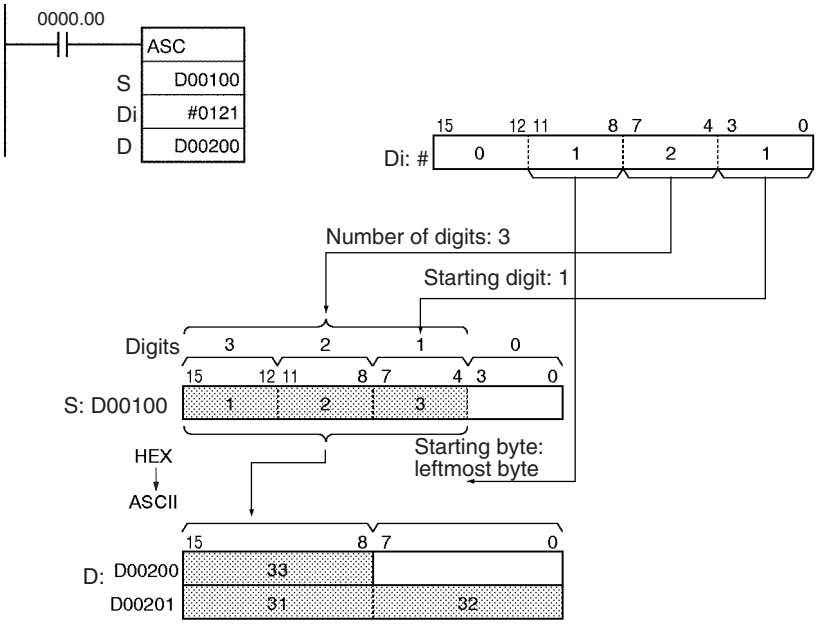


Flags

Name	Label	Operation
Error Flag	ER	ON if the content of Di is not within the specified ranges. OFF in all other cases.

Example

When CIO 0000.00 is ON in the following example, ASC(086) converts three hexadecimal digits in D00100 (beginning with digit 1) into their ASCII equivalents and writes this data to D00200 and D00201 beginning with the leftmost byte in D00200. In this case, a digit designator of #0121 specifies no parity, the starting byte (when writing) = leftmost byte, the number of digits to read = 3, and the starting digit (when reading) = digit 1.



ASCII Conversion Example

Contents of digit being converted					Converted (output) data								
Value	Bit status				Code	(MSB)	Bit status						(LSB)
0	0	0	0	0	30 hex	*	0	1	1	0	0	0	0
1	0	0	0	1	31 hex	*	0	1	1	0	0	0	1
2	0	0	1	0	32 hex	*	0	1	1	0	0	1	0
3	0	0	1	1	33 hex	*	0	1	1	0	0	1	1
4	0	1	0	0	34 hex	*	0	1	1	0	1	0	0
5	0	1	0	1	35 hex	*	0	1	1	0	1	0	1
6	0	1	1	0	36 hex	*	0	1	1	0	1	1	0
7	0	1	1	1	37 hex	*	0	1	1	0	1	1	1
8	1	0	0	0	38 hex	*	0	1	1	1	0	0	0
9	1	0	0	1	39 hex	*	0	1	1	1	0	0	1
A	1	0	1	0	41 hex	*	1	0	0	0	0	0	1
B	1	0	1	1	42 hex	*	1	0	0	0	0	1	0
C	1	1	0	0	43 hex	*	1	0	0	0	0	1	1
D	1	1	0	1	44 hex	*	1	0	0	0	1	0	0
E	1	1	1	0	45 hex	*	1	0	0	0	1	0	1
F	1	1	1	1	46 hex	*	1	0	0	0	1	1	0

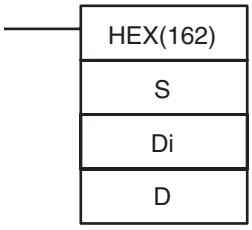
*: Parity bit: Depends on parity designation.

3-11-8 ASCII TO HEX: HEX(162)

Purpose

Converts up to 4 bytes of ASCII data in the source word to their hexadecimal equivalents and writes these digits in the specified destination word. This instruction can be used only in the Coordinator Module.

Ladder Symbol



S: First source word

Di: Digit designator

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	HEX(162)
	Executed Once for Upward Differentiation	@HEX(162)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

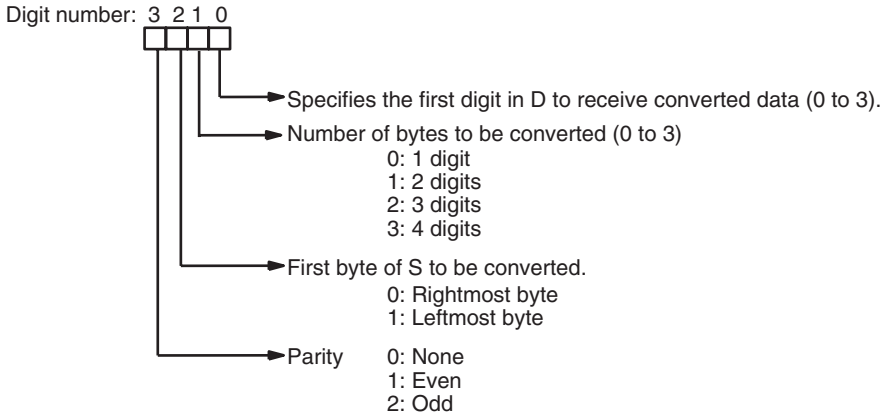
Operands

S: First Source Word

The contents of the source words are treated as ASCII data. Up to three source words can be used. (Three source words will be required if 4 bytes are being converted and the leftmost byte is selected as the first byte in S.) The source words must be in the same data area.

Di: Digit Designator

The digit designator specifies various parameters for the conversion, as shown in the following diagram.



D: Destination word

The converted hexadecimal digits are written into D from right to left, beginning with the specified first digit. Any digits in the destination word that are not overwritten with the converted data will be left unchanged.

Operand Specifications

Area	S	Di	D
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---	Specified values only	---
Data Resisters	---	DR0 to DR15	---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

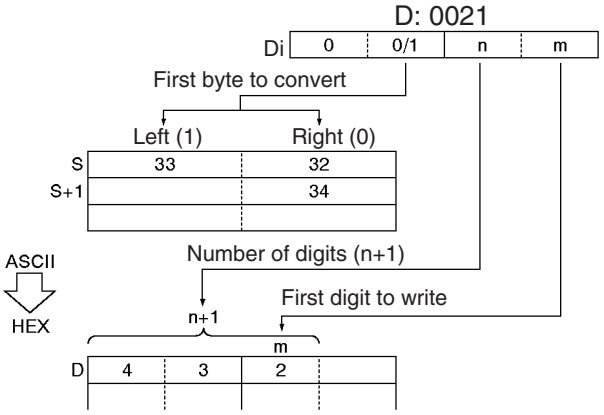
Description

HEX(162) treats the contents of the source word(s) as ASCII data representing hexadecimal digits (0 to 9 and A to F), converts the specified number of bytes to hexadecimal, and writes the hexadecimal data to the destination word beginning at the specified digit.

An error will occur if the source words contain data which is not an ASCII equivalent of hexadecimal digits. The following table shows hexadecimal digits and their ASCII equivalents (excluding parity bits).

Hexadecimal digits (4 bits)	ASCII equivalent (2 hexadecimal digits)
0 to 9	30 to 39
A to F	41 to 46

The following diagram shows the basic operation of HEX(162) with Di=0021.



Parity

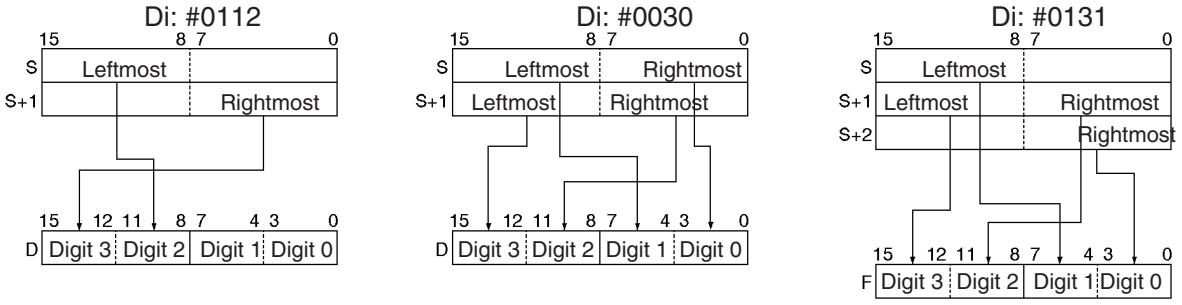
It is possible to specify the parity of the ASCII data for use in error control during data transmissions. The leftmost bit in each byte is the parity bit. With no parity the parity bit should always be zero, with even parity the status of the parity bit should result in an even number of ON bits, and with odd parity the status of the parity bit should result in an odd number of ON bits.

The following table shows the operation of HEX(162) for each parity setting.

Parity setting (leftmost digit of Di)	Operation of HEX(162)
No parity (0)	HEX(162) will be executed only when the parity bit in each byte is 0. An error will occur if a parity bit is non-zero.
Even parity (1)	HEX(162) will be executed only when there is an even number of ON bits in each byte. An error will occur if a byte has an odd number of ON bits.
Odd parity (2)	HEX(162) will be executed only when there is an odd number of ON bits in each byte. An error will occur if a byte has an even number of ON bits.

Examples of Di

When two or more bytes are being converted, HEX(162) will write the converted digits to the destination word from right to left and will wrap around to the rightmost digit if necessary. The following diagram shows some example values for Di and the conversions that they produce.



Flags

Name	Label	Operation
Error Flag	ER	ON if there is a parity error in the ASCII data. ON if the ASCII data in the source words is not equivalent to hexadecimal digits ON if the content of Di is not within the specified ranges. OFF in all other cases.

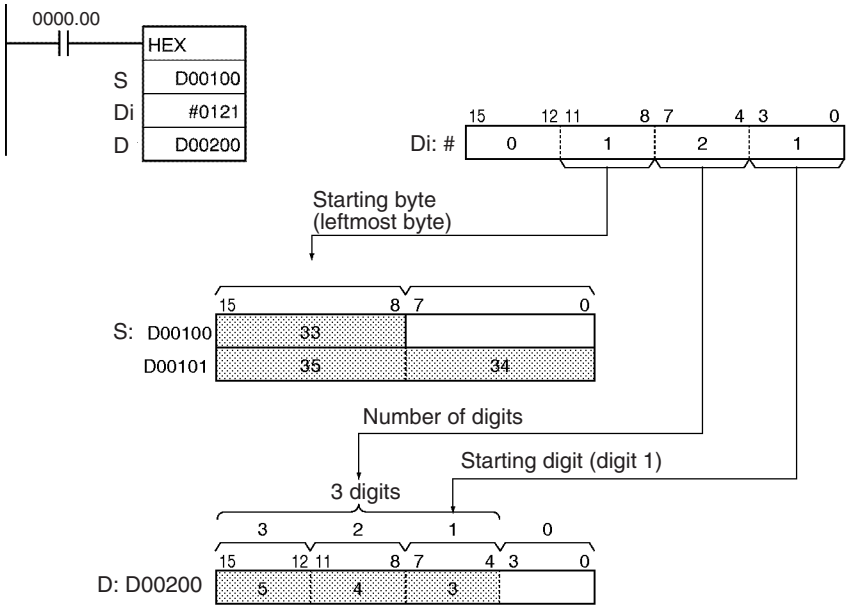
Precautions

An error will occur and the Error Flag will be turned ON if there is a parity error in the ASCII data, the ASCII data in the source words is not equivalent to hexadecimal digits, or the content of Di is not within the specified ranges.

Examples

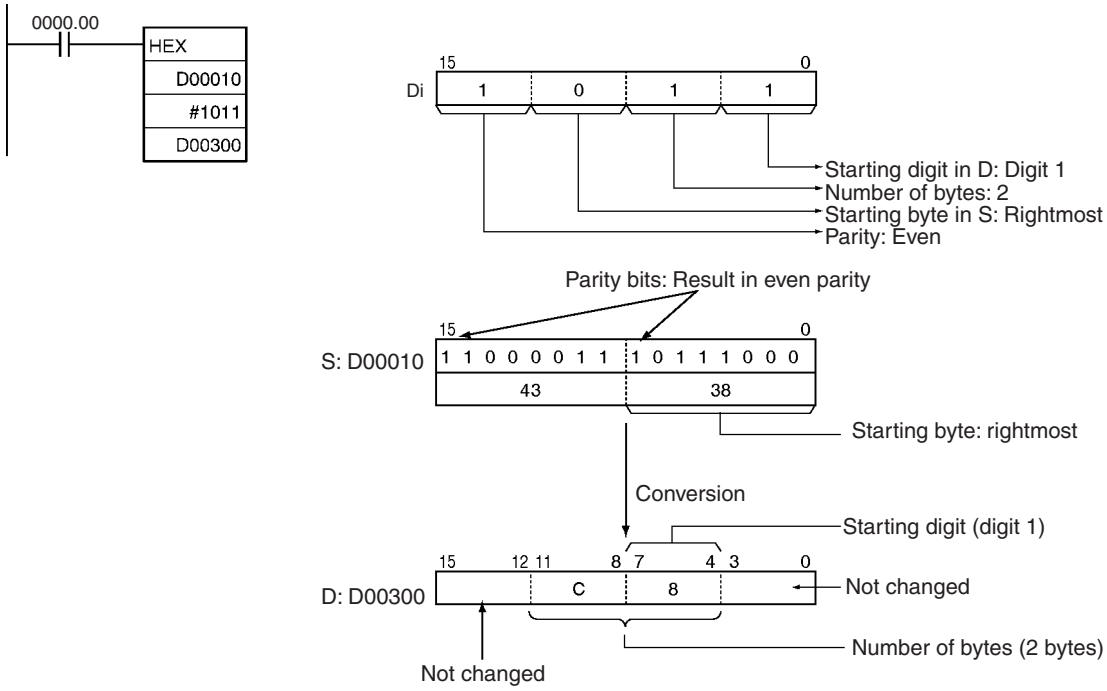
When CIO 0000.00 is ON in the following example, HEX(162) converts the ASCII data in D00100 and D00101 according to the settings of the digit designator. (Di=#0121 specifies no parity, the starting byte (when reading) = leftmost byte, the number of bytes to read = 3, and the starting digit (when writing) = digit 1.)

HEX(162) converts three bytes of ASCII data (3 characters) beginning with the leftmost byte of D00100 into their hexadecimal equivalents and writes this data to D00200 beginning with digit 1.



When CIO 0000.00 is ON in the following example, HEX(162) converts the ASCII data in D00010 beginning with the rightmost byte and writes the hexadecimal equivalents in D00300 beginning with digit 1.

The digit designator setting of #1011 specifies even parity, the starting byte (when reading) = rightmost byte, the number of bytes to read = 2, and the starting digit (when writing) = digit 1.)

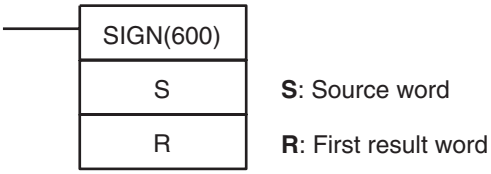


3-11-9 16-BIT TO 32-BIT SIGNED BINARY: SIGN(600)

Purpose

Expands a 16-bit signed binary value to its 32-bit equivalent.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	SIGN(600)
	Executed Once for Upward Differentiation	@SIGN(600)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142
Work Area	W000 to W255	W000 to W254
Auxiliary Bit Area	A000 to A959	A448 to A958
Timer Area	T0000 to T0255	T0000 to T0254
Counter Area	C0000 to C0255	C0000 to C0254
DM Area	D00000 to D32767	D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	

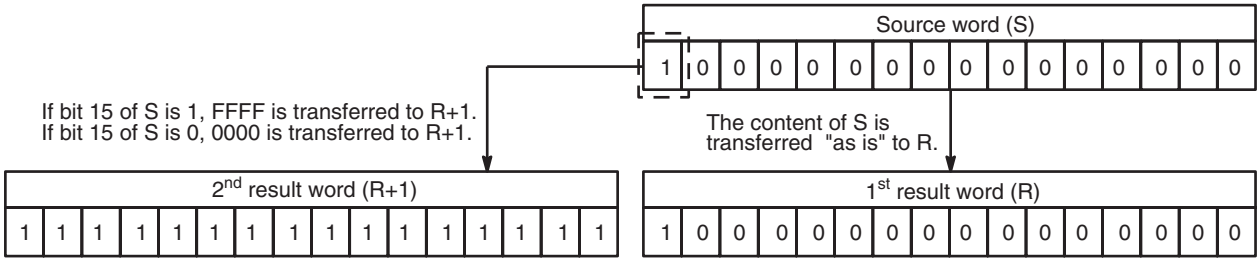
Area	S	R
Constants	#0000 to #FFFF (binary)	---
Data Registers	DR0 to DR15	---
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Note R and R+1 must be in the same data area.

Description

SIGN(600) converts the 16-bit signed binary number in S to its 32-bit signed binary equivalent and writes the result in R+1 and R.

The conversion is accomplished by copying the content of S to R and writing FFFF to R+1 if bit 15 of S is 1 or writing 0000 to R+1 if bit 15 of S is 0.

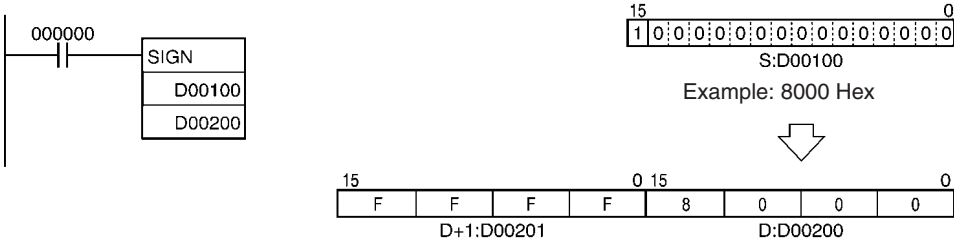


Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if the result is 0000 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 of R+1 is ON. OFF in all other cases.

Example

When CIO 000000 is ON in the following example, SIGN(600) converts the 16-bit signed binary content of D00100 (#8000 = –32,768 decimal) to its 32-bit equivalent (#FFFF 8000 = –32,768 decimal) and writes that result to D00201 and D00200.



3-12 Logic Instructions

Logic Instructions

This section describes instructions which perform logic operations on word data.

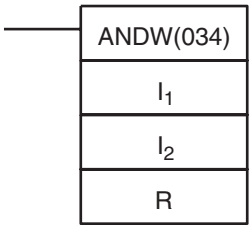
Instruction	Mnemonic	Function code	Page
LOGICAL AND	ANDW	034	351
DOUBLE LOGICAL AND	ANDL	610	353
LOGICAL OR	ORW	035	354
DOUBLE LOGICAL OR	ORWL	611	356
EXCLUSIVE OR	XORW	036	358
DOUBLE EXCLUSIVE OR	XORL	612	360
EXCLUSIVE NOR	XNRW	037	362
DOUBLE EXCLUSIVE NOR	XNRL	613	363
COMPLEMENT	COM	029	365
DOUBLE COMPLEMENT	COML	614	366

3-12-1 LOGICAL AND: ANDW(034)

Purpose

Takes the logical AND of corresponding bits in single words of word data and/or constants.

Ladder Symbol



I1: Input 1

I2: Input 2

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	ANDW(034)
	Executed Once for Upward Differentiation	@ANDW(034)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	I1	I2	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)		---

Area	I_1	I_2	R
Data Resisters	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

ANDW(034) takes the logical AND of data specified in I_1 and I_2 and outputs the result to R.

- The logical AND is taken of corresponding bits in I_1 and I_2 in succession.
- When the contents of corresponding bits in both I_1 and I_2 are 1, a 1 will be output to the corresponding bit in R. When one or more bits is 0, a 0 will be output to the corresponding bit in R.

I_1, I_2 → R

I_1	I_2	R
1	1	1
1	0	0
0	1	0
0	0	0

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000 hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

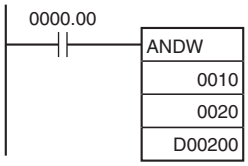
When ANDW(034) is executed, the Error Flag will turn OFF.

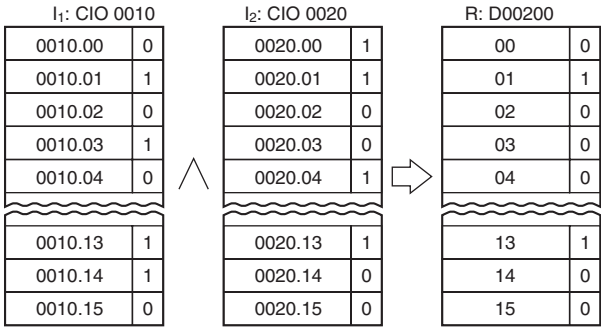
If as a result of the AND, the content of R is 0000 hex, the Equals Flag will turn ON.

If as a result of the AND, the leftmost bit of R is 1, the Negative Flag will turn ON.

Examples

When the execution condition CIO 0000.00 is ON, the logical AND will be taken of corresponding bits in CIO 0010 and CIO 0020, and the results will be output to corresponding bits in D00200.





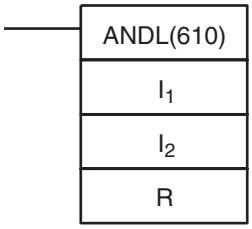
Note: The vertical arrow indicates logical AND.

3-12-2 DOUBLE LOGICAL AND: ANDL(610)

Purpose

Takes the logical AND of corresponding bits in double words of word data and/or constants.

Ladder Symbol



I1: Input 1

I2: Input 2

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	ANDL(610)
	Executed Once for Upward Differentiation	@ANDL(610)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	I_1	I_2	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)		---
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

ANDL(610) takes the logical AND of data specified in I_1, I_1+1 and I_2, I_2+1 and outputs the result to R, R+1.

(I_1, I_1+1), (I_2, I_2+1) → (R, R+1)

I_1, I_1+1	I_2, I_2+1	R, R+1
1	1	1
1	0	0
0	1	0
0	0	0

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000 0000 hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R+1 is 1. OFF in all other cases.

Precautions

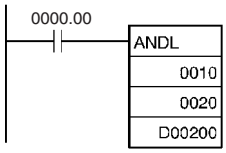
When ANDL(610) is executed, the Error Flag will turn OFF.

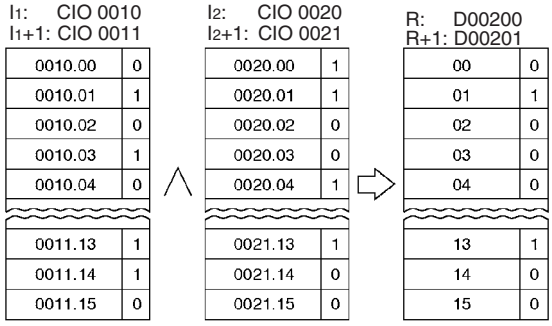
If as a result of the AND, the content of R, R+1 is 0000 0000 hex, the Equals Flag will turn ON.

If as a result of the AND, the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When the execution condition CIO 0000.00 is ON, the logical AND will be taken of corresponding bits in CIO 0011, CIO 0010 and CIO 0021, CIO 0020 and the results will be output to corresponding bits in D00201 and D00200.





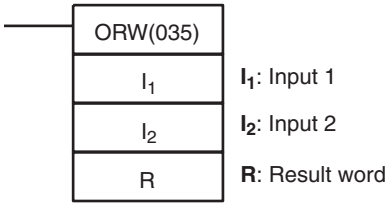
Note: The vertical arrow indicates logical AND.

3-12-3 LOGICAL OR: ORW(035)

Purpose

Takes the logical OR of corresponding bits in single words of word data and/or constants.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	ORW(035)
	Executed Once for Upward Differentiation	@ORW(035)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	I_1	I_2	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)		---
Data Resisters	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

ORW(035) takes the logical OR of data specified in I_1 and I_2 and outputs the result to R.

- The logical OR is taken of corresponding bits in I_1 and I_2 in succession.
- When either one of the corresponding bits in I_1 and I_2 is 1, a 1 will be output to the corresponding bit in R. When both of them are 0, a 0 will be output to the corresponding bit in R.

$I_1 + I_2 \rightarrow R$

I_1	I_2	R
1	1	1
1	0	1
0	1	1
0	0	0

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000 hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

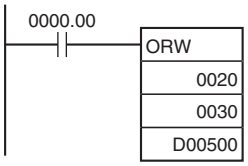
When ORW(035) is executed, the Error Flag will turn OFF.

If as a result of the OR, the content of R is 0000 hex, the Equals Flag will turn ON.

If as a result of the OR, the leftmost bit of R is 1, the Negative Flag will turn ON.

Examples

When the execution condition CIO 0000.00 is ON, the logical OR will be taken of corresponding bits in CIO 0020 and CIO 0030, and the results will be output to corresponding bits in D00500.



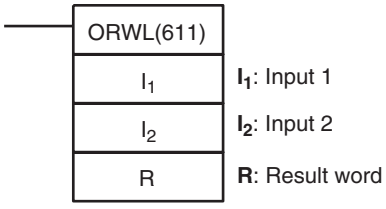
Note: The vertical arrow indicates logical OR.

3-12-4 DOUBLE LOGICAL OR: ORWL(611)

Purpose

Takes the logical OR of corresponding bits in double words of word data and/or constants.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	ORWL(611)
	Executed Once for Upward Differentiation	@ORWL(611)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	I_1	I_2	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)		---
Data Resisters	---		

Area	I_1	I_2	R
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

ORWL(611) takes the logical OR of data specified in I_1, I_1+1 and I_2, I_2 +1 as double-word data and outputs the result to R, R+1.

- When any of the corresponding bits in I_1, I_1+1, I_2, and I_2 +1 are 1, a 1 will be output to the corresponding bit in R, R+1. When both of them are 0, a 0 will be output to the corresponding bit in R, R+1.

(I_1, I_1+1) + (I_2, I_2+1) → (R, R+1)

I_1, I_1+1	I_2, I_2+1	R, R+1
1	1	1
1	0	1
0	1	1
0	0	0

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000 0000 hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

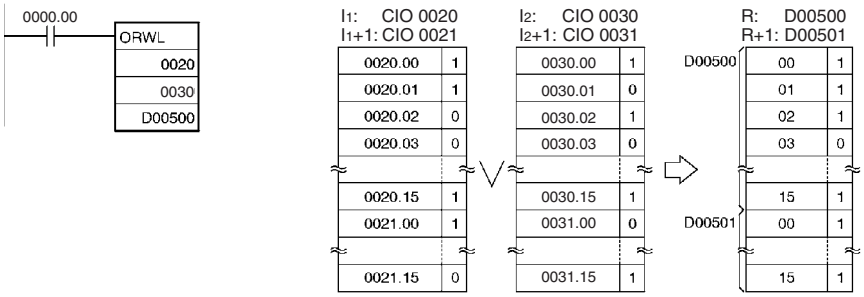
When ORWL(611) is executed, the Error Flag will turn OFF.

If as a result of the OR, the content of R, R+1 is 0000 0000 hex, the Equals Flag will turn ON.

If as a result of the OR, the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When the execution condition CIO 0000.00 is ON, the logical OR will be taken of corresponding bits in CIO 0021, CIO 0020 and CIO 0031, CIO 0030 and the results will be output to corresponding bits in D00501 and D00500.



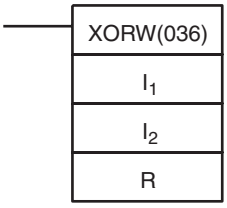
Note: The vertical arrow indicates logical OR.

3-12-5 EXCLUSIVE OR: XORW(036)

Purpose

Takes the logical exclusive OR of corresponding bits in single words of word data and/or constants.

Ladder Symbol



I1: Input 1

I2: Input 2

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	XORW(036)
	Executed Once for Upward Differentiation	@XORW(036)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	I_1	I_2	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)		---
Data Resisters	DR0 to DR15		

Area	I_1	I_2	R
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

XORW(036) takes the logical exclusive OR of data specified in I_1 and I_2 and outputs the result to R.

- The logical exclusive OR is taken of corresponding bits in I_1 and I_2 in succession.
- When the contents of corresponding bits of I_1 and I_2 are different, a 1 will be output to the corresponding bit of R and when they are the same, 0 will be output to the corresponding bit in R.

$$I_1, \overline{I_2} + \overline{I_1}, I_2 \rightarrow R$$

I_1	I_2	R
1	1	0
1	0	1
0	1	1
0	0	0

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

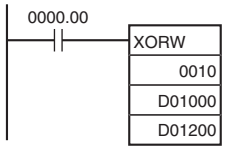
When XORW(036) is executed, the Error Flag will turn OFF.

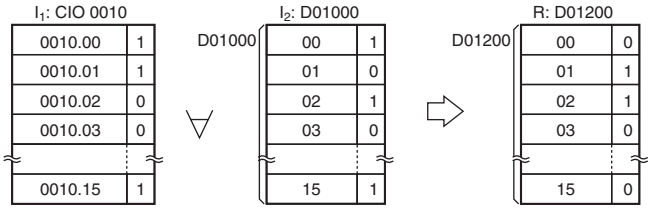
If as a result of the OR, the content of R is 0000 hex, the Equals Flag will turn ON.

If as a result of the OR, the leftmost bit of R is 1, the Negative Flag will turn ON.

Examples

When the execution condition CIO 0000.00 is ON, the logical exclusive OR will be taken of corresponding bits in CIO 0010 and D01000, and the results will be output to corresponding bits in D01200.





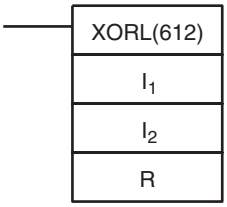
Note: The symbol indicates logical exclusive OR.

3-12-6 DOUBLE EXCLUSIVE OR: XORL(612)

Purpose

Takes the logical exclusive OR of corresponding bits in double words of word data and/or constants.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	XORL(612)
	Executed Once for Upward Differentiation	@XORL(612)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	I_1	I_2	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)		---
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

XORL(612) takes the logical exclusive OR of data specified in I_1, I_1+1 and I_2, I_2+1 as double-word data and outputs the result to R, R+1.

- When the contents of any of the corresponding bits in I_1, I_1+1, I_2, and I_2+1 are different, a 1 will be output to the corresponding bit it R, R+1. When both of them are the same, a 0 will be output to the corresponding bit in R, R+1.

$(I_1, I_1+1), (\overline{I_2, I_2+1}) + (\overline{I_1, I_1+1}), (I_2, I_2+1) \rightarrow (R, R+1)$

I_1, I_1+1	I_2, I_2+1	R, R+1
1	1	0
1	0	1
0	1	1
0	0	0

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000 0000 hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

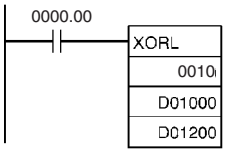
When XORL(612) is executed, the Error Flag will turn OFF.

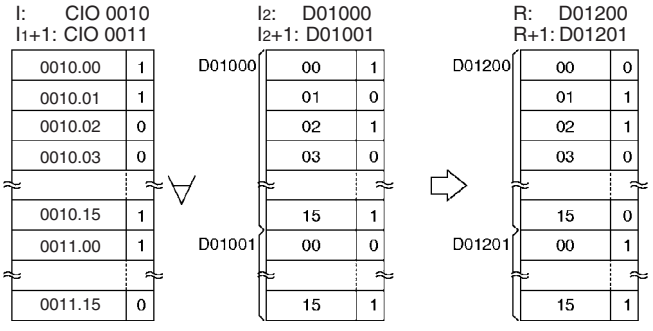
If as a result of the exclusive OR, the content of R, R+1 is 0000 0000 hex, the Equals Flag will turn ON.

If as a result of the exclusive OR, the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When the execution condition CIO 0000.00 is ON, the logical exclusive OR will be taken of corresponding bits in CIO 0011, CIO 0010 and D01001, D01000 and the results will be output to corresponding bits in D01201 and D01200.





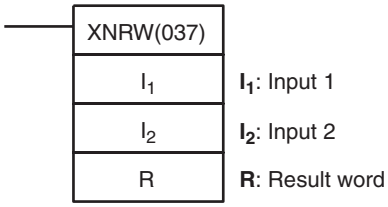
Note: The symbol indicates logical exclusive OR.

3-12-7 EXCLUSIVE NOR: XNRW(037)

Purpose

Takes the logical exclusive NOR of corresponding single words of word data and/or constants.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	XNRW(037)
	Executed Once for Upward Differentiation	@XNRW(037)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	I_1	I_2	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)		---
Data Resisters	DR0 to DR15		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

XNRW(037) takes the logical exclusive NOR of data specified in I_1 and I_2 and outputs the result to R.

- The logical exclusive NOR is taken of corresponding bits in I_1 and I_2 in succession.

- When the contents of corresponding bits of I_1 and I_2 are different, a 0 will be output to the corresponding bit of R and when they are the same, 1 will be output to the corresponding bit in R.

$I_1, I_2 + \overline{I_1}, \overline{I_2} \rightarrow R$

I_1	I_2	R
1	1	1
1	0	0
0	1	0
0	0	1

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000 hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

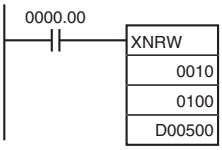
When XNRW(037) is executed, the Error Flag will turn OFF.

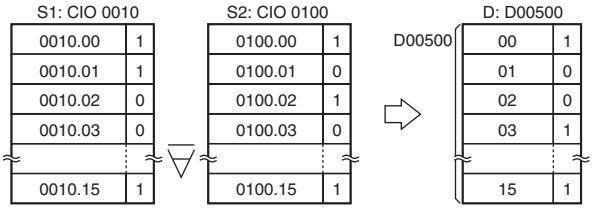
If as a result of the NOR, the content of R is 0000 hex, the Equals Flag will turn ON.

If as a result of the NOR, the leftmost bit of R is 1, the Negative Flag will turn ON.

Examples

When the execution condition CIO 0000.00 is ON, the logical exclusive NOR will be taken of corresponding bits in CIO 0010 and CIO 0100, and the results will be output to corresponding bits in D00500.





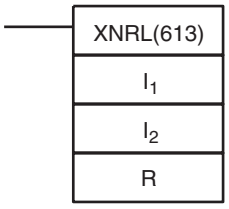
Note: The symbol indicates logical exclusive NOR.

3-12-8 DOUBLE EXCLUSIVE NOR: XNRL(613)

Purpose

Takes the logical exclusive NOR of corresponding bits in double words of word data and/or constants.

Ladder Symbol



I_1: Input 1

I_2: Input 2

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	XNRL(613)
	Executed Once for Upward Differentiation	@XNRL(613)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	I_1	I_2	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 toW254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)		---
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

XNRL(613) takes the logical exclusive NOR of data specified in I_1, I_1+1 and I_2, I_2+1 and outputs the result to R, R+1.

- When the contents of any of the corresponding bits in I_1, I_1+1, I_2, and I_2 +1 are different, a 0 will be output to the corresponding bit in R, R+1. When both of them are the same, a 1 will be output to the corresponding bit in R, R+1.

$(I_1, I_1+1), (I_2, I_2+1) + \overline{(I_1, I_1+1)}, \overline{(I_2, I_2+1)} \rightarrow (R, R+1)$

I_1, I_1+1	I_2, I_2+1	R, R+1
1	1	1
1	0	0
0	1	0
0	0	1

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000 0000 hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R+1 is 1. OFF in all other cases.

Precautions

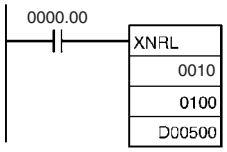
When XNRL(613) is executed, the Error Flag will turn OFF.

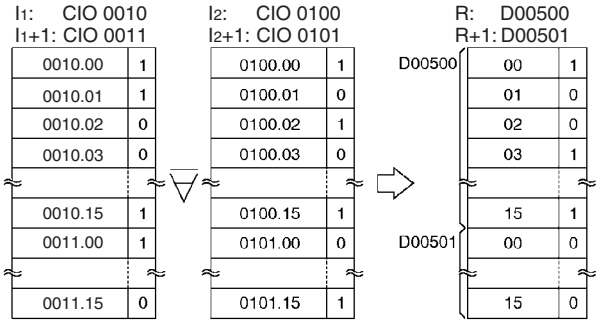
If as a result of the exclusive NOR, the content of R, R+1 is 0000 0000 hex, the Equals Flag will turn ON.

If as a result of the exclusive NOR, the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When the execution condition CIO 0000.00 is ON, the logical exclusive NOR will be taken of corresponding bits in CIO 0011, CIO 0010, and CIO 0101, CIO 0100 and the results will be output to corresponding bits in D00501 and D00500.





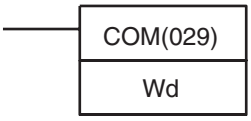
Note: The symbol indicates logical exclusive NOR.

3-12-9 COMPLEMENT: COM(029)

Purpose

Turns OFF all ON bits and turns ON all OFF bits in Wd.

Ladder Symbol



Wd: Word

Variations

Variations		
	Executed Each Cycle for ON Condition	COM(029)
	Executed Once for Upward Differentiation	@COM(029)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W255
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
DM Area	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	DR0 to DR15

Area	Wd
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

COM(029) reverses the status of every bit in the specified word.
$\overline{Wd} \rightarrow Wd$: 1 → 0 and 0 → 1

Note When using the COM instruction, be aware that the status of each bit will change each cycle in which the execution condition is ON.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000 hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R is 1. OFF in all other cases.

Precautions

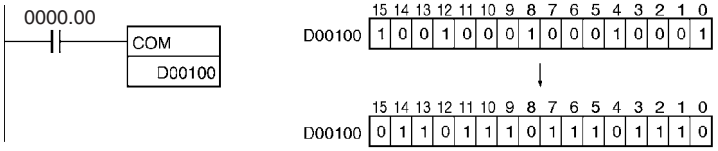
When COM(029) is executed, the Error Flag will turn OFF.

If as a result of COM, the content of R is 0000 hex, the Equals Flag will turn ON.

If as a result of COM, the leftmost bit of R is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, the status of each bit in D00100 will be reversed.

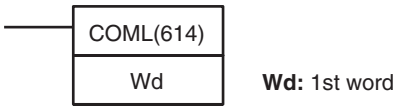


3-12-10 DOUBLE COMPLEMENT: COML(614)

Purpose

Turns OFF all ON bits and turns ON all OFF bits in Wd and Wd+1.

Ladder Symbol



Wd: 1st word

Variations

Variations		
	Executed Each Cycle for ON Condition	COML(614)
	Executed Once for Upward Differentiation	@COML(614)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Wd
CIO Area	CIO 0000 to CIO 6142
Work Area	W000 to W254
Auxiliary Bit Area	A448 to A958
Timer Area	T0000 to T0254
Counter Area	C0000 to C0254
DM Area	D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767
Indirect DM addresses in BCD	*D00000 to *D32767
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

COML(614) reverses the status of every bit in Wd and Wd+1.
$\overline{(Wd+1, Wd)} \rightarrow (Wd+1, Wd)$

Note When using the COML instruction, be aware that the status of each bit will change each cycle in which the execution condition is ON.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON when the result is 0000 0000 hex. OFF in all other cases.
Negative Flag	N	ON when the leftmost bit of R+1 is 1. OFF in all other cases.

Precautions

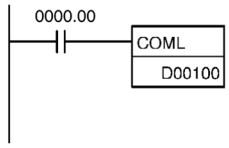
When COML(614) is executed, the Error Flag will turn OFF.

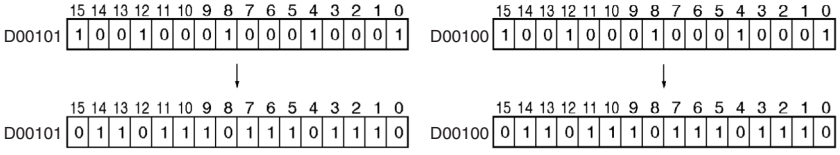
If as a result of COML, the content of R, R+1 is 0000 0000 hex, the Equals Flag will turn ON.

If as a result of COML, the leftmost bit of R+1 is 1, the Negative Flag will turn ON.

Examples

When CIO 0000.00 is ON in the following example, the status of each bit in D00100 and D00101 will be reversed.





3-13 Special Math Instructions

This section describes instructions used for special math calculations.

Instruction	Mnemonic	Function code	Page
ARITHMETIC PROCESS	APR	069	368
BIT COUNTER	BCNT	067	375
VIRTUAL AXIS	AXIS	981	376

3-13-1 ARITHMETIC PROCESS: APR(069)

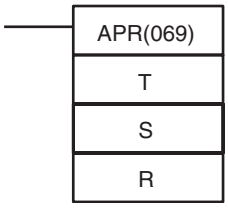
Purpose

Calculates the linear extrapolation of the source data (16-bit or 32-bit binary). The linear extrapolation function allows any relationship between X and Y to be approximated with line segments.

The high-speed counter PV can be used directly as input data.

With the Motion Control Modules, the linear extrapolation data table can be transferred to the high-speed buffer so the linear extrapolation calculations can be processed at high speed.

Ladder Symbol



T: First word of linear extrapolation data table

S: Source data

R: Result word

Variations

Variations		
	Executed Each Cycle for ON Condition	APR(069)
	Executed Once for Upward Differentiation	@APR(069)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

Operand	Value	Data range
T	Data area address	--- (See note.)
S	16-bit unsigned binary data	0000 to 65,535
	16-bit signed binary data	–32,768 to 32,767
	32-bit signed binary data	–2,147,483,648 to 2,147,483,647
R	32-bit unsigned binary data	00000000 to 4,294,967,295
	16-bit signed binary data	–32,768 to 32,767
	32-bit signed binary data	–2,147,483,648 to 2,147,483,647

Note T is the first word of the range containing the linear extrapolation data table (binary).

Operand Specifications

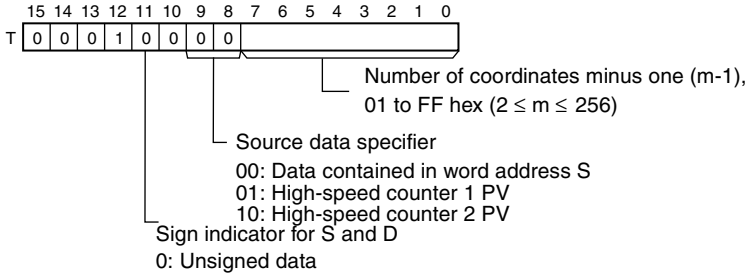
Area	T	S	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959

Area	T	S	R
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	Specified values only		---
Data Resisters	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

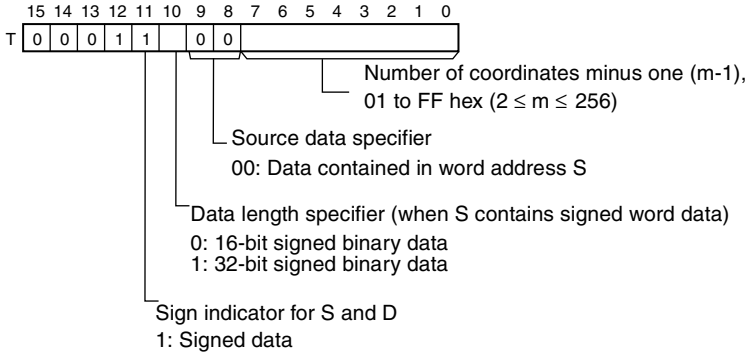
Operand Description (Table Words)

The following diagrams show the structure of the linear extrapolation data table for the various data formats.

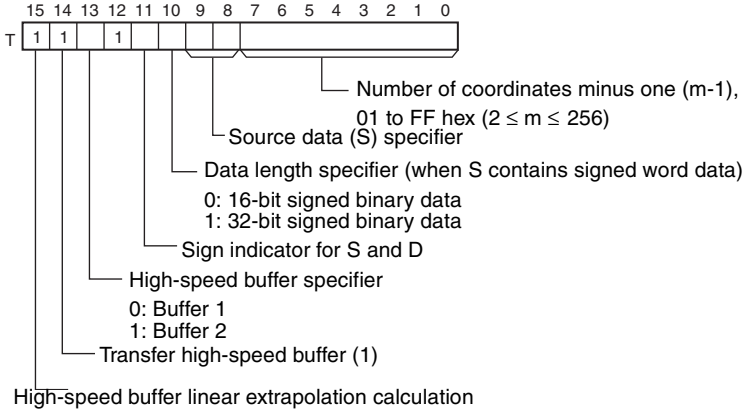
Unsigned Integer Data (Binary)



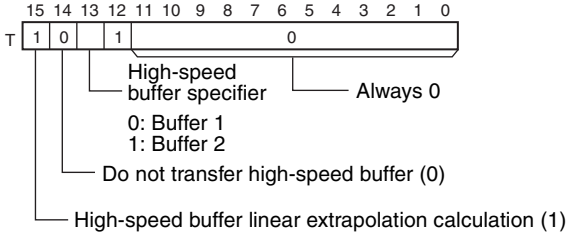
Signed Integer Data (Binary)



High-speed Buffer Transfer of Extrapolation Table (FQM1-MMP22 and FQM1-MMA22 Only)



High-speed Buffer Linear Extrapolation Calculation (FQM1-MMP22 and FQM1-MMA22 Only)



Unsigned integer data

T+1	Xm (Max. X value)
T+2	Y0 (rightmost 16 bits)
T+3	Y0 (leftmost 16 bits)
T+4	X1 (16 bits)
T+5	Y1 (rightmost 16 bits)
T+6	Y1 (leftmost 16 bits)
to	to
T+ (3m+1)	Xm (Max. X value)
T+ (3m+2)	Ym (rightmost 16 bits)
T+ (3m+3)	Ym (rightmost 16 bits)

16-bit signed binary data

T+1	X0
T+2	Y0
T+3	X1
T+4	Y1
T+5	X2
T+6	Y2
	to
T+ (2n+1)	Xn
T+ (2n+2)	Yn
to	to
T+ (2m+1)	Xm
T+ (2m+2)	Ym

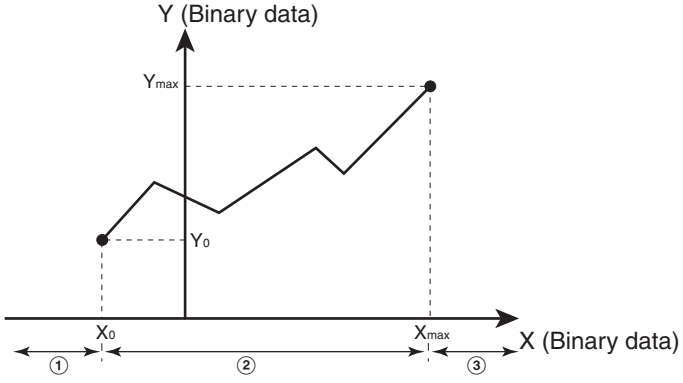
32-bit signed binary data

T+1	X0 (rightmost 16 bits)
T+2	X0 (leftmost 16 bits)
T+3	Y0 (rightmost 16 bits)
T+4	Y0 (leftmost 16 bits)
T+5	X1 (rightmost 16 bits)
T+6	X1 (rightmost 16 bits)
T+7	Y1 (rightmost 16 bits)
T+8	Y0 (leftmost 16 bits)
to	to
T+ (4n+1)	Xn (rightmost 16 bits)
T+ (4n+2)	Xn (leftmost 16 bits)
T+ (4n+3)	Yn (rightmost 16 bits)
T+ (4n+4)	Yn (leftmost 16 bits)
to	to
T+ (4m+1)	Xm (rightmost 16 bits)
T+ (4m+2)	Xm (leftmost 16 bits)
T+ (4m+3)	Ym (rightmost 16 bits)
T+ (4m+4)	Ym (leftmost 16 bits)

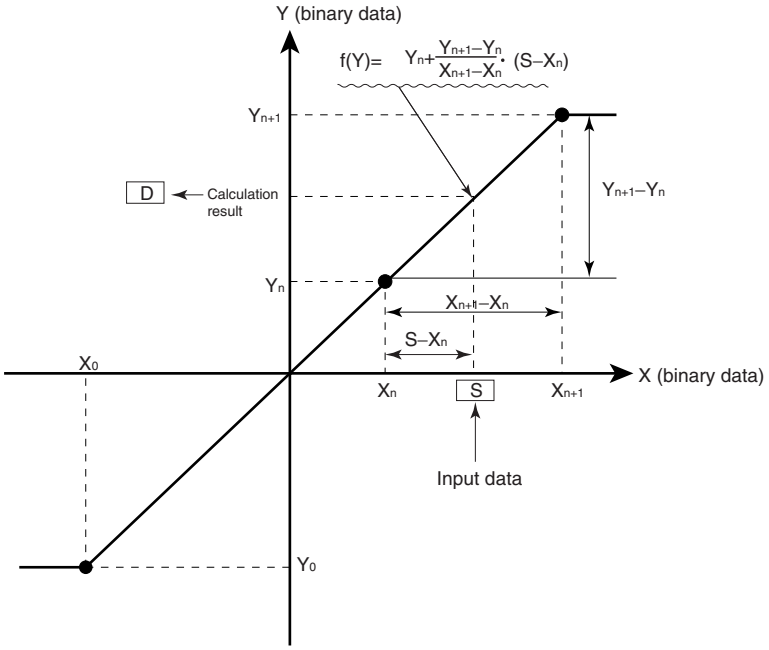
Note The X coordinates must be in ascending order: $X_1 < X_2 < ... < X_m$. Input all values of (X_n, Y_n) as binary data, regardless of the data format specified in control word T.

Description of the Linear Extrapolation Function

APR(069) processes the input data specified in S with the following equation and the line-segment data (X_n, Y_n) specified in the table beginning at T+1. The result is output to the destination word(s) specified with D.



1. For $S < X_0$
   Converted value = Y_0
2. For $X_0 \leq S \leq X_{max}$, if $X_n < S < X_{n+1}$
   Converted value = $Y_n + [\{Y_n + 1 - Y_n\}/\{X_n + 1 - X_n\}] \times [$*Input data* $S - X_n\}$



3. $X_{max} < S$
   Converted value = Y_{max}

Up to 256 endpoints can be stored in the line-segment data table beginning at T+1. The following 3 kinds of I/O data can be used:

- 16-bit unsigned binary input data/32-bit unsigned binary output data
- 16-bit signed binary I/O data
- 32-bit signed binary I/O data

Flags

Name	Label	Operation
Error Flag	ER	ON if the X coordinates in the table starting at T are not in ascending order ($X_1 \leq X_2 \leq ... \leq X_m$). ON if bits 9 and 8 of T are not 00, 01 or 10. ON if there is an error in the linear extrapolation data table starting at T. ON if the linear extrapolation calculation's source data is not within the defined graph. ON if APR(069) is being used in a Coordinator Module and the source data is set to high-speed counter data. ON if APR(069) is being used in a Coordinator Module and high-speed buffer linear extrapolation is selected. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 of D is ON. OFF in all other cases.

- An error will occur and the ER Flag will be turned ON if the X coordinates are not in ascending order ($X_1 < X_2 < ... < X_m$).

- The Equals Flag will be turned ON if the calculation result in D or D and D+1 is 0, as shown in the following table.)

Data format	Content of D (or
Unsigned 16-bit data	0000 0000 hex
Signed 16-bit data	0000 hex
Signed 32-bit data	0000 0000 hex

- The N Flag will be turned ON if the most significant bit of the calculation result (D or D and D+1) is 1.

Examples

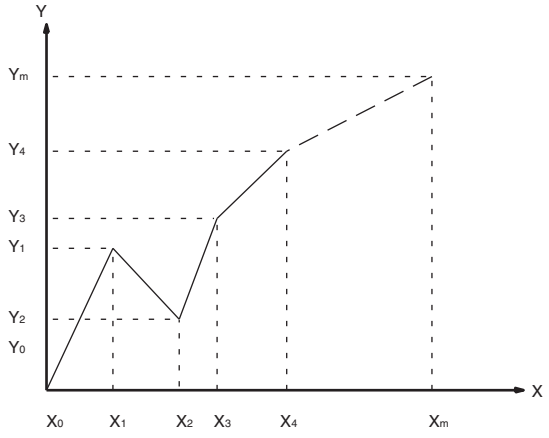
Linear Extrapolation Using 16-bit Unsigned Binary Data

APR(069) processes the input data specified in S based on the control data in T and the line-segment data specified in the table beginning at T+1. The result is output to D.

The following table shows the control data in T.

Bit	Setting name	Setting
15	High-speed buffer calculation specifier	0: Normal 1: High-speed buffer
14	Linear approximation data table to high-speed buffer transfer	0: Do not transfer 1: Transfer
13	High-speed buffer specifier (See note.)	0: Buffer 1 1: Buffer 2
12	---	1
08 to 11	---	0000
00 to 07	m-1 (m is the number of coordinates.)	

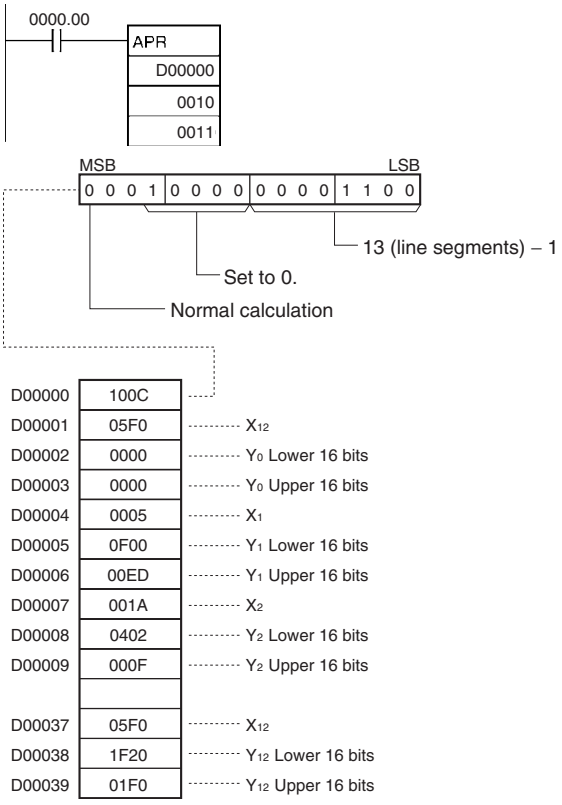
Note Bit 13 determines the processing of source data S.
When bit 13 is 0, the source data in S is used as the input data.
When bit 13 is 1, the source data is subtracted from the maximum X value and that result (X_m–S) is used as the input data. Use this method when the extrapolation data table is based on the Y-axis instead of the X-axis.



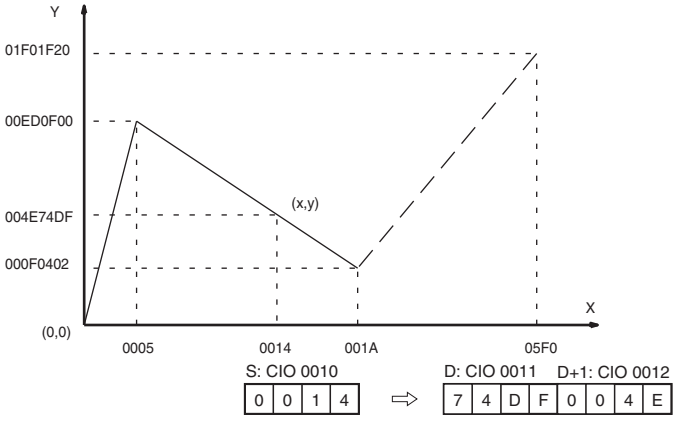
Word	Coordinate
T+1	Xm (max. X value)
T+2	Y_0 (rightmost 16 bits)
T+3	Y_0 (leftmost 16 bits)
T+4	X_1
T+5	Y_1 (rightmost 16 bits)
T+6	Y_1 (leftmost 16 bits)
↓	↓
T+(3m+1)	X_m
T+(3m+2)	Y_m (rightmost 16 bits)
T+(3m+3)	Y_m (leftmost 16 bits)

- $Y_n = f(X_n)$, $Y_0 = f(X_0)$
- Be sure that $X_{n-1} < X_n$ in all cases.
- Input all values of (X_n, Y_n) as binary data.

This example shows how to construct a linear extrapolation with 12 coordinates. The block of data is continuous, as it must be, from D00000 to D0039 (T to T + (3 × 12 + 3)). The input data is taken from CIO 0010, and the result is output to CIO 0011 and CIO 0011.



In this case, the source word, CIO 0010, contains 0014, and f(0014) = 004E 74DF is output to D and D+1, CIO 0011 and CIO 0011.



The linear-extrapolation calculation is shown below.

$$Y = 00ED0F00 + \frac{000F0402 - 00ED0F00}{001A - 0005} \times (0014 - 0005)$$

$$= 00ED0F00 - (A92CF \times 000F)$$

$$= 004E74DF$$

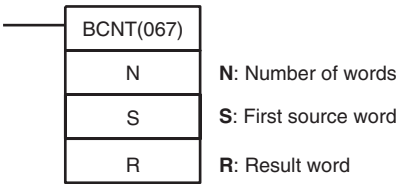
Values are all hexadecimal (Hex).

3-13-2 BIT COUNTER: BCNT(067)

Purpose

Counts the total number of ON bits in the specified word(s).

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	BCNT(067)
	Executed Once for Upward Differentiation	@BCNT(067)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

N: Number of words

The number of words must be 0001 to FFFF (1 to 65,535 words).

S: First source word

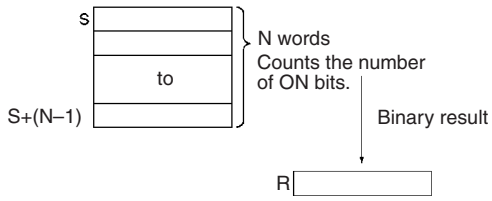
S and S+(N–1) must be in the same data area.

Operand Specifications

Area	N	S	R
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		A448 to A959
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0001 to #FFFF (binary) or &1 to &65,535	---	
Data Resisters	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

BCNT(067) counts the total number of bits that are ON in all words between S and S+(N–1) and places the result in R.



Flags

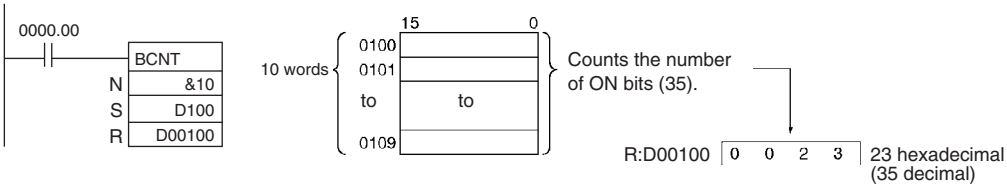
Name	Label	Operation
Error Flag	ER	ON if N is 0000. ON if result exceeds FFFF. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.

Precautions

An error will occur if N=0000 or the result exceeds FFFF.

Example

When CIO 0000.00 is ON in the following example, BCNT(067) counts the total number of ON bits in the 10 words from CIO 0100 through CIO 0109 and writes the result to D00100.



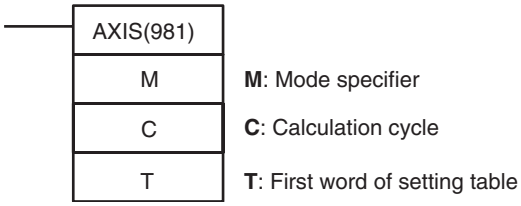
3-13-3 VIRTUAL AXIS: AXIS(981)

Purpose

Generates a virtual pulse output with trapezoidal acceleration/deceleration.

The operands for AXIS(981) are a target position specified in pulses or as an absolute position, and a target speed specified in pulses/s (Hz). While the input condition is ON, AXIS(981) internally generates the specified number of pulses and integrates (counts) the number of pulses (area) in the trapezoid.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	AXIS(981)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

M: Mode specifier

#0000: Relative mode
#0001: Absolute mode

C: Calculation cycle

#0000: 2 ms calculation cycle
#0001: 1 ms calculation cycle
#0002: 0.5 ms calculation cycle
#0003: 3 ms calculation cycle (Can be selected in unit version 3.2 or later.)
#0004: 4 ms calculation cycle (Can be selected in unit version 3.2 or later.)

T: First Word of Setting Table

Address		Name	Description	Setting range	Set/ monitored
T		Internal pulse count (8-digit hexadecimal)	The present value of internal pulse counter is stored here.	Relative mode: 0000 0000 to FFFF FFFF Absolute mode: 8000 0000 to 7FFF FFFF	Monitored (Read)
T+1					
T+2	Bit 15	Virtual pulse output status	Indicates whether or not the virtual pulse output has started.	OFF: Pulse output stopped ON: Pulse being output	
	Bit 08		Indicates the direction of virtual pulse currently being output.	OFF: CW ON: CCW	
	Bit 07		Indicates whether or not the virtual pulse output is being counted internally.	OFF: Pulse being counted ON: Target position reached (Counting stopped)	
	Bit 01		Indicates whether the virtual pulse output is accelerating or decelerating. The status of bit 01 can be logically ANDed with bit 00 to determine the status. For example, if bit 00 = 1 and bit 01 = 0, the virtual pulse output is accelerating.	OFF: Accelerating or constant speed ON: Decelerating Note: This function is supported only in CPU Units with unit version 3.2 or later.	
	Bit 00		Indicates whether or not the virtual pulse output is accelerating/decelerating.	OFF: Constant speed ON: Accelerating/decelerating	
T+3 to T+4		Present speed (8-digit hexadecimal)	The frequency of the virtual pulse output is stored here.	0000 0000 to 000F 4240 hex (0 to 1 MHz in 1-Hz units)	
T+5 to T+6		Target position (8-digit hexadecimal)	Set the number of virtual output pulses here.	Relative mode: 0000 0000 to FFFF FFFF Absolute mode: 8000 0000 to 7FFF FFFF	Set (Read/ Write)
T+7 to T+8		Target frequency (8-digit hexadecimal)	Set the target frequency of virtual pulses here.	0000 0001 to 000F 4240 hex (1 to 1 MHz in 1-Hz units)	
T+9 to T+10		Starting frequency (8-digit hexadecimal)	Set the starting frequency of virtual pulses here.	0000 0000 to 000F 4240 hex (0 to 1 MHz in 1-Hz units)	
T+11		Acceleration rate (4-digit hexadecimal)	Set the acceleration rate of virtual pulses here.	0001 to 270F (1 to 9,999 Hz, in 1-Hz units)	
T+12		Deceleration rate (4-digit hexadecimal)	Set the deceleration rate of virtual pulses here.	0001 to 270F (1 to 9,999 Hz, in 1-Hz units)	
T+13 to T+26		Work area	Used by the system.		---

Description

- Use the AXIS instruction with an input condition that is ON for one cycle. AXIS cannot be used as a differentiated instruction (the @ prefix is not supported).

- AXIS is executed at the rising edge of the input condition. If the input remains ON, the virtual pulse output continues until the target position is reached. Once the target position is reached, the virtual pulse output is stopped. If the input condition goes OFF during the virtual pulse output, the output stops at that point.
- The AXIS instruction's mode specifier operand (M) specifies whether the virtual pulse output operates in relative or absolute mode.
  - In relative mode, the internal pulse counter initializes the internal pulse count to 0 when AXIS is executed and starts incrementing from 0.
  - In absolute mode, the internal pulse counter retains the internal pulse count when AXIS is executed and starts incrementing or decrementing from that existing pulse count.
- The internal pulse counts are refreshed every cycle at the interval specified in the calculation cycle (4 ms, 3 ms, 2 ms, 1 ms, or 0.5 ms) with a constant execution cycle. If the specified calculation cycle time does not match the execution cycle time, the time difference between the cycles can cause an error in the count. If highly accurate pulse counts are required, use the constant cycle time function and match the execution cycle time and calculation cycle time. (Set the constant cycle time in the System Setup's Cycle Time Tab Page.)
- When AXIS is being used, the virtual axis operates in the following manner.
  a) AXIS starts the internal pulse count at the starting frequency and increases the frequency each calculation cycle by the frequency increment set in the acceleration rate.
  b) When the target frequency is reached, incrementing the frequency stops and the pulse count continues at a constant frequency.
  c) The point to start decreasing the frequency (the deceleration point) is determined from the deceleration rate and the remaining number of travel pulses, which is calculated from the preset target position. When the deceleration point is reached, AXIS decreases the frequency each calculation cycle by the frequency increment set in the deceleration rate. The internal pulse count stops when the target position is reached.
- When trapezoidal control cannot be performed with the specified target position, target frequency, and acceleration/deceleration, AXIS will automatically compensate as follows:

  The acceleration and deceleration rates will be set to the same rate (symmetrical trapezoidal control).

  OR

  When one-half of the specified target pulses have been output, AXIS will start decelerating at the specified acceleration rate (symmetrical triangular control).

Note When the AXIS instruction's input condition goes OFF, the contents of setting table words T+2 to T+4 will be initialized to 0.

Operand Specifications

Area	M	C	T
CIO Area	---		CIO 0000 to CIO 6117
Work Area	---		W000 to W229
Auxiliary Bit Area	---		A000 to A933

Area	M	C	T
Timer Area	---		T0000 to T0229
Counter Area	---		C0000 to C0229
DM Area	---		D00000 to D32741
Indirect DM addresses in binary	---		@ D00000 to @ D32767
Indirect DM addresses in BCD	---		*D00000 to *D32767
Constants	Specified values only		---
Data Resisters	---	---	---
Index Registers	---		
Indirect addressing using Index Registers	---		,IR0 or ,IR1 –2048 to +2047 ,IR0 or –2048 to +2047 ,IR1 ,IR0+(++) or ,IR1+(++) , –(– –)IR0 or, –(– –)IR1

Flags

Name	Label	Operation
Error Flag	ER	ON if the settings for the target frequency, starting frequency, and acceleration/deceleration rate are inconsistent. ON if the setting table starting at T contains one of the following invalid settings when the instruction is executed: • Target frequency < Deceleration rate • Target frequency > 1,000,000 or Target frequency = 0 • Starting frequency > 1,000,000 • Target frequency < Starting frequency • Acceleration rate > 9,999 or Acceleration rate = 0 • Deceleration rate > 9,999 or Deceleration rate = 0 • Target position (travel amount in relative mode) = 0 • Target position (target position in absolute mode) = Present position OFF in all other cases.

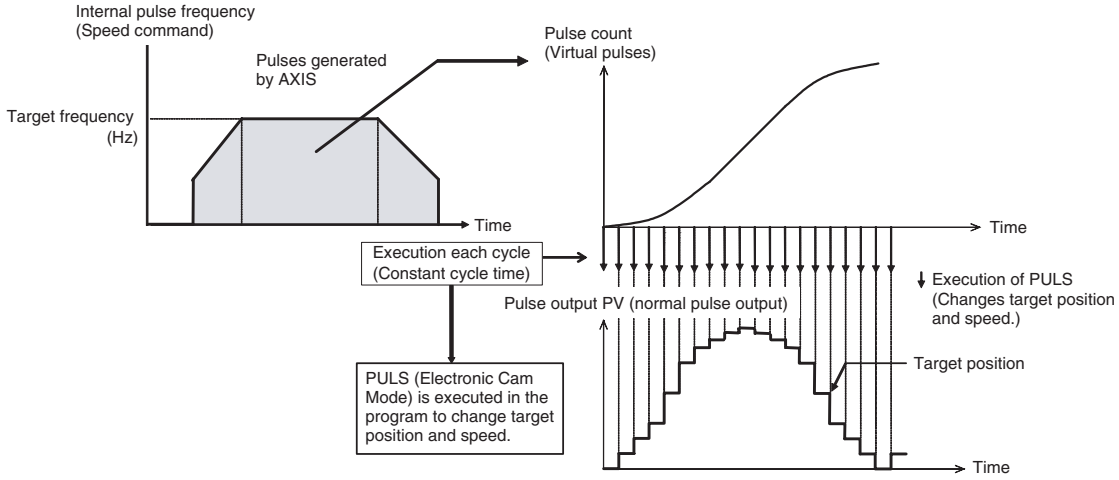
Example

Positioning or Speed Control on a Virtual Axis

The internal pulse count can be treated as a virtual axis position in order to perform electronic cam operation on that virtual axis position with simple linear approximation.

First, the AXIS instruction is executed to generate an internal pulse count. The internal pulse count is read at every cycle, that pulse count is processed with basic arithmetic operations or the APR instruction, and the result is used as a target position or target speed in the PULS(886) instruction. The PULS(886)

instruction (in electronic cam control) is executed immediately after the target position or speed is calculated.



Simple locus control can be performed by executing electronic cam control simultaneously on virtual axes for both pulse outputs 1 and 2.

3-14 Floating-point Math Instructions

Instruction	Mnemonic	Function code	Page
FLOATING TO 16-BIT	FIX	450	386
FLOATING TO 32-BIT	FIXL	451	388
16-BIT TO FLOATING	FLT	452	389
32-BIT TO FLOATING	FLTL	453	390
FLOATING-POINT ADD	+F	454	392
FLOATING-POINT SUB-TRACT	–F	455	394
FLOATING-POINT MULTI-PLY	*F	456	396
FLOATING-POINT DIVIDE	/F	457	397
DEGREES TO RADIANS	RAD	458	400
RADIANS-TO-DEGREES	DEG	459	401
SINE	SIN	460	403
COSINE	COS	461	404
TANGENT	TAN	462	406
ARC SINE	ASIN	463	408
ARC COSINE	ACOS	464	410
ARC TANGENT	ATAN	465	412
SQUARE ROOT	SQRT	466	413
EXPONENT	EXP	467	415
LOGARITHM	LOG	468	417
EXPONENTIAL POWER	PWR	840	419

The following floating-point comparison instructions are supported in addition to the instructions listed above.

Instruction	Mnemonic	Function code	Page
Single-precision Floating-point Symbol Comparison Instructions	LD, AND, OR + =F, <>F, <F, <=F, >F, or >=F	329 to 334	421

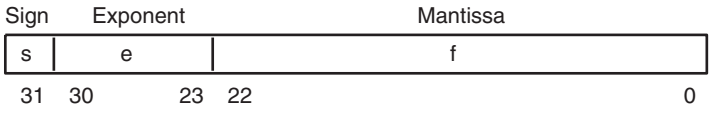
Data Format

Floating-point data expresses real numbers using a sign, exponent, and mantissa. When data is expressed in floating-point format, the following formula applies.

Real number = $(-1)^s \, 2^{e-127} \, (1.f)$

s: Sign
e: Exponent
f: Mantissa

The floating-point data format conforms to the IEEE754 standards. Data is expressed in 32 bits, as follows:



Data	No. of bits	Contents
s: sign	1	0: positive; 1: negative
e: exponent	8	The exponent (e) value ranges from 0 to 255. The actual exponent is the value remaining after 127 is subtracted from e, resulting in a range of –127 to 128. “e=0” and “e=255” express special numbers.
f: mantissa	23	The mantissa portion of binary floating-point data fits the formal 2.0 > 1.f ≥1.0.

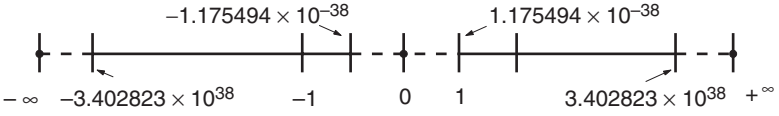
Number of Digits

The number of effective digits for floating-point data is 24 bits for binary (approximately seven digits decimal).

Floating-point Data

The following data can be expressed by floating-point data:

- $-\infty$
- $-3.402823 \times 10^{38} \leq \text{value} \leq -1.175494 \times 10^{-38}$
- 0
- $1.175494 \times 10^{-38} \leq \text{value} \leq 3.402823 \times 10^{38}$
- $+\infty$
- Not a number (NaN)



Special Numbers

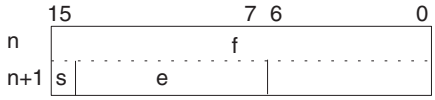
The formats for NaN, ±∞, and 0 are as follows:

NaN*: e = 255, f ≠ 0
+∞: e = 255, f = 0, s= 0
–∞: e = 255, f = 0, s= 1
0: e = 0

*NaN (not a number) is not a valid floating-point number. Executing floating-point calculation instructions will not result in NaN.

Writing Floating-point Data

When floating-point is specified for the data format in the I/O memory edit display in the CX-Programmer, standard decimal numbers input in the display are automatically converted to the floating-point format shown above (IEEE754-format) and written to I/O Memory. Data written in the IEEE754-format is automatically converted to standard decimal format when monitored on the display.



It is not necessary for the user to be aware of the IEEE754 data format when reading and writing floating-point data. It is only necessary to remember that floating point values occupy two words each.

Numbers Expressed as Floating-point Values

The following types of floating-point numbers can be used.

Mantissa (f)	Exponent (e)		
	0	Not 0 and not all 1's	All 1's (255)
0	0	Normalized number	Infinity
Not 0	Non-normalized number		NaN

Note A non-normalized number is one whose absolute value is too small to be expressed as a normalized number. Non-normalized numbers have fewer significant digits. If the result of calculations is a non-normalized number (including intermediate results), the number of significant digits will be reduced.

Normalized Numbers

Normalized numbers express real numbers. The sign bit will be 0 for a positive number and 1 for a negative number.

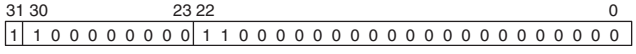
The exponent (e) will be expressed from 1 to 254, and the real exponent will be 127 less, i.e., –126 to 127.

The mantissa (f) will be expressed from 0 to $2^{23} – 1$, and it is assume that, in the real mantissa, bit 2^{23} is 1 and the binary point follows immediately after it.

Normalized numbers are expressed as follows:

$(–1)^{(\text{sign s})} \times 2^{(\text{exponent e})–127} \times (1 + \text{mantissa} \times 2^{–23})$

Example



Sign: –
Exponent: 128 – 127 = 1
Mantissa: $1 + (2^{22} + 2^{21}) \times 2^{–23} = 1 + (2^{–1} + 2^{–2}) = 1 + 0.75 = 1.75$
Value: $–1.75 \times 2^{1} = –3.5$

Non-normalized Numbers

Non-normalized numbers express real numbers with very small absolute values. The sign bit will be 0 for a positive number and 1 for a negative number.

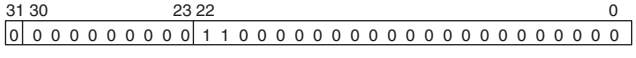
The exponent (e) will be 0, and the real exponent will be –126.

The mantissa (f) will be expressed from 1 to $2^{23} – 1$, and it is assume that, in the real mantissa, bit 2^{23} is 0 and the binary point follows immediately after it.

Non-normalized numbers are expressed as follows:

$(–1)^{(\text{sign s})} \times 2^{–126} \times (\text{mantissa} \times 2^{–23})$

Example



Sign: +
Exponent: –126
Mantissa: $0 + (2^{22} + 2^{21}) \times 2^{–23} = 0 + (2^{–1} + 2^{–2}) = 0 + 0.75 = 0.75$
Value: $0.75 \times 2^{–126}$

Zero

Values of +0.0 and –0.0 can be expressed by setting the sign to 0 for positive or 1 for negative. The exponent and mantissa will both be 0. Both +0.0 and –0.0 are equivalent to 0.0. Refer to *Floating-point Arithmetic Results*, below, for differences produced by the sign of 0.0.

Infinity

Values of +∞ and –∞ can be expressed by setting the sign to 0 for positive or 1 for negative. The exponent will be 255 ($2^8 - 1$) and the mantissa will be 0.

NaN

NaN (not a number) is produced when the result of calculations, such as 0.0/0.0, ∞/∞, or ∞–∞, does not correspond to a number or infinity. The exponent will be 255 ($2^8 - 1$) and the mantissa will be not 0.

Note There are no specifications for the sign of NaN or the value of the mantissa field (other than to be not 0).

Floating-point Arithmetic Results

Rounding Results

The following methods will be used to round results when the number of digits in the accurate result of floating-point arithmetic exceeds the significant digits of internal processing expressions.

If the result is close to one of two internal floating-point expressions, the closer expression will be used. If the result is midway between two internal floating-point expressions, the result will be rounded so that the last digit of the mantissa is 0.

Overflows, Underflows, and Illegal Calculations

Overflows will be output as either positive or negative infinity, depending on the sign of the result. Underflows will be output as either positive or negative zero, depending on the sign of the result.

Illegal calculations will result in NaN. Illegal calculations include adding infinity to a number with the opposite sign, subtracting infinity from a number with the same sign, multiplying zero and infinity, dividing zero by zero, or dividing infinity by infinity.

The value of the result may not be correct if an overflow occurs when converting a floating-point number to an integer.

Precautions in Handling Special Values

The following precautions apply to handling zero, infinity, and NaN.

- The sum of positive zero and negative zero is positive zero.
- The difference between zeros of the same sign is positive zero.
- If any operand is a NaN, the results will be a NaN.
- Positive zero and negative zero are treated as equivalent in comparisons.
- Comparison or equivalency tests on one or more NaN will always be true for != and always be false for all other instructions.

Floating-point Calculation Results

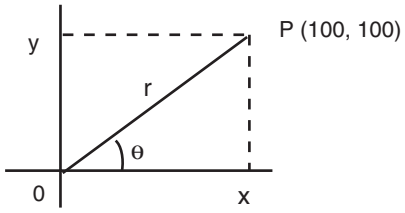
When the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞. If the result is positive, it will be output as +∞; if negative, then –∞.

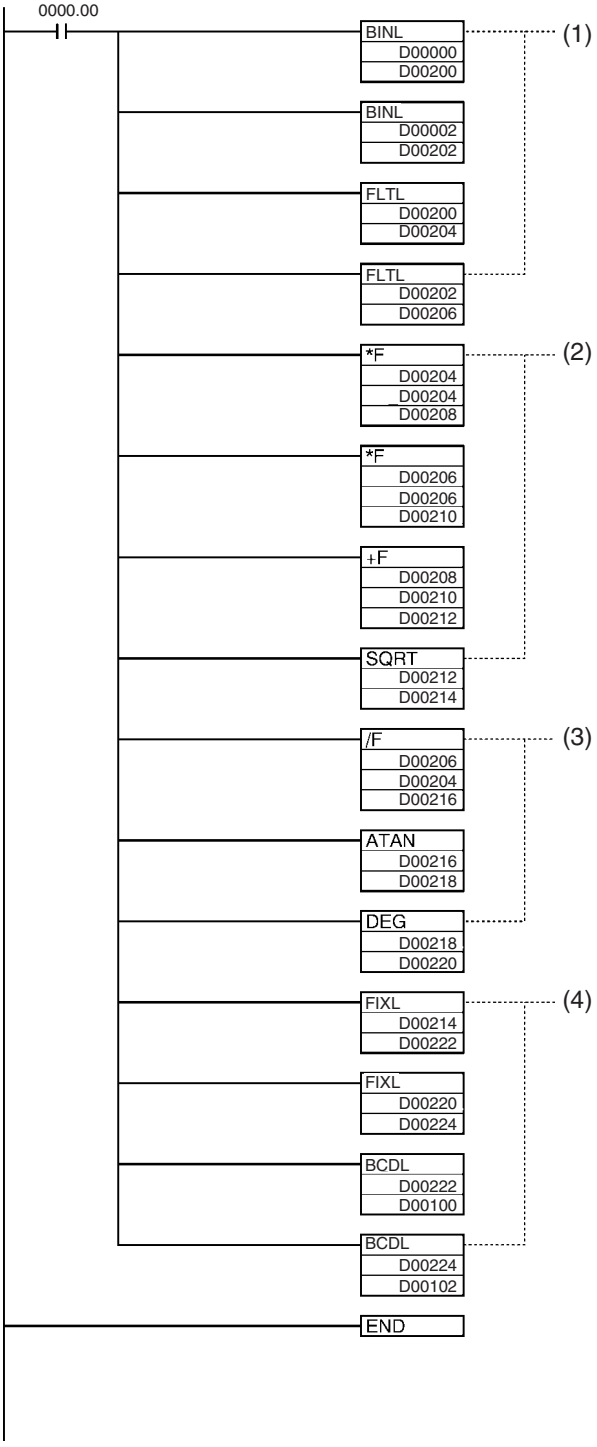
The Equals Flag will only turn ON when both the exponent (e) and the mantissa (f) are zero after a calculation. A calculation result will also be output as zero when the absolute value of the result is less than the minimum value that can be expressed for floating-point data. In that case the Underflow Flag will turn ON.

Example

In this program example, the X-axis and Y-axis coordinates (x, y) are provided by 8-digit BCD content of D00000, D00001 and D00002, D00003. The dis-

tance (r) from the origin and the angle (θ, in degrees) are found and output to D00100, D00101 and D00102, D00103. In the result, everything to the right of the decimal point is truncated.




0000.00
BINL
D00000
D00200
BINL
D00002
D00202
FLTL
D00200
D00204
FLTL
D00202
D00206
*F
D00204
D00204
D00208
*F
D00206
D00206
D00210
+F
D00208
D00210
D00212
SQRT
D00212
D00214
/F
D00206
D00204
D00216
ATAN
D00216
D00218
DEG
D00218
D00220
FIXL
D00214
D00222
FIXL
D00220
D00224
BCDL
D00222
D00100
BCDL
D00224
D00102
END
(1)
(2)
(3)
(4)

Calculations

Distance $r = \sqrt{x^2 + y^2}$

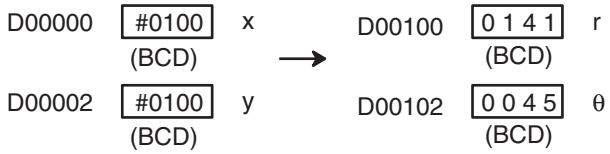
Angle $\theta = \tan^{-1}\left(\frac{y}{x}\right)$

Example

Distance $r = \sqrt{100^2 + 100^2} = 141.4214$

Angle $\theta = \tan^{-1}\left(\frac{100}{100}\right) = 45.0$

DM Contents



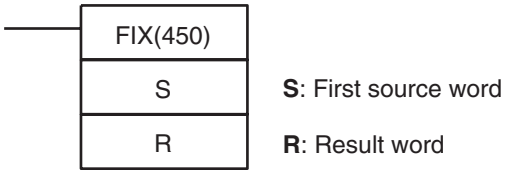
1. This section of the program converts the data from BCD to floating-point.
   a) The data area from D00200 onwards is used as a work area.
   b) First BINL(058) is used to temporarily convert the BCD data to binary data, and then FLTL(453) is used to convert the binary data to floating-point data.
   c) The value of x that has been converted to floating-point data is output to D00205 and D00204.
   d) The value of y that has been converted to floating-point data is output to D00207 and D00206.
2. In order to find the distance r, Floating-point Math Instructions are used to calculate the square root of x^2+y^2. The result is then output to D00215 and D00214 as floating-point data.
3. In order to find the angle θ, Floating-point Math Instructions are used to calculate $\tan^{-1}$ (y/x). ATAN(465) outputs the result in radians, so DEG(459) is used to convert to degrees. The result is then output to D00221 and D00220 as floating-point data.
4. The data is converted back from floating-point to BCD.
   a) First FIXL(451) is used to temporarily convert the floating-point data to binary data, and then BCDL(059) is used to convert the binary data to BCD data.
   b) The distance r is output to D00101 and D00100 in BCD.
   c) The angle θ is output to D00103 and D00102 in BCD.

3-14-1 FLOATING TO 16-BIT: FIX(450)

Purpose

Converts a 32-bit floating-point value to 16-bit signed binary data and places the result in the specified result word.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	FIX(450)
	Executed Once for Upward Differentiation	@FIX(450)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143
Work Area	W000 to W254	W000 to W255
Auxiliary Bit Area	A000 to A958	A448 to A959
Timer Area	T0000 to T0254	T0000 to T0255
Counter Area	C0000 to C0254	C0000 to C0255
DM Area	D00000 to D32766	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	DR0 to DR15
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

FIX(450) converts the integer portion of the 32-bit floating-point number in S+1 and S (IEEE754-format) to 16-bit signed binary data and places the result in

Only the integer portion of the floating-point data is converted, and the fraction portion is truncated. The integer portion of the floating-point data must be within the range of –32,768 to 32,767.

Example conversions:
A floating-point value of 3.5 is converted to 3.
A floating-point value of –3.5 is converted to –3.

Flags

Name	Label	Operation
Error Flag	ER	ON if the data in S+1 and S is not a number (NaN). ON if the integer portion of S+1 and S is not within the range of –32,768 to 32,767. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 of the result is ON. OFF in all other cases.

Precautions

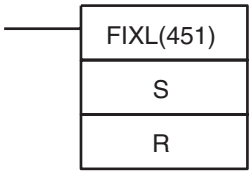
The content of S+1 and S must be floating-point data and the integer portion must be in the range of –32,768 to 32,767.

3-14-2 FLOATING TO 32-BIT: FIXL(451)

Purpose

Converts a 32-bit floating-point value to 32-bit signed binary data and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	FIXL(451)
	Executed Once for Upward Differentiation	@FIXL(451)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

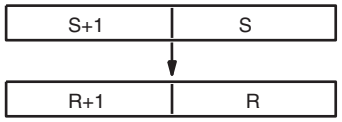
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –()IR15	

Description

FIXL(451) converts the integer portion of the 32-bit floating-point number in S+1 and S (IEEE754-format) to 32-bit signed binary data and places the result in R+1 and R.



Floating-point data (32 bits)

Signed binary data (32 bits)

Only the integer portion of the floating-point data is converted, and the fraction portion is truncated. (The integer portion of the floating-point data must be within the range of –2,147,483,648 to 2,147,483,647.)

Example conversions:
A floating-point value of 2,147,483,640.5 is converted to 2,147,483,640 in 32-bit signed binary.
A floating-point value of –214,748,340.5 is converted to –214,748,340 in 32-bit signed binary.

Flags

Name	Label	Operation
Error Flag	ER	ON if the data in S+1 and S is not a number (NaN). ON if the integer portion of S+1 and S is not within the range of –2,147,483,648 to 2,147,483,647. OFF in all other cases.
Equals Flag	=	ON if the result is 0000 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 of R+1 is ON after execution. OFF in all other cases.

Precautions

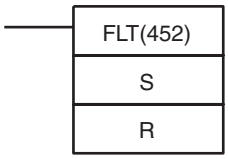
The content of S+1 and S must be floating-point data and the integer portion must be in the range of –2,147,483,648 to 2,147,483,647.

3-14-3 16-BIT TO FLOATING: FLT(452)

Purpose

Converts a 16-bit signed binary value to 32-bit floating-point data and places the result in the specified result words.

Ladder Symbol



S: Source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	FLT(452)
	Executed Once for Upward Differentiation	@FLT(452)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

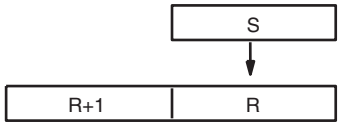
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6142
Work Area	W000 to W255	W000 to W254
Auxiliary Bit Area	A000 to A959	A448 to A958
Timer Area	T0000 to T0255	T0000 to T0254
Counter Area	C0000 to C0255	C0000 to C0254
DM Area	D00000 to D32767	D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	

Area	S	R
Constants	#0000 to #FFFF (binary)	---
Data Registers	DR0 to DR15	---
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

FLT(452) converts the 16-bit signed binary value in S to 32-bit floating-point data (IEEE754-format) and places the result in R+1 and R. A single 0 is added after the decimal point in the floating-point result.



Only values within the range of –32,768 to 32,767 can be specified for S. To convert signed binary data outside of that range, use FLTL(453).

Example conversions:
A signed binary value of 3 is converted to 3.0.
A signed binary value of –3 is converted to –3.0.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

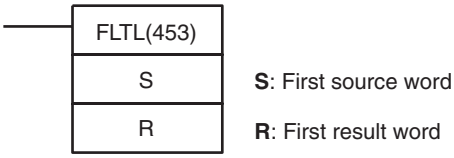
The content of S must contain signed binary data with a (decimal) value in the range of –32,768 to 32,767.

3-14-4 32-BIT TO FLOATING: FLTL(453)

Purpose

Converts a 32-bit signed binary value to 32-bit floating-point data and places the result in the specified result words.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	FLTL(453)
	Executed Once for Upward Differentiation	@FLTL(453)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

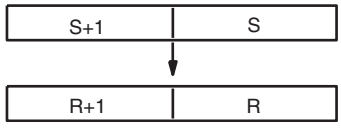
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	---
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

FLTL(453) converts the 32-bit signed binary value in S+1 and S to 32-bit floating-point data (IEEE754-format) and places the result in R+1 and R. A single 0 is added after the decimal point in the floating-point result.



Signed binary data within the range of –2,147,483,648 to 2,147,483,647 can be specified for S+1 and S. The floating point value has 24 significant binary digits (bits). The result will not be exact if a number greater than 16,777,215 (the maximum value that can be expressed in 24-bits) is converted by FLTL(453).

Example Conversions:
A signed binary value of 16,777,215 is converted to 16,777,215.0.
A signed binary value of –16,777,215 is converted to –16,777,215.0.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

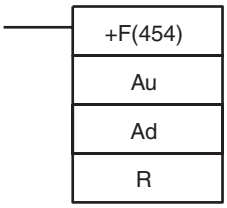
The result will not be exact if a number with an absolute value greater than 16,777,215 (the maximum value that can be expressed in 24-bits) is converted.

3-14-5 FLOATING-POINT ADD: +F(454)

Purpose

Adds two 32-bit floating-point numbers and places the result in the specified result words.

Ladder Symbol



Au: First augend word

AD: First addend word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	+F(454)
	Executed Once for Upward Differentiation	@+F(454)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

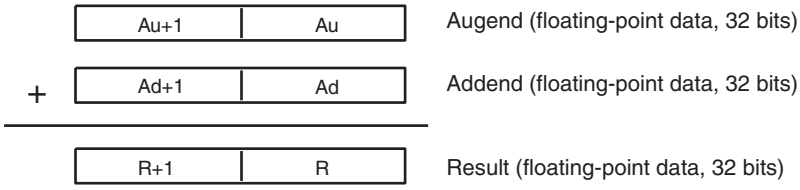
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)		---
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

+F(454) adds the 32-bit floating-point number in Ad+1 and Ad to the 32-bit floating-point number in Au+1 and Au and places the result in R+1 and R. (The floating point data must be in IEEE754 format.)



If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

The various combinations of augend and addend data will produce the results shown in the following table.

	Augend				
Addend	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	0	Numeral	+∞	–∞	---
Numeral	Numeral	See note 1.	+∞	–∞	---
+∞	+∞	+∞	+∞	See note 2.	---
–∞	–∞	–∞	See note 2.	–∞	---
NaN	---				See note 2.

Note

1. The results could be zero (including underflows), a numeral, +∞, or –∞.
2. The Error Flag will be turned ON and the instruction will not be executed.

Flags

Name	Label	Operation
Error Flag	ER	ON if the augend or addend data is not recognized as floating-point data. ON if the augend or addend data is not a number (NaN). ON if +∞ and –∞ are added. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

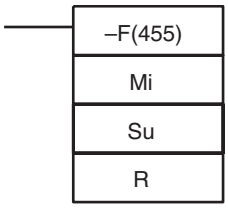
The augend (Au+1 and Au) and Addend (Ad+1 and Ad) data must be in IEEE754 floating-point data format.

3-14-6 FLOATING-POINT SUBTRACT: –F(455)

Purpose

Subtracts one 32-bit floating-point number from another and places the result in the specified result words.

Ladder Symbol



Mi: First Minuend word

Su: First Subtrahend word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	–F(455)
	Executed Once for Upward Differentiation	@–F(455)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

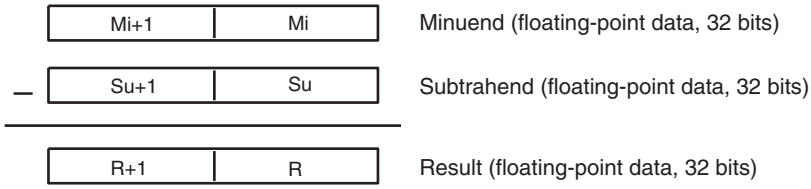
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)		
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

–F(455) subtracts the 32-bit floating-point number in Su+1 and Su from the 32-bit floating-point number in Mi+1 and Mi and places the result in R+1 and R. (The floating point data must be in IEEE754 format.)



If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

The various combinations of minuend and subtrahend data will produce the results shown in the following table.

	Minuend				
Subtrahend	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	0	Numeral	+∞	–∞	---
Numeral	Numeral	See note 1.	+∞	–∞	---
+∞	–∞	–∞	See note 2.	–∞	---
–∞	+∞	+∞	+∞	See note 2.	---
NaN	---				See note 2.

Note

1. The results could be zero (including underflows), a numeral, +∞, or –∞.
2. The Error Flag will be turned ON and the instruction will not be executed.

Flags

Name	Label	Operation
Error Flag	ER	ON if the minuend or subtrahend data is not recognized as floating-point data. ON if the minuend or subtrahend is not a number (NaN). ON if +∞ is subtracted from +∞. ON if –∞ is subtracted from –∞. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

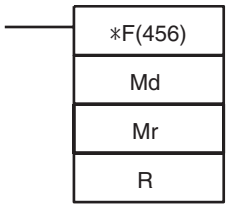
The Minuend (Mi+1 and Mi) and Subtrahend (Su+1 and Su) data must be in IEEE754 floating-point data format.

3-14-7 FLOATING-POINT MULTIPLY: ∗F(456)

Purpose

Multiplies two 32-bit floating-point numbers and places the result in the specified result words.

Ladder Symbol



Md: First Multiplicand word

Mr: First Multiplier word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	*F(456)
	Executed Once for Upward Differentiation	@*F(456)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

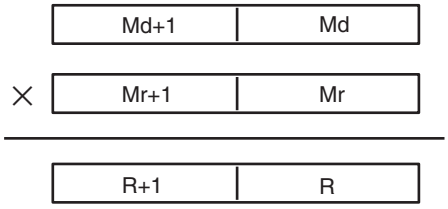
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Md	Mr	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)		---
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

∗F(456) multiplies the 32-bit floating-point number in Md+1 and Md by the 32-bit floating-point number in Mr+1 and Mr and places the result in R+1 and R. (The floating point data must be in IEEE754 format.)



If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

The various combinations of multiplicand and multiplier data will produce the results shown in the following table.

	Multiplicand				
Multiplier	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	0	0	See note 2.	See note 2.	---
Numeral	0	See note 1.	+/–∞	+/–∞	---
+∞	See note 2.	+/–∞	+∞	–∞	---
–∞	See note 2.	+/–∞	–∞	+∞	---
NaN	---				See note 2.

Note
1. The results could be zero (including underflows), a numeral, +∞, or –∞.
2. The Error Flag will be turned ON and the instruction will not be executed.

Flags

Name	Label	Operation
Error Flag	ER	ON if the multiplicand or multiplier data is not recognized as floating-point data. ON if the multiplicand or multiplier is not a number (NaN). ON if +∞ and 0 are multiplied. ON if –∞ and 0 are multiplied. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

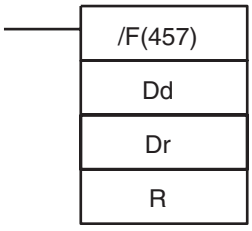
The Multiplicand (Md+1 and Md) and Multiplier (Mr+1 and Mr) data must be in IEEE754 floating-point data format.

3-14-8 FLOATING-POINT DIVIDE: /F(457)

Purpose

Divides one 32-bit floating-point number by another and places the result in the specified result words.

Ladder Symbol



Dd: First Dividend word

Dr: First Divisor word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	/F(457)
	Executed Once for Upward Differentiation	@/F(457)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

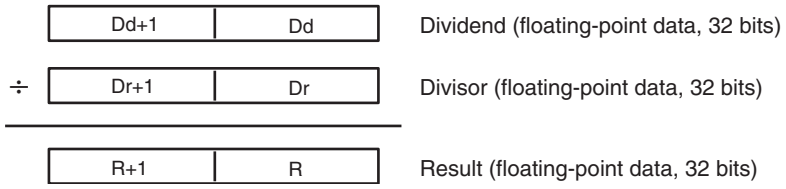
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Dd	Dr	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)		---
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

/F(457) divides the 32-bit floating-point number in Dd+1 and Dd by the 32-bit floating-point number in Dr+1 and Dr and places the result in R+1 and R. (The floating point data must be in IEEE754 format.)



If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

The various combinations of dividend and divisor data will produce the results shown in the following table.

	Dividend				
Divisor	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	See note 3.	+/–∞	+∞	–∞	---
Numeral	0	See note 1.	+/–∞	+/–∞	---
+∞	0	See note 2.	See note 3.	See note 3.	---
–∞	0	See note 2.	See note 3.	See note 3.	---
NaN	---				See note 3.

Note

1. The results could be zero (including underflows), a numeral, +∞, or –∞.
2. The results will be zero for underflows.
3. The Error Flag will be turned ON and the instruction will not be executed.

Flags

Name	Label	Operation
Error Flag	ER	ON if the dividend or divisor data is not recognized as floating-point data. ON if the dividend or divisor is not a number (NaN). ON if the dividend and divisor are both 0. ON if the dividend and divisor are both +∞ or –∞. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

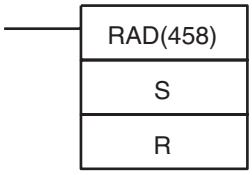
The Dividend (Dd+1 and Dd) and Divisor (Dr+1 and Dr) data must be in IEEE754 floating-point data format.

3-14-9 DEGREES TO RADIANS: RAD(458)

Purpose

Converts a 32-bit floating-point number from degrees to radians and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	RAD(458)
	Executed Once for Upward Differentiation	@RAD(458)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

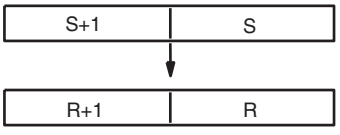
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	---
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

RAD(458) converts the 32-bit floating-point number in S+1 and S from degrees to radians and places the result in R and R+1. (The floating point source data must be in IEEE754 format.)



Source (degrees, 32-bit floating-point data)

Result (radians, 32-bit floating-point data)

Degrees are converted to radians by means of the following formula:

Degrees $\times \pi/180$ = radians

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

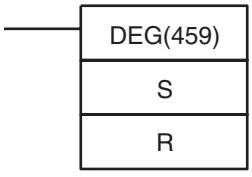
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-14-10 RADIANS TO DEGREES: DEG(459)

Purpose

Converts a 32-bit floating-point number from radians to degrees and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	DEG(459)
	Executed Once for Upward Differentiation	@DEG(459)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

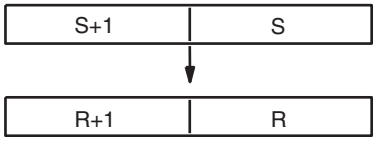
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	

Area	S	R
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 000 to #FFFF FFFF (binary)	---
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

DEG(459) converts the 32-bit floating-point number in S+1 and S from radians to degrees and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)



Source (radians, 32-bit floating-point data)

Result (degrees, 32-bit floating-point data)

Radians are converted to degrees by means of the following formula:

Radians $\times$ 180/π = degrees

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as $\pm\infty$.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

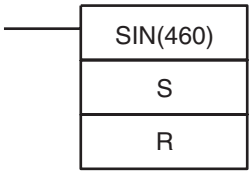
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-14-11 SINE: SIN(460)

Purpose

Calculates the sine of a 32-bit floating-point number (in radians) and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	SIN(460)
	Executed Once for Upward Differentiation	@SIN(460)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

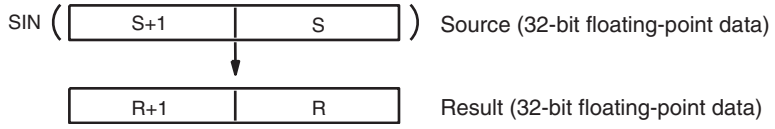
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	---
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

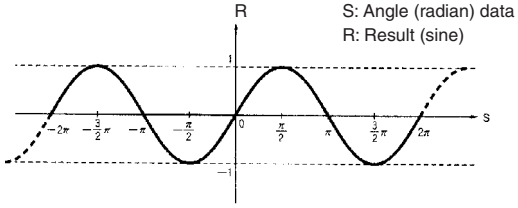
Description

SIN(460) calculates the sine of the angle (in radians) expressed as a 32-bit floating-point value in S+1 and S and places the result in R+1 and R.
(The floating point source data must be in IEEE754 format.)



Specify the desired angle (–65,535 to 65,535) in radians in S+1 and S. If the angle is outside of the range –65,535 to 65,535, an error will occur and the instruction will not be executed. For information on converting from degrees to radians, see *3-14-19 DEGREES-TO-RADIANS: RAD(458)*.

The following diagram shows the relationship between the angle and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 65,535. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	OFF
Underflow Flag	UF	OFF
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

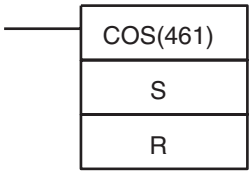
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-14-12 COSINE: COS(461)

Purpose

Calculates the cosine of a 32-bit floating-point number (in radians) and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	COS(461)
	Executed Once for Upward Differentiation	@COS(461)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

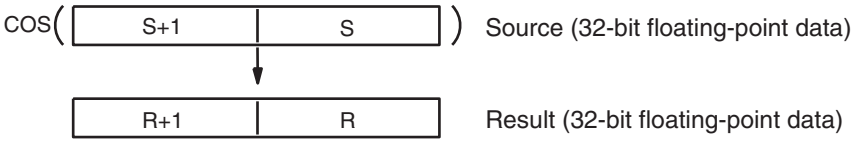
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	

Area	S	R
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	---
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

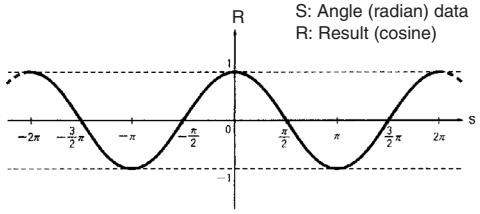
Description

COS(461) calculates the cosine of the angle (in radians) expressed as a 32-bit floating-point value in S+1 and S and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)



Specify the desired angle (–65,535 to 65,535) in radians in S+1 and S. If the angle is outside of the range –65,535 to 65,535, an error will occur and the instruction will not be executed. For information on converting from degrees to radians, see *3-14-9 DEGREES TO RADIANS: RAD(458)*.

The following diagram shows the relationship between the angle and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not a number (NaN). ON if the absolute value of the source data is not between 0 to 65,535. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	OFF
Underflow Flag	UF	OFF
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

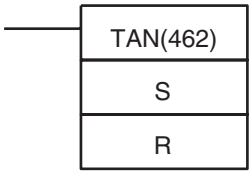
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-14-13 TANGENT: TAN(462)

Purpose

Calculates the tangent of a 32-bit floating-point number (in radians) and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	TAN(462)
	Executed Once for Upward Differentiation	@TAN(462)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

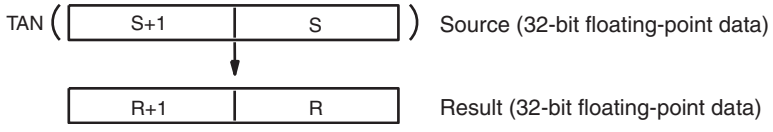
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	---

Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

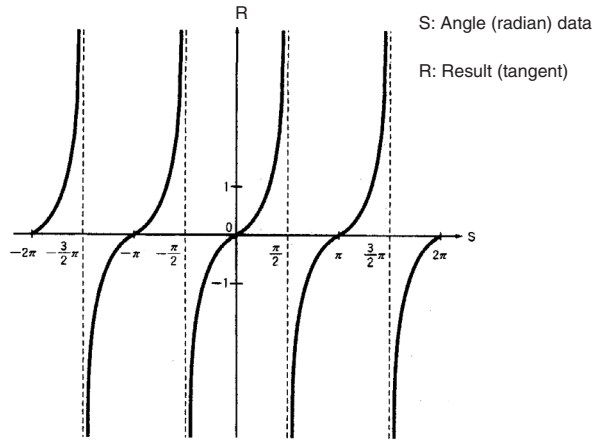
TAN(462) calculates the tangent of the angle (in radians) expressed as a 32-bit floating-point value in S+1 and S and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)



Specify the desired angle (–65,535 to 65,535) in radians in S+1 and S. If the angle is outside of the range –65,535 to 65,535, an error will occur and the instruction will not be executed. For information on converting from degrees to radians, see *3-14-9 DEGREES TO RADIANS: RAD(458)*.

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

The following diagram shows the relationship between the angle and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not a number (NaN). ON if the absolute value of the source data is not between 0 to 65,535. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	OFF
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

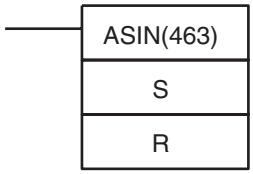
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-14-14 ARC SINE: ASIN(463)

Purpose

Calculates the arc sine of a 32-bit floating-point number and places the result in the specified result words. (The arc sine function is the inverse of the sine function; it returns the angle that produces a given sine value between –1 and 1.)

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	ASIN(463)
	Executed Once for Upward Differentiation	@ASIN(463)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

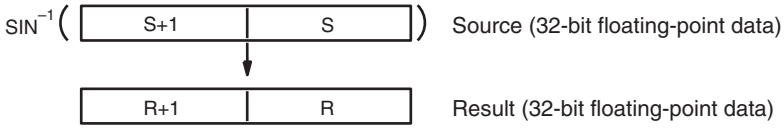
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	---
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

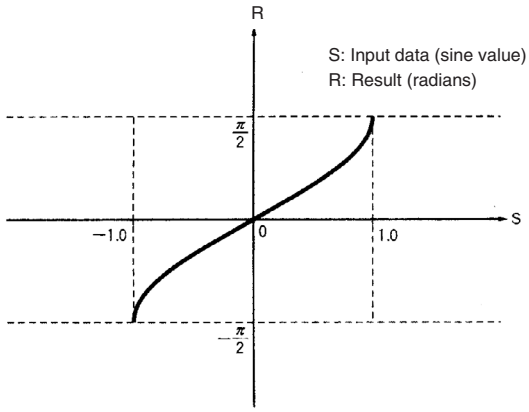
ASIN(463) computes the angle (in radians) for a sine value expressed as a 32-bit floating-point number in S+1 and S and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)



The source data must be between –1.0 and 1.0. If the absolute value of the source data exceeds 1.0, an error will occur and the instruction will not be executed.

The result is output to words R+1 and R as an angle (in radians) within the range of $-\pi/2$ to $\pi/2$.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 1.0. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	OFF
Underflow Flag	UF	OFF
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

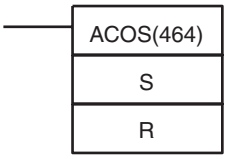
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-14-15 ARC COSINE: ACOS(464)

Purpose

Calculates the arc cosine of a 32-bit floating-point number and places the result in the specified result words. (The arc cosine function is the inverse of the cosine function; it returns the angle that produces a given cosine value between –1 and 1.)

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	ACOS(464)
	Executed Once for Upward Differentiation	@ACOS(464)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

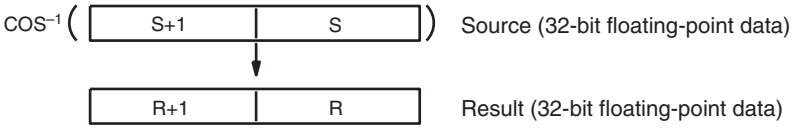
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	---
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

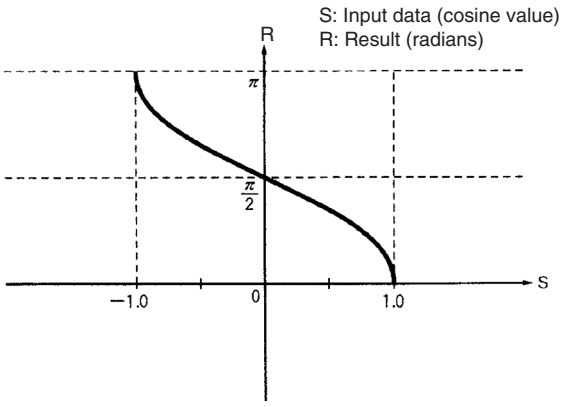
ACOS(464) computes the angle (in radians) for a cosine value expressed as a 32-bit floating-point number in S+1 and S and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)



The source data must be between –1.0 and 1.0. If the absolute value of the source data exceeds 1.0, an error will occur and the instruction will not be executed.

The result is output to words R+1 and R as an angle (in radians) within the range of 0 to π.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 1.0. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	OFF
Underflow Flag	UF	OFF
Negative Flag	N	OFF

Precautions

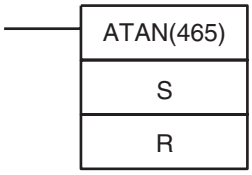
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-14-16 ARC TANGENT: ATAN(465)

Purpose

Calculates the arc tangent of a 32-bit floating-point number and places the result in the specified result words. (The arc tangent function is the inverse of the tangent function; it returns the angle that produces a given tangent value.)

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	ATAN(465)
	Executed Once for Upward Differentiation	@ATAN(465)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

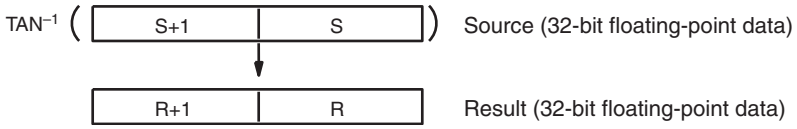
Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	---
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

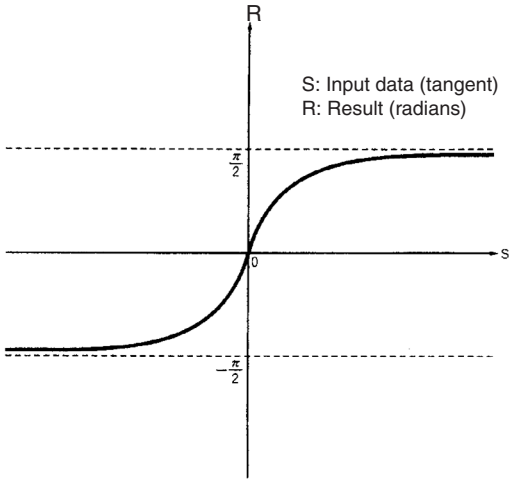
ATAN(465) computes the angle (in radians) for a tangent value expressed as a 32-bit floating-point number in S+1 and S and places the result in R+1 and R.

(The floating point source data must be in IEEE754 format.)



The result is output to words R+1 and R as an angle (in radians) within the range of $-\pi/2$ to $\pi/2$.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	OFF
Underflow Flag	UF	OFF
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

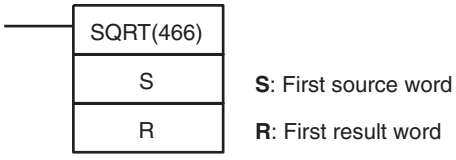
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-14-17 SQUARE ROOT: SQRT(466)

Purpose

Calculates the square root of a 32-bit floating-point number and places the result in the specified result words.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	SQRT(466)
	Executed Once for Upward Differentiation	@SQRT(466)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

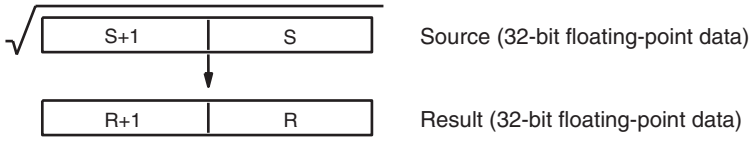
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	---
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

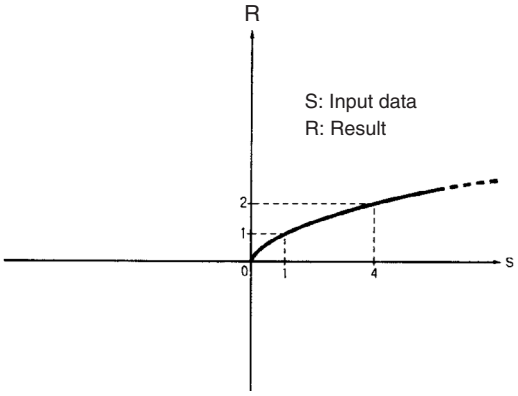
SQRT(466) calculates the square root of the 32-bit floating-point number in S+1 and S and places the result in R+1 and R. (The floating point source data must be in IEEE754 format.)



The source data must be positive; if it is negative, an error will occur and the instruction will not be executed.

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as +∞.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is negative. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	OFF
Negative Flag	N	OFF

Precautions

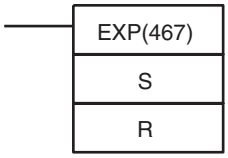
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-14-18 EXPONENT: EXP(467)

Purpose

Calculates the natural (base e) exponential of a 32-bit floating-point number and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	EXP(467)
	Executed Once for Upward Differentiation	@EXP(467)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

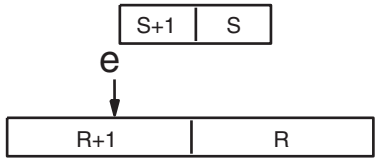
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	---
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

EXP(467) calculates the natural (base e) exponential of the 32-bit floating-point number in S+1 and S and places the result in R+1 and R. In other words, EXP(467) calculates e^x (x = source) and places the result in R+1 and R.



Source (32-bit floating-point data)

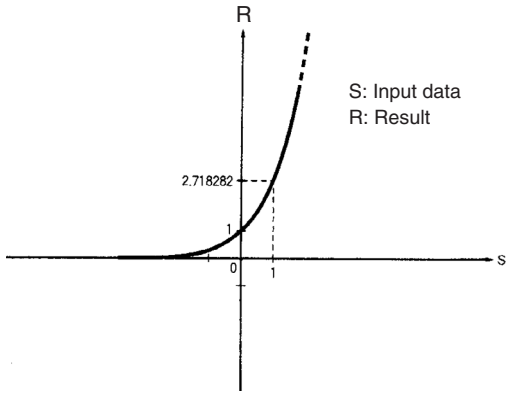
Result (32-bit floating-point data)

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as +∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

Note The constant e is 2.718282.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value.
Negative Flag	N	OFF

Precautions

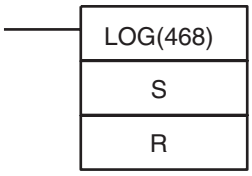
The source data in S+1 and S must be in IEEE754 floating-point data format.

3-14-19 LOGARITHM: LOG(468)

Purpose

Calculates the natural (base e) logarithm of a 32-bit floating-point number and places the result in the specified result words.

Ladder Symbol



S: First source word

R: First result word

Variations

Variations		
	Executed Each Cycle for ON Condition	LOG(468)
	Executed Once for Upward Differentiation	@LOG(468)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

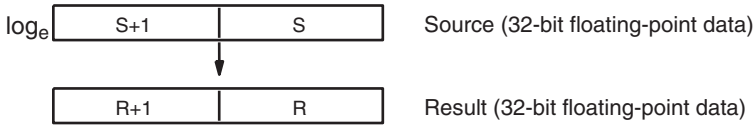
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	R
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	A448 to A958
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	---
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

LOG(468) calculates the natural (base e) logarithm of the 32-bit floating-point number in S+1 and S and places the result in R+1 and R.

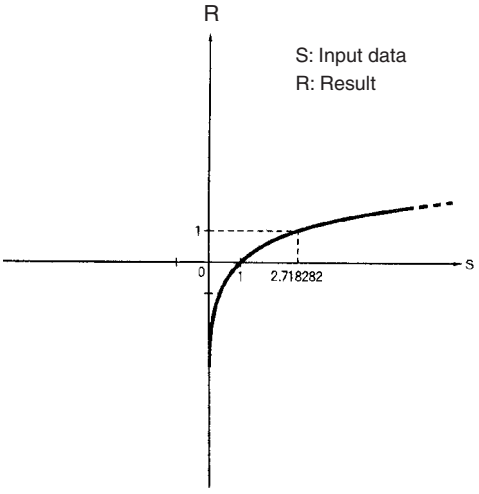


The source data must be positive; if it is negative, an error will occur and the instruction will not be executed.

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

Note The constant e is 2.718282.

The following diagram shows the relationship between the input data and result.



Flags

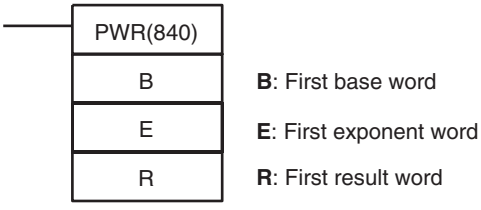
Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is negative. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	OFF
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions The source data in S+1 and S must be in IEEE754 floating-point data format.

3-14-20 EXPONENTIAL POWER: PWR(840)

Purpose Raises a 32-bit floating-point number to the power of another 32-bit floating-point number.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	PWR(840)
	Executed Once for Upward Differentiation	@PWR(840)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

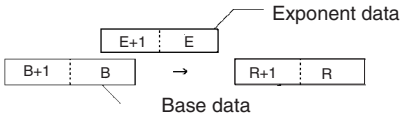
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	B	E	R
CIO Area	CIO 0000 to CIO 6142		
Work Area	W000 to W254		
Auxiliary Bit Area	A000 to A958		A448 to A958
Timer Area	T0000 to T0254		
Counter Area	C0000 to C0254		
DM Area	D00000 to D32766		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 0000 to #FFFF FFFF (binary)		---
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

PWR(840) raises the 32-bit floating-point number in B+1 and B to the power of the 32-bit floating-point number in E+1 and E. In other words, PWR(840) calculates X^Y (X = B+1 and B; Y = E+1 and E).



For example, when the base words (B+1 and B) contain 3.1 and the exponent words (E+1 and E) contain 3, the result is 3.1^3 or 29.791.

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON.

Flags

Name	Label	Operation
Error Flag	ER	ON if the base (B+1 and B) or exponent (E+1 and E) is not recognized as floating-point data. ON if the base (B+1 and B) or exponent (E+1 and E) is not a number (NaN). ON if the base (B+1 and B) is 0 and the exponent (E+1 and E) is less than 0. (Division by 0) ON if the base (B+1 and B) is negative and the exponent (E+1 and E) is non-integer. (Root of a negative number) OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.

Name	Label	Operation
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a 32-bit floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a 32-bit floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The base (B+1 and B) and the exponent (E+1 and E) must be in IEEE754 floating-point data format.

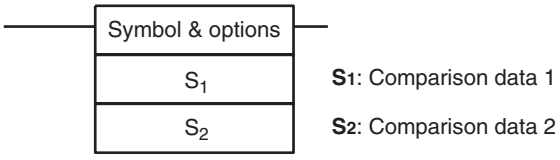
3-14-21 Single-precision Floating-point Comparison Instructions

Purpose

These input comparison instructions compare two single-precision floating point values (32-bit IEEE754 constants and/or the contents of specified words) and create an ON execution condition when the comparison condition is true.

Note Refer to *3-6-1 Input Comparison Instructions (300 to 328)* for details on the signed and unsigned binary input comparison instructions.

Ladder Symbol



Variations

Variations	Creates ON Each Cycle Comparison is True	Input comparison instruction

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S_1	S_2
CIO Area	CIO 0000 to CIO 6142	
Work Area	W000 to W254	
Auxiliary Bit Area	A000 to A958	
Timer Area	T0000 to T0254	
Counter Area	C0000 to C0254	
DM Area	D00000 to D32766	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 0000 to #FFFF FFFF (binary)	
Data Resisters	---	

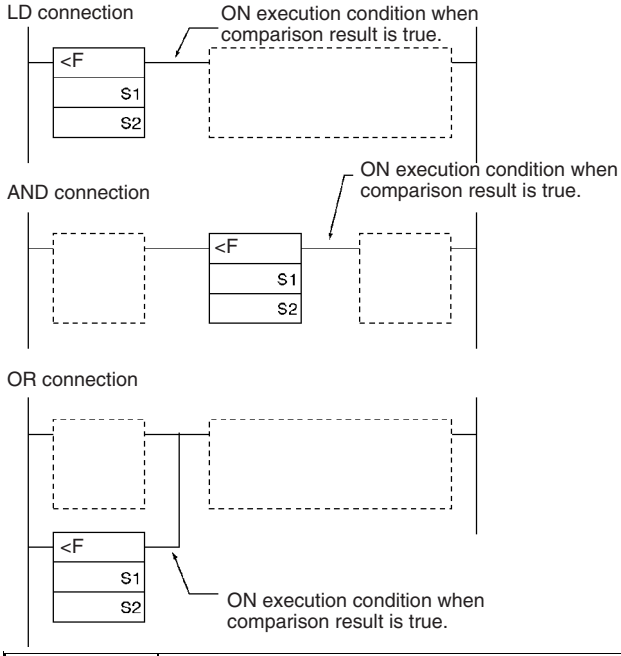
Area	S_1	S_2
Index Registers	IR0 or IR15	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

The input comparison instruction compares the data specified in S_1 and S_2 as single-precision floating point values (32-bit IEEE754 data) and creates an ON execution condition when the comparison condition is true. When the data is stored in words, S_1 and S_2 specify the first of two words containing the 32-bit data. It is also possible to input the floating-point data as an 8-digit hexadecimal constant in S_1 and S_2.

Inputting the Instructions

The input comparison instructions are treated just like the LD, AND, and OR instructions to control the execution of subsequent instructions.



Input type	Operation
LD	The instruction can be connected directly to the left bus bar.
AND	The instruction cannot be connected directly to the left bus bar.
OR	The instruction can be connected directly to the left bus bar.

Options

With the three input types and six symbols, there are 18 different possible combinations.

Symbol		Option (data format)
=	(Equal)	F: Single-precision floating-point data
< >	(Not equal)	
<	(Less than)	
<=	(Less than or equal)	
>	(Greater than)	
>=	(Greater than or equal)	

Summary of Input Comparison Instructions

The following table shows the function codes, mnemonics, names, and functions of the 18 single-precision floating-point input comparison instructions. (C1=S_1+1, S_1 and C2=S_2+1, S_2.)

Code	Mnemonic	Name	Function
329	LD=F	LOAD FLOATING EQUAL	True if C1 = C2
	AND=F	AND FLOATING EQUAL	
	OR=F	OR FLOATING EQUAL	
330	LD<>F	LOAD FLOATING NOT EQUAL	True if C1 ≠ C2
	AND<>F	AND FLOATING NOT EQUAL	
	OR<>F	OR FLOATING NOT EQUAL	
331	LD<F	LOAD FLOATING LESS THAN	True if C1 < C2
	AND<F	AND FLOATING LESS THAN	
	OR<F	OR FLOATING LESS THAN	
332	LD<=F	LOAD FLOATING LESS THAN OR EQUAL	True if C1 ≤ C2
	AND<=F	AND FLOATING LESS THAN OR EQUAL	
	OR<=F	OR FLOATING LESS THAN OR EQUAL	
333	LD>F	LOAD FLOATING GREATER THAN	True if C1 > C2
	AND>F	AND FLOATING GREATER THAN	
	OR>F	OR FLOATING GREATER THAN	
325	LD>=F	LOAD FLOATING GREATER THAN OR EQUAL	True if C1 ≥ C2
	AND>=F	AND FLOATING GREATER THAN OR EQUAL	
	OR>=F	OR FLOATING GREATER THAN OR EQUAL	

Flags

Name	Label	Operation
Error Flag	ER	ON if S_1+1, S_1 or S_2+1, S_2 is not a valid floating-point number (NaN). ON if S_1+1, S_1 or S_2+1, S_2 is +∞. ON if S_1+1, S_1 or S_2+1, S_2 is −∞. OFF in all other cases.
Greater Than Flag	>	ON if S_1+1, S_1 > S_2+1, S_2. OFF in all other cases.
Greater Than or Equal Flag	> =	ON if S_1+1, S_1 ≥ S_2+1, S_2. OFF in all other cases.
Equal Flag	=	ON if S_1+1, S_1 = S_2+1, S_2. OFF in all other cases.
Not Equal Flag	=	ON if S_1+1, S_1 ≠ S_2+1, S_2. OFF in all other cases.
Less Than Flag	<	ON if S_1+1, S_1 < S_2+1, S_2. OFF in all other cases.
Less Than or Equal Flag	< =	ON if S_1+1, S_1 ≤ S_2+1, S_2. OFF in all other cases.
Negative Flag	N	Unchanged

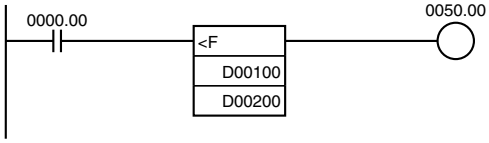
Precautions

Input comparison instructions cannot be used as right-hand instructions, i.e., another instruction must be used between them and the right bus bar.

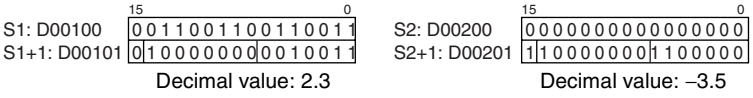
Example

AND FLOATING LESS THAN: AND<F(331)

When CIO 0000.00 is ON in the following example, the floating point data in D00101, D00100 is compared to the floating point data in D00201, D00200. If the content of D00101, D00100 is less than that of D00201, D00200, execution proceeds to the next line and CIO 0050.00 is turned ON. If the content of D00101, D00100 is not less than that of D00201, D00200, execution does not proceed to the next instruction line.

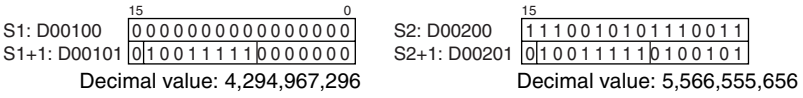


FLOATING LESS THAN Comparison (<F)



2.3>–3.5

Does not yield an ON condition.



4294967296<5566555656

Yields an ON condition.

3-15 Double-precision Floating-point Instructions

The Double-precision Floating-point Instructions convert data and perform floating-point arithmetic operations on double-precision floating-point data. The FQM1 Units support the following instructions.

Instruction	Mnemonic	Function code	Page
DOUBLE FLOATING TO 16-BIT	FIXD	841	430
DOUBLE FLOATING TO 32-BIT	FIXLD	842	432
16-BIT TO DOUBLE FLOATING	DBL	843	433
32-BIT TO DOUBLE FLOATING	DBLL	844	434
DOUBLE FLOATING-POINT ADD	+D	845	436
DOUBLE FLOATING-POINT SUBTRACT	–D	846	437
DOUBLE FLOATING-POINT MULTIPLY	*D	847	439
DOUBLE FLOATING-POINT DIVIDE	/D	848	441
DOUBLE DEGREES TO RADIANS	RADD	849	443
DOUBLE RADIANS TO DEGREES	DEGD	850	444
DOUBLE SINE	SIND	851	446
DOUBLE COSINE	COSD	852	447
DOUBLE TANGENT	TAND	853	449
DOUBLE ARC SINE	ASIND	854	450
DOUBLE ARC COSINE	ACOSD	855	452
DOUBLE ARC TANGENT	ATAND	856	454
DOUBLE SQUARE ROOT	SQRTD	857	456
DOUBLE EXPONENT	EXPD	858	457
DOUBLE LOGARITHM	LOGD	859	459
DOUBLE EXPONENTIAL POWER	PWRD	860	461
Double-precision Floating-point Symbol Comparison Instructions	LD, AND, OR + =D, <>D, <D, <=D, >D, or >=D	335 to 340	462

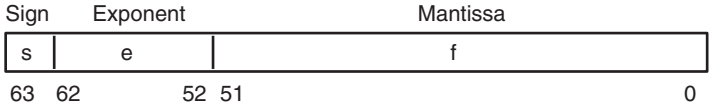
Data Format

Floating-point data expresses real numbers using a sign, exponent, and mantissa. When data is expressed in floating-point format, the following formula applies.

Real number = $(-1)^s \, 2^{e-1,023} \, (1.f)$

s: Sign
e: Exponent
f: Mantissa

The floating-point data format conforms to the IEEE754 standards. Data is expressed in 32 bits, as follows:



Data	No. of bits	Contents
s: sign	1	0: positive; 1: negative
e: exponent	11	The exponent (e) value ranges from 0 to 2,047. The actual exponent is the value remaining after 1,023 is subtracted from e, resulting in a range of –1,023 to 1,024. “e=0” and “e=2,047” express special numbers.
f: mantissa	52	The mantissa portion of binary floating-point data fits the format 2.0 > 1.f ≥1.0.

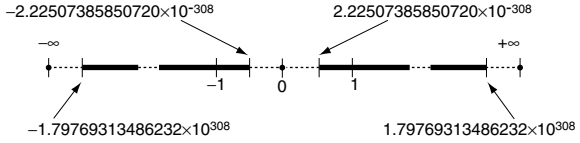
Number of Digits

The number of effective digits for floating-point data is 53 bits for binary (approximately 15 digits decimal).

Floating-point Data

The following data can be expressed by floating-point data:

- $-\infty$
- $-1.79769313486232 \times 10^{308} \leq$ value $\leq -2.22507385850720 \times 10^{-308}$
- 0
- $2.22507385850720 \times 10^{-308} \leq$ value $\leq 1.79769313486232 \times 10^{30}$
- $+\infty$
- Not a number (NaN)



Special Numbers

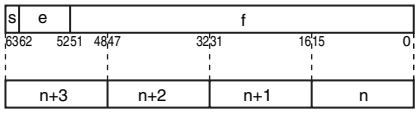
The formats for NaN, ±∞, and 0 are as follows:

NaN*: e = 1,024 and f ≠ 0
+∞: e = 1,024, f = 0, and s= 0
–∞: e = 1,024, f = 0, and s= 1
0: e = 0 and f = 0

*NaN (not a number) is not a valid floating-point number. Executing Double-precision Floating-point instructions will not result in NaN.

Writing Floating-point Data

When double-precision floating-point is specified for the data format in the I/O memory edit display in the CX-Programmer, standard decimal numbers input in the display are automatically converted to the double-precision floating-point format shown above (IEEE754-format) and written to I/O Memory. Data written in the IEEE754-format is automatically converted to standard decimal format when monitored on the display.



It is not necessary for the user to be aware of the IEEE754 data format when reading and writing double-precision floating-point data. It is only necessary to remember that double-precision floating point values occupy four words each.

Numbers Expressed as Floating-point Values

The following types of floating-point numbers can be used.

Mantissa (f)	Exponent (e)		
	0	Not 0 and not all 1's (1,024)	All 1's (1,024)
0	0	Normalized number	Infinity
Not 0	Non-normalized number		NaN

Note A non-normalized number is one whose absolute value is too small to be expressed as a normalized number. Non-normalized numbers have fewer significant digits. If the result of calculations is a non-normalized number (including intermediate results), the number of significant digits will be reduced.

Normalized Numbers

Normalized numbers express real numbers. The sign bit will be 0 for a positive number and 1 for a negative number.

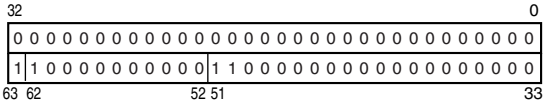
The exponent (e) will be expressed from 1 to 2,046, and the real exponent will be 1,023 less, i.e., –1,022 to 1,023.

The mantissa (f) will be expressed from 0 to $(2^{52} – 1)$, and it is assumed that, in the real mantissa, bit 2^{52} is 1 and the decimal point follows immediately after it.

Normalized numbers are expressed as follows:

$(–1)^{(\text{sign s})} \times 2^{(\text{exponent e})–1,023} \times (1 + \text{mantissa} \times 2^{–52})$

Example



Sign: –
Exponent: 1,024 – 1,023 = 1
Mantissa: $1 + (2^{51} + 2^{50}) \times 2^{–52} = 1 + (2^{–1} + 2^{–2}) = 1 + (0.75) = 1.75$
Value: $–1.75 \times 2^{1} = –3.5$

Non-normalized numbers

Non-normalized numbers express real numbers with very small absolute values. The sign bit will be 0 for a positive number and 1 for a negative number.

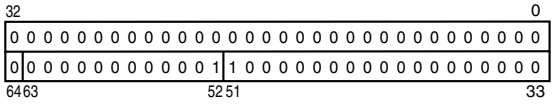
The exponent (e) will be 0, and the real exponent will be –1,022.

The mantissa (f) will be expressed from 1 to $(2^{52} – 1)$, and it is assumed that, in the real mantissa, bit 2^{52} is 0 and the decimal point follows immediately after it.

Non-normalized numbers are expressed as follows:

$(–1)^{(\text{sign s})} \times 2^{–1,022} \times (\text{mantissa} \times 2^{–52})$

Example



Sign: –
Exponent: –1,022
Mantissa: $0 + (2^{51} + 2^{50}) \times 2^{–52} = 0 + (2^{–1} + 2^{–2}) = 0 + (0.75) = 0.75$
Value: $–0.75 \times 2^{–1,022} = 1.668805 \times 10^{–308}$

Zero

Values of +0.0 and –0.0 can be expressed by setting the sign to 0 for positive or 1 for negative. The exponent and mantissa will both be 0. Both +0.0 and –

0.0 are equivalent to 0.0. Refer to *Floating-point Arithmetic Results*, below, for differences produced by the sign of 0.0.

Infinity

Values of +∞ and –∞ can be expressed by setting the sign to 0 for positive or 1 for negative. The exponent will be 2,047 ($2^{11} - 1$) and the mantissa will be 0.

NaN

NaN (not a number) is produced when the result of calculations, such as 0.0/0.0, ∞/∞, or ∞–∞, does not correspond to a number or infinity. The exponent will be 255 ($2^8 - 1$) and the mantissa will be not 0.

Note There are no specifications for the sign of NaN or the value of the mantissa field (other than to be not 0).

Floating-point Arithmetic Results

Rounding Results

The following methods will be used to round results when the number of digits in the accurate result of floating-point arithmetic exceeds the significant digits of internal processing expressions.

If the result is close to one of two internal floating-point expressions, the closer expression will be used. If the result is midway between two internal floating-point expressions, the result will be rounded so that the last digit of the mantissa is 0.

Overflows, Underflows, and Illegal Calculations

Overflows will be output as either positive or negative infinity, depending on the sign of the result. Underflows will be output as either positive or negative zero, depending on the sign of the result.

Illegal calculations will result in NaN. Illegal calculations include adding infinity to a number with the opposite sign, subtracting infinity from a number with the opposite sign, multiplying zero and infinity, dividing zero by zero, or dividing infinity by infinity.

The value of the result may not be correct if an overflow occurs when converting a floating-point number to an integer.

Precautions in Handling Special Values

The following precautions apply to handling zero, infinity, and NaN.

- The sum of positive zero and negative zero is positive zero.
- The difference between zeros of the same sign is positive zero.
- If any operand is a NaN, the results will be a NaN.
- Positive zero and negative zero are treated as equivalent in comparisons.
- Comparison or equivalency tests on one or more NaN will always be true for != and always be false for all other instructions.

Double-precision Floating-point Calculation Results

When the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞. If the result is positive, it will be output as +∞; if negative, then –∞.

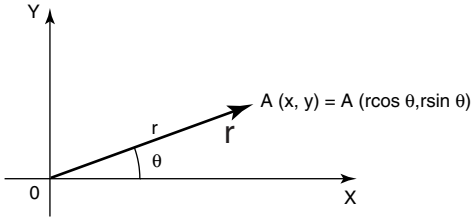
The Equals Flag will only turn ON when both the exponent (e) and the mantissa (f) are zero after a calculation. A calculation result will also be output as zero when the absolute value of the result is less than the minimum value that can be expressed for floating-point data. In that case the Underflow Flag will turn ON.

Comparing Single-precision and Double-precision Calculations

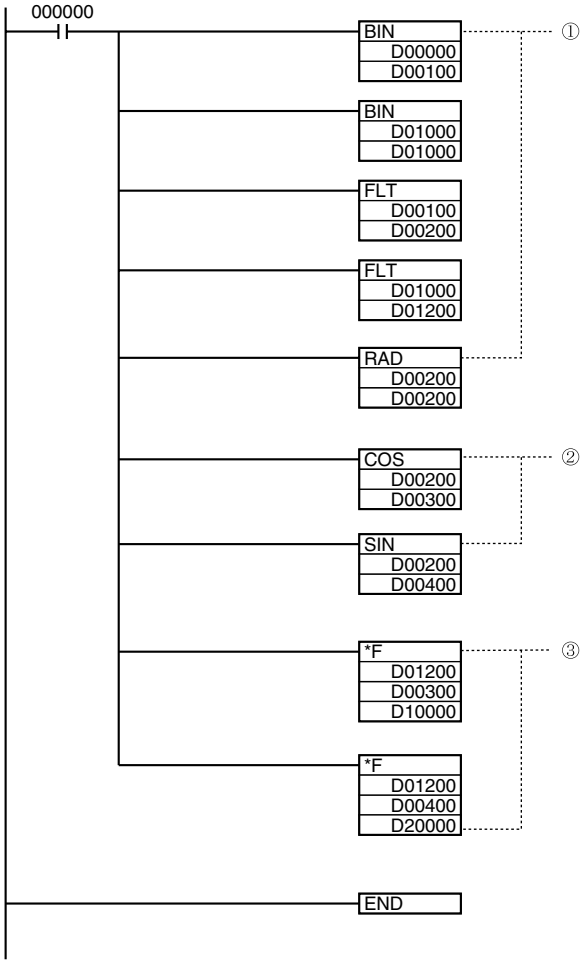
This example shows the differences in between single-precision and double-precision calculations when the following vector expressed in polar coordinates is converted to rectangular coordinates A (x,y).

$$r = re^{j\left(\frac{\pi}{360}\right)\theta}$$

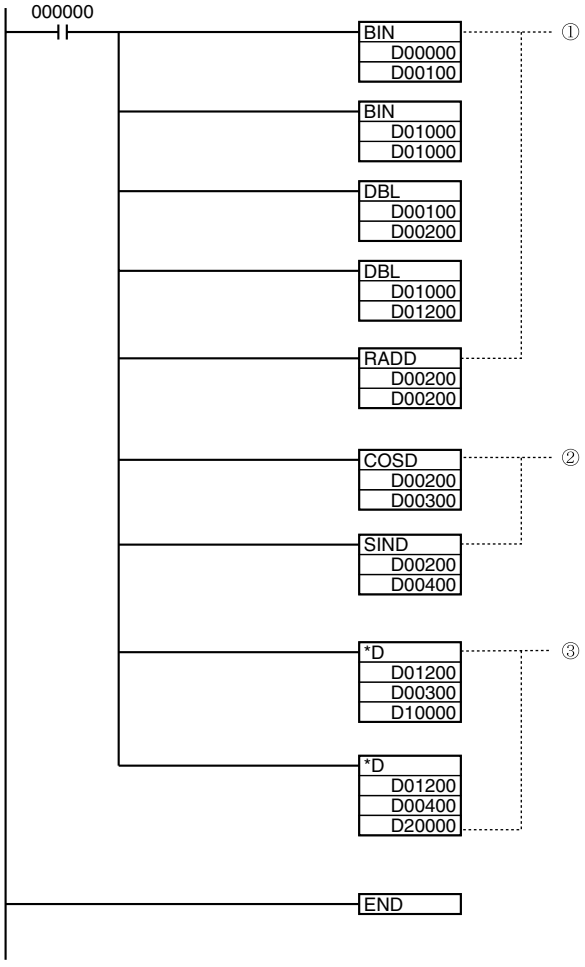
In this example, the 4-digit BCD angle (θ, in degrees) is read from D00000 and the 4-digit BCD distance (r) is read from D01000.



- Ladder Program for the Single-precision Calculation



- Ladder Program for the Double-precision Calculation



1. This program section converts the BCD data to single-precision floating-point data (32 bits, IEEE754-format).
   a) The BIN(023) instructions convert the BCD data to binary and the FLT(452) instructions convert the binary data to single-precision floating-point data.
   b) The floating-point data for the angle θ is output to D00200 and D00201.
   c) RAD(458) converts the angle data in D00200 and D00201 to radians.
   d) The floating-point data for the radius r is output to D01200 and D01201.
2. This program section calculates the sin θ and the cos θ as single-precision floating-point values.
   a) The value for cos θ is output to D00300 and D00301.
   b) The value for sin θ is output to D00400 and D00401.
3. This program section calculates x ($r \times \cos \theta$) and y ($r \times \sin \theta$).
   a) The value for x ($r \times \cos \theta$) is output to D10000 and D10001.
   b) The value for y ($r \times \sin \theta$) is output to D20000 and D20001.

Coordinate	Floating-point number	Real number
x	4116 59CF	3.4202015399933
y	405A E492	9.3969259262085

1. This program section converts the BCD data to double-precision floating-point data (64 bits, IEEE754-format).
   a) The BIN(023) instructions convert the BCD data to binary and the DBL(843) instructions convert the binary data to double-precision floating-point data.
   b) The floating-point data for the angle θ is output to words D00200 to D00203.
   c) RADD(849) converts the angle data in words D00200 to D00203 to radians.
   d) The floating-point data for the radius r is output to words D01200 to D01203.
2. This program section calculates the sin θ and the cos θ as double-precision floating-point values.
   a) The value for cos θ is output to words D00300 to D00303.
   b) The value for sin θ is output to words D00400 and D00403.
3. This program section calculates x ($r \times \cos \theta$) and y ($r \times \sin \theta$).
   a) The value for x ($r \times \cos \theta$) is output to words D10000 to D10003.
   b) The value for y ($r \times \sin \theta$) is output to D20000 and D20003.

Coordinate	Floating-point number	Real number
x	4022 CB39 E973 5C32	3.4202014332567
y	400B 5C92 91AC 8EF1	9.3969262078591

Comparison of the Calculation Results

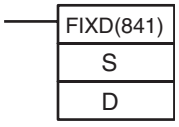
When the real-number results are compared, it is clear that the double-precision calculation yields a more accurate result.

3-15-1 DOUBLE FLOATING TO 16-BIT: FIXD(841)

Purpose

Converts a double-precision (64-bit) floating-point value to 16-bit signed binary data and places the result in the specified result word.

Ladder Symbol



S: First source word
D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	FIXD(841)
	Executed Once for Upward Differentiation	@FIXD(841)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

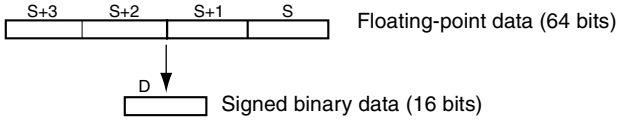
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	CIO 0000 to CIO 6143
Work Area	W000 to W252	W000 to W255
Auxiliary Bit Area	A000 to A956	A448 to A959
Timer Area	T0000 to T0252	T0000 to T0255
Counter Area	C0000 to C0252	C0000 to C0255
DM Area	D00000 to D32764	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Registers	---	DR0 to DR15
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

FIXD(841) converts the integer portion of the double-precision (64-bit) floating-point number in words S to S+3 (IEEE754-format) to 16-bit signed binary data and places the result in D.



Only the integer portion of the floating-point data is converted, and the fraction portion is truncated. The integer portion of the floating-point data must be within the range of –32,768 to 32,767.

Example conversions:
A floating-point value of 3.5 is converted to 3.
A floating-point value of –3.5 is converted to –3.

Flags

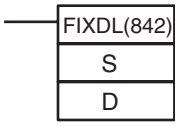
Name	Label	Operation
Error Flag	ER	ON if the source data (S to S+3) is not a number (NaN). ON if the integer portion of the source data (S to S+3) is not within the range of –32,768 to 32,767. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 of the result is ON. OFF in all other cases.

3-15-2 DOUBLE FLOATING TO 32-BIT: FIXLD(842)

Purpose

Converts a double-precision (64-bit) floating-point value to 32-bit signed binary data and places the result in the specified result words.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	FIXLD(842)
	Executed Once for Upward Differentiation	@FIXLD(842)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

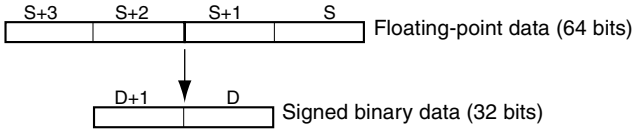
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	CIO 0000 to CIO 6142
Work Area	W000 to W252	W000 to W254
Auxiliary Bit Area	A000 to A956	A448 to A958
Timer Area	T0000 to T0252	T0000 to T0254
Counter Area	C0000 to C0252	C0000 to C0254
DM Area	D00000 to D32764	D00000 to D32766
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

FIXLD(842) converts the integer portion of the double-precision (64-bit) floating-point number in words S to S+3 (IEEE754-format) to 32-bit signed binary data and places the result in D+1 and D.



Only the integer portion of the floating-point data is converted, and the fraction portion is truncated. (The integer portion of the floating-point data must be within the range of –2,147,483,648 to 2,147,483,647.)

Example conversions:
A floating-point value of 2,147,483,640.5 is converted to 2,147,483,640 in 32-bit signed binary.
A floating-point value of –2,147,483,640.5 is converted to –2,147,483,640 in 32-bit signed binary.

Flags

Name	Label	Operation
Error Flag	ER	ON if the data in words S to S+3 is not a number (NaN). ON if the integer portion of words S to S+3 is not within the range of –2,147,483,648 to 2,147,483,647. OFF in all other cases.
Equals Flag	=	ON if the result is 0000 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 of D+1 is ON after execution. OFF in all other cases.

Precautions

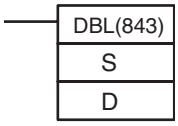
The content of words S to S+3 must be floating-point data and the integer portion must be in the range of –2,147,483,648 to 2,147,483,647.

3-15-3 16-BIT TO DOUBLE FLOATING: DBL(843)

Purpose

Converts a 16-bit signed binary value to double-precision (64-bit) floating-point data and places the result in the specified destination words.

Ladder Symbol



S: Source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	DBL(843)
	Executed Once for Upward Differentiation	@DBL(843)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

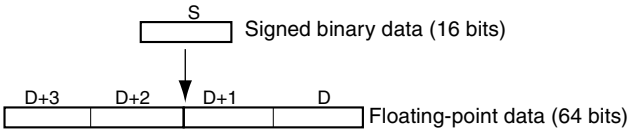
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6140
Work Area	W000 to W255	W000 to W252
Auxiliary Bit Area	A000 to A959	A448 to A956
Timer Area	T0000 to T0255	T0000 to T0252
Counter Area	C0000 to C0255	C0000 to C0252
DM Area	D00000 to D32767	D00000 to D32764
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#0000 to #FFFF (binary)	---

Area	S	D
Data Registers	DR0 to DR15	---
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

DBL(843) converts the 16-bit signed binary value in S to double-precision (64-bit) floating-point data (IEEE754-format) and places the result in words D to D+3. A single 0 is added after the decimal point in the floating-point result.



Only values within the range of –32,768 to 32,767 can be specified for S. To convert signed binary data outside of that range, use DBLL(844).

Example conversions:
A signed binary value of 3 is converted to 3.0.
A signed binary value of –3 is converted to –3.0.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

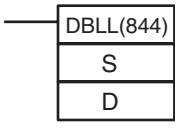
The content of S must contain signed binary data with a (decimal) value in the range of –32,768 to 32,767.

3-15-4 32-BIT TO DOUBLE FLOATING: DBLL(844)

Purpose

Converts a 32-bit signed binary value to double-precision (64-bit) floating-point data and places the result in the specified destination words.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	DBLL(844)
	Executed Once for Upward Differentiation	@DBLL(844)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

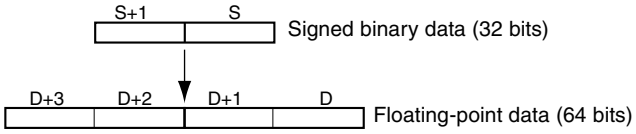
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6140
Work Area	W000 to W254	W000 to W252
Auxiliary Bit Area	A000 to A958	A448 to A956
Timer Area	T0000 to T0254	T0000 to T0252
Counter Area	C0000 to C0254	C0000 to C0252
DM Area	D00000 to D32766	D00000 to D32764
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	#00000000 to #FFFFFFFF (binary)	---
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

DBLL(844) converts the 32-bit signed binary value in S+1 and S to double-precision (64-bit) floating-point data (IEEE754-format) and places the result in words D to D+3. A single 0 is added after the decimal point in the floating-point result.



Signed binary data within the range of –2,147,483,648 to 2,147,483,647 can be specified for S+1 and S. The floating point value has 24 significant binary digits (bits). The result will not be exact if a number greater than 16,777,215 (the maximum value that can be expressed in 24-bits) is converted by DBLL(844).

Example Conversions:
A signed binary value of 16,777,215 is converted to 16,777,215.0.
A signed binary value of –16,777,215 is converted to –15,777,215.0.

Flags

Name	Label	Operation
Error Flag	ER	OFF
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

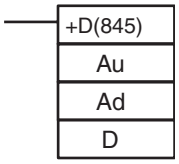
The result will not be exact if a number with an absolute value greater than 16,777,215 (the maximum value that can be expressed in 24-bits) is converted.

3-15-5 DOUBLE FLOATING-POINT ADD: +D(845)

Purpose

Adds two double-precision (64-bit) floating-point numbers and places the result in the specified destination words.

Ladder Symbol



Au: First augend word
Ad: First addend word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	+D(845)
	Executed Once for Upward Differentiation	@+D(845)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Au	Ad	D
CIO Area	CIO 0000 to CIO 6140		
Work Area	W000 to W252		
Auxiliary Bit Area	A000 to A956		A448 to A956
Timer Area	T0000 to T0252		
Counter Area	C0000 to C0252		
DM Area	D00000 to D32764		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

+D(845) adds the double-precision (64-bit) floating-point number in words Ad to Ad+3 the double-precision (64-bit) floating-point number in words Au to Au+3 and places the result in words D to D+3. (The floating point data must be in IEEE754 format.)

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

The various combinations of augend and addend data will produce the results shown in the following table.

	Augend				
Addend	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	0	Numeral	+∞	–∞	
Numeral	Numeral	See note 1.	+∞	–∞	
+∞	+∞	+∞	+∞	See note 2.	
–∞	–∞	–∞	See note 2.	–∞	
NaN					See note 2.

Note

1. The results could be zero (including underflows), a numeral, +∞, or –∞.
2. The Error Flag will be turned ON and the instruction will not be executed.

Flags

Name	Label	Operation
Error Flag	ER	ON if the augend or addend data is not recognized as floating-point data. ON if the augend or addend data is not a number (NaN). ON if +∞ is to –∞. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

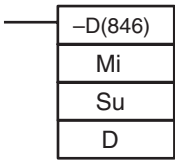
The augend (Au to Au+3) and Addend (Ad to Ad+3) data must be in IEEE754 floating-point data format.

3-15-6 DOUBLE FLOATING-POINT SUBTRACT: –D(846)

Purpose

Subtracts one double-precision (64-bit) floating-point number from another and places the result in the specified destination words.

Ladder Symbol



Mi: First Minuend word
Su: First Subtrahend word
D: First destination word

Variations

Variations	Executed Each Cycle for ON Condition	–D(846)
	Executed Once for Upward Differentiation	@–D(846)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

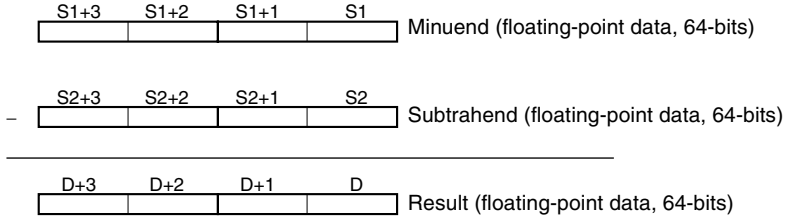
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Mi	Su	D
CIO Area	CIO 0000 to CIO 6140		
Work Area	W000 to W252		
Auxiliary Bit Area	A000 to A956		A448 to A956
Timer Area	T0000 to T0252		
Counter Area	C0000 to C0252		
DM Area	D00000 to D32764		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

–D(846) subtracts the double-precision (64-bit) floating-point number in words Su to Su+3 from the double-precision (64-bit) floating-point number in Mi to Mi+3 and places the result in words D to D+3. (The floating point data must be in IEEE754 format.)



If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

The various combinations of minuend and subtrahend data will produce the results shown in the following table.

	Minuend				
Subtrahend	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	0	Numeral	+∞	–∞	
Numeral	Numeral	See note 1.	+∞	–∞	
+∞	–∞	–∞	See note 2.	–∞	
–∞	+∞	+∞	+∞	See note 2.	
NaN					See note 2.

Note
1. The results could be zero (including underflows), a numeral, +∞, or –∞.
2. The Error Flag will be turned ON and the instruction will not be executed.

Flags

Name	Label	Operation
Error Flag	ER	ON if the minuend or subtrahend data is not recognized as floating-point data. ON if the minuend or subtrahend is not a number (NaN). ON if +∞ is subtracted from +∞. ON if –∞ is subtracted from –∞. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

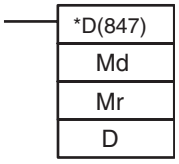
The Minuend (Mi to Mi+3) and Subtrahend (Su to Su+3) data must be in IEEE754 floating-point data format.

3-15-7 DOUBLE FLOATING-POINT MULTIPLY: ✳D(847)

Purpose

Multiplies two double-precision (64-bit) floating-point numbers and places the result in the specified result words.

Ladder Symbol



Md: First Multiplicand word
Mr: First Multiplier word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	*D(847)
	Executed Once for Upward Differentiation	@*D(847)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

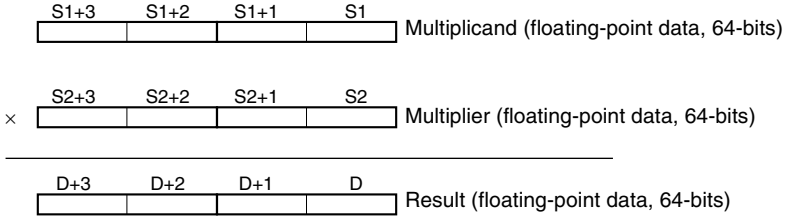
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	Md	Mr	D
CIO Area	CIO 0000 to CIO 6140		
Work Area	W000 to W252		
Auxiliary Bit Area	A000 to A956		A448 to A956
Timer Area	T0000 to T0252		
Counter Area	C0000 to C0252		
DM Area	D00000 to D32764		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

∗D(847) multiplies the double-precision (64-bit) floating-point number in words Md to Md+3 by the double-precision (64-bit) floating-point number in words Mr to Mr+3 and places the result in words D to D+3. (The floating point data must be in IEEE754 format.)



If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

The various combinations of multiplicand and multiplier data will produce the results shown in the following table.

	Multiplicand				
Multiplier	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	0	0	See note 2.	See note 2.	
Numeral	0	See note 1.	+/–∞	+/–∞	
+∞	See note 2.	+/–∞	+∞	–∞	
–∞	See note 2	+/–∞	–∞	+∞	
NaN					See note 2.

Note

1. The results could be zero (including underflows), a numeral, +∞, or –∞.
2. The Error Flag will be turned ON and the instruction will not be executed.

Flags

Name	Label	Operation
Error Flag	ER	ON if the multiplicand or multiplier data is not recognized as floating-point data. ON if the multiplicand or multiplier is not a number (NaN). ON if +∞ and 0 are multiplied. ON if –∞ and 0 are multiplied. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

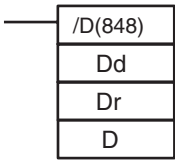
The Multiplicand (Md to Md+3) and Multiplier (Mr to Mr+3) data must be in IEEE754 floating-point data format.

3-15-8 DOUBLE FLOATING-POINT DIVIDE: /D(848)

Purpose

Divides one double-precision (64-bit) floating-point number by another and places the result in the specified destination words.

Ladder Symbol



Dd: First Dividend word
Dr: First Divisor word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	/D(848)
	Executed Once for Upward Differentiation	@/D(848)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

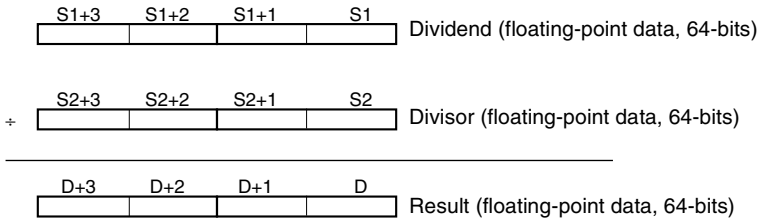
Operand Specifications

Area	Dd	Dr	D
CIO Area	CIO 0000 to CIO 6140		
Work Area	W000 to W252		
Auxiliary Bit Area	A000 to A956		A448 to A956
Timer Area	T0000 to T0252		
Counter Area	C0000 to C0252		
DM Area	D00000 to D32764		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---		

Area	Dd	Dr	D
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

/D(848) divides the double-precision (64-bit) floating-point number in words Dd to Dd+3 by the double-precision (64-bit) floating-point number in words Dr to Dr+3 and places the result in words D to D+3. (The floating point data must be in IEEE754 format.)



If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

The various combinations of dividend and divisor data will produce the results shown in the following table.

	Dividend				
Divisor	**0**	**Numeral**	**+∞**	**–∞**	**NaN**
0	See note 3.	+/–∞	+∞	–∞	
Numeral	0	See note 1.	+/–∞	+/–∞	
+∞	0	See note 2.	See note 3.	See note 3.	
–∞	0	See note 2.	See note 3.	See note 3.	
NaN					See note 3.

Note

1. The results could be zero (including underflows), a numeral, +∞, or –∞.
2. The results will be zero for underflows.
3. The Error Flag will be turned ON and the instruction will not be executed.

Flags

Name	Label	Operation
Error Flag	ER	ON if the dividend or divisor data is not recognized as floating-point data. ON if the dividend or divisor is not a number (NaN). ON if the dividend and divisor are both 0. ON if the dividend and divisor are both +∞ or –∞. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value.

Name	Label	Operation
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

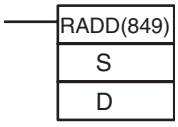
The Dividend (Dd to Dd+3) and Divisor (Dr to Dr+3) data must be in IEEE754 floating-point data format.

3-15-9 DOUBLE DEGREES TO RADIANS: RADD(849)

Purpose

Converts a double-precision (64-bit) floating-point number from degrees to radians and places the result in the specified result words.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	RADD(849)
	Executed Once for Upward Differentiation	@RADD(849)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

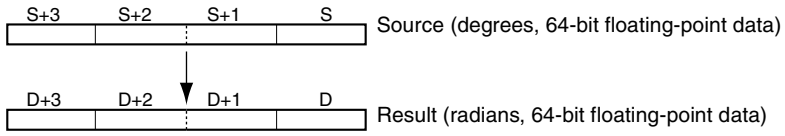
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W252	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T0252	
Counter Area	C0000 to C0252	
DM Area	D00000 to D32764	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

RADD(849) converts the double-precision (64-bit) floating-point number in words S to S+3 from degrees to radians and places the result in words D to D+3. (The floating point source data must be in IEEE754 format.)



Degrees are converted to radians by means of the following formula:

Degrees $\times \pi/180$ = radians

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as $\pm\infty$.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

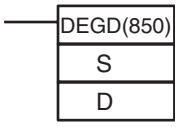
The source data in words S to S+3 must be in IEEE754 floating-point data format.

3-15-10 DOUBLE RADIANS TO DEGREES: DEGD(850)

Purpose

Converts a double-precision (64-bit) floating-point number from radians to degrees and places the result in the specified result words.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	DEGD(850)
	Executed Once for Upward Differentiation	@DEGD(850)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

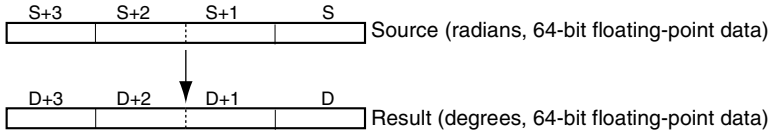
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W252	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T0252	
Counter Area	C0000 to C0252	
DM Area	D00000 to D32764	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

DEGD(850) converts the double-precision (64-bit) floating-point number in words S to S+3 from radians to degrees and places the result in words D to D+3. (The floating point source data must be in IEEE754 format.)



Radians are converted to degrees by means of the following formula:

Radians $\times$ 180/π = degrees

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as $\pm\infty$.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

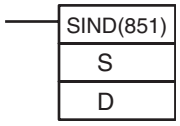
The source data in words S to S+3 must be in IEEE754 floating-point data format.

3-15-11 DOUBLE SINE: SIND(851)

Purpose

Calculates the sine of a double-precision (64-bit) floating-point number (in radians) and places the result in the specified destination words.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	SIND(851)
	Executed Once for Upward Differentiation	@SIND(851)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

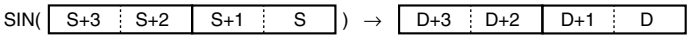
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W252	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T0252	
Counter Area	C0000 to C0252	
DM Area	D00000 to D32764	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

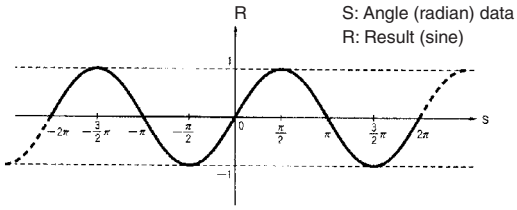
SIND(851) calculates the sine of the angle (in radians) expressed as a double-precision (64-bit) floating-point value in words S to S+3 and places the result in words D to D+3.
(The floating point source data must be in IEEE754 format.)



Specify the desired angle (–65,535 to 65,535) in radians in words S to S+3. If the angle is outside of the range –65,535 to 65,535, an error will occur and the instruction will not be executed. For information on converting between

degrees and radians, see *3-15-9 DOUBLE DEGREES TO RADIANS: RADD(849)* or *3-15-10 DOUBLE RADIANS TO DEGREES: DEGD(850)*.

The following diagram shows the relationship between the angle and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 65,535. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	Unchanged
Underflow Flag	UF	Unchanged
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

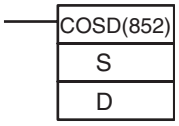
The source data in words S to S+3 must be in IEEE754 floating-point data format.

3-15-12 DOUBLE COSINE: COSD(852)

Purpose

Calculates the cosine of a double-precision (64-bit) floating-point number (in radians) and places the result in the specified destination words.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	COSD(852)
	Executed Once for Upward Differentiation	@COSD(852)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

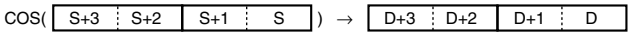
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W252	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T0252	
Counter Area	C0000 to C0252	
DM Area	D00000 to D32764	

Area	S	D
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

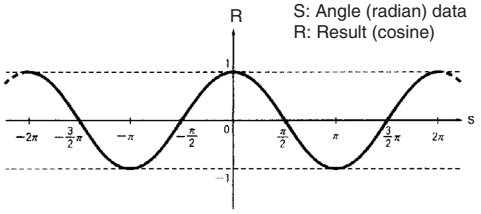
Description

COSD(852) calculates the cosine of the angle (in radians) expressed as a double-precision (64-bit) floating-point value in words S to S+3 and places the result in words D to D+3.
(The floating point source data must be in IEEE754 format.)



Specify the desired angle (–65,535 to 65,535) in radians in words S to S+3. If the angle is outside of the range –65,535 to 65,535, an error will occur and the instruction will not be executed. For information on converting between degrees and radians, see *3-15-9 DOUBLE DEGREES TO RADIANS: RADD(849)* or *3-15-10 DOUBLE RADIANS TO DEGREES: DEGD(850)*.

The following diagram shows the relationship between the angle and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 65,535. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	Unchanged
Underflow Flag	UF	Unchanged
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

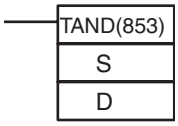
The source data in words S to S+3 must be in IEEE754 floating-point data format.

3-15-13 DOUBLE TANGENT: TAND(853)

Purpose

Calculates the tangent of a double-precision (64-bit) floating-point number (in radians) and places the result in the specified destination words.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	TAND(853)
	Executed Once for Upward Differentiation	@TAND(853)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

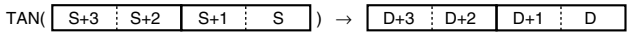
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W252	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T0252	
Counter Area	C0000 to C0252	
DM Area	D00000 to D32764	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

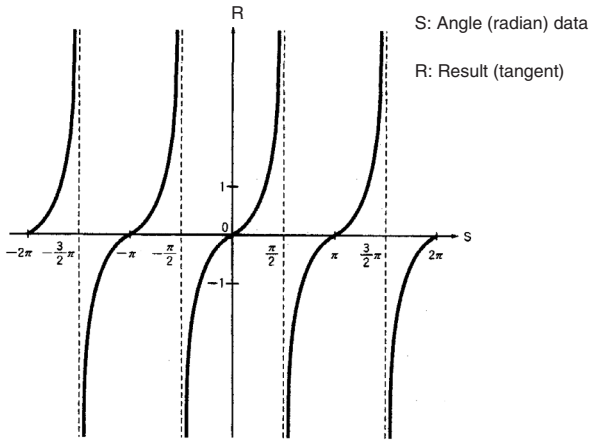
TAND(853) calculates the tangent of the angle (in radians) expressed as a double-precision (64-bit) floating-point value in words S to S+3 and places the result in words D to D+3.
(The floating point source data must be in IEEE754 format.)



Specify the desired angle (–65,535 to 65,535) in radians in words S to S+3. If the angle is outside of the range –65,535 to 65,535, an error will occur and the instruction will not be executed. For information on converting between degrees and radians, see *3-15-9 DOUBLE DEGREES TO RADIANS: RADD(849)* or *3-15-10 DOUBLE RADIANS TO DEGREES: DEGD(850)*.

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

The following diagram shows the relationship between the angle and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 65,535. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision (64-bit) floating-point value.
Underflow Flag	UF	Unchanged
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

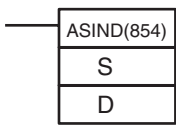
The source data in words S to S+3 must be in IEEE754 floating-point data format.

3-15-14 DOUBLE ARC SINE: ASIND(854)

Purpose

Calculates the arc sine of a double-precision (64-bit) floating-point number and places the result in the specified destination words. (The arc sine function is the inverse of the sine function; it returns the angle that produces a given sine value between –1 and 1.)

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	ASIND(854)
	Executed Once for Upward Differentiation	@ASIND(854)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

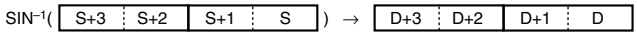
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W252	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T0252	
Counter Area	C0000 to C0252	
DM Area	D00000 to D32764	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

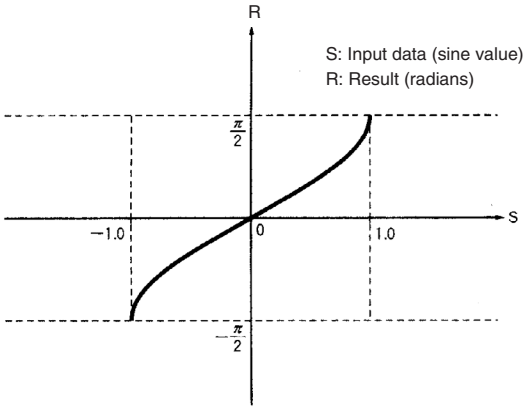
ASIND(854) computes the angle (in radians) for a sine value expressed as a double-precision (64-bit) floating-point number in words S to S+3 and places the result in words D to D+3.
(The floating point source data must be in IEEE754 format.)



The source data must be between –1.0 and 1.0. If the absolute value of the source data exceeds 1.0, an error will occur and the instruction will not be executed.

The result is output to words D to D+3 as an angle (in radians) within the range of $-\pi/2$ to $\pi/2$.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 1.0. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	Unchanged
Underflow Flag	UF	Unchanged
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

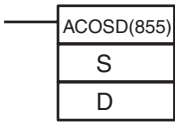
The source data in words S to S+3 must be in IEEE754 floating-point data format.

3-15-15 DOUBLE ARC COSINE: ACOSD(855)

Purpose

Calculates the arc cosine of a double-precision (64-bit) floating-point number and places the result in the specified result words. (The arc cosine function is the inverse of the cosine function; it returns the angle that produces a given cosine value between –1 and 1.)

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	ACOSD(855)
	Executed Once for Upward Differentiation	@ACOSD(855)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

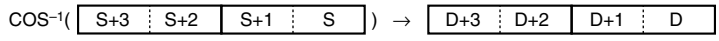
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W252	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T0252	
Counter Area	C0000 to C0252	
DM Area	D00000 to D32764	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

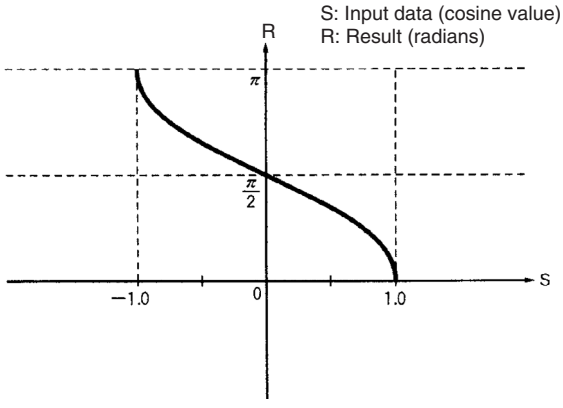
ACOSD(855) computes the angle (in radians) for a cosine value expressed as a double-precision (64-bit) floating-point number in words S to S+3 and places the result in words D to D+3.
(The floating point source data must be in IEEE754 format.)



The source data must be between –1.0 and 1.0. If the absolute value of the source data exceeds 1.0, an error will occur and the instruction will not be executed.

The result is output to words D to D+3 as an angle (in radians) within the range of 0 to π.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). ON if the absolute value of the source data exceeds 1.0. OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	Unchanged
Underflow Flag	UF	Unchanged
Negative Flag	N	Unchanged

Precautions

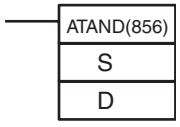
The source data in words S to S+3 must be in IEEE754 floating-point data format.

3-15-16 DOUBLE ARC TANGENT: ATAND(856)

Purpose

Calculates the arc tangent of a double-precision (64-bit) floating-point number and places the result in the specified result words. (The arc tangent function is the inverse of the tangent function; it returns the angle that produces a given tangent value.)

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	ATAND(856)
	Executed Once for Upward Differentiation	@ATAND(856)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

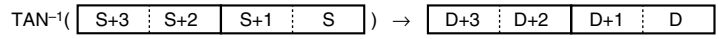
Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W252	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T0252	
Counter Area	C0000 to C0252	
DM Area	D00000 to D32764	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Registers	---	

Area	S	D
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

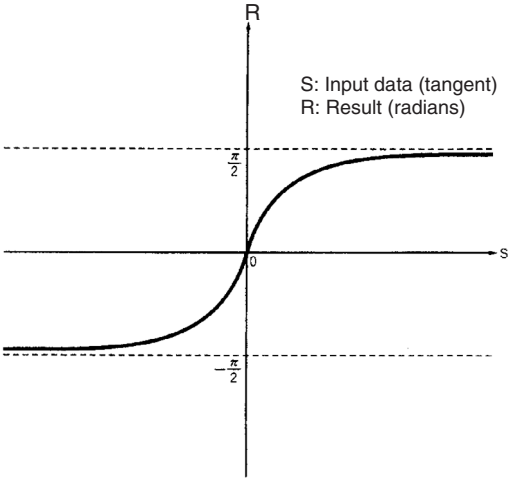
Description

ATAND(856) computes the angle (in radians) for a tangent value expressed as a double-precision (64-bit) floating-point number in words S to S+3 and places the result in D to D+3.
(The floating point source data must be in IEEE754 format.)



The result is output to words D to D+3 as an angle (in radians) within the range of $-\pi/2$ to $\pi/2$.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	Unchanged
Underflow Flag	UF	Unchanged
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

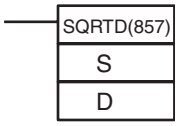
The source data in words S to S+3 must be in IEEE754 floating-point data format.

3-15-17 DOUBLE SQUARE ROOT: SQRTD(857)

Purpose

Calculates the square root of a double-precision (64-bit) floating-point number and places the result in the specified result words.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	SQRTD(857)
	Executed Once for Upward Differentiation	@SQRTD(857)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

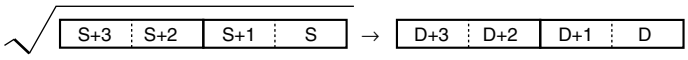
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W252	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T0252	
Counter Area	C0000 to C0252	
DM Area	D00000 to D32764	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

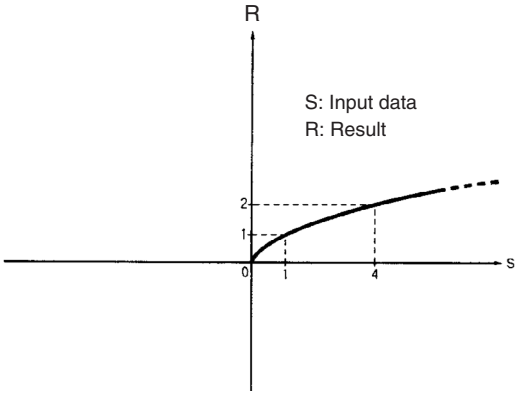
SQRTD(857) calculates the square root of the double-precision (64-bit) floating-point number in words S to S+3 and places the result in words D to D+3. (The floating point source data must be in IEEE754 format.)



The source data must be positive; if it is negative, an error will occur and the instruction will not be executed.

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is negative. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision (64-bit) floating-point value.
Underflow Flag	UF	Unchanged
Negative Flag	N	Unchanged

Precautions

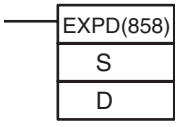
The source data in words S to S+3 must be in IEEE754 floating-point data format.

3-15-18 DOUBLE EXPONENT: EXPD(858)

Purpose

Calculates the natural (base e) exponential of a double-precision (64-bit) floating-point number and places the result in the specified result words.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	EXPD(858)
	Executed Once for Upward Differentiation	@EXPD(858)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

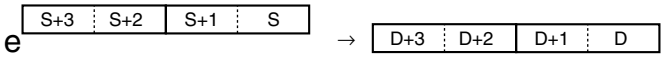
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W252	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T0252	
Counter Area	C0000 to C0252	
DM Area	D00000 to D32764	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

EXPD(858) calculates the natural (base e) exponential of the double-precision (64-bit) floating-point number in words S to S+3 and places the result in words D to D+3. In other words, EXP(467) calculates e^x (x = source) and places the result in words D to D+3.

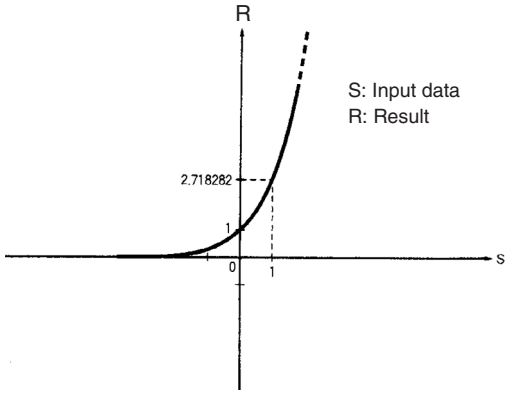


If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON and the result will be output as 0.

Note The constant e is 2.718282.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision (64-bit) floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a double-precision (64-bit) floating-point value.
Negative Flag	N	Unchanged

Precautions

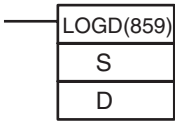
The source data in words S to S+3 must be in IEEE754 floating-point data format.

3-15-19 DOUBLE LOGARITHM: LOGD(859)

Purpose

Calculates the natural (base e) logarithm of a double-precision (64-bit) floating-point number and places the result in the specified destination words.

Ladder Symbol



S: First source word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	LOGD(859)
	Executed Once for Upward Differentiation	@LOGD(859)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S	D
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W252	
Auxiliary Bit Area	A000 to A956	A448 to A956
Timer Area	T0000 to T0252	
Counter Area	C0000 to C0252	
DM Area	D00000 to D32764	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Registers	---	

Area	S	D
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

LOGD(859) calculates the natural (base e) logarithm of the double-precision (64-bit) floating-point number in words S to S+3 and places the result in words D to D+3.

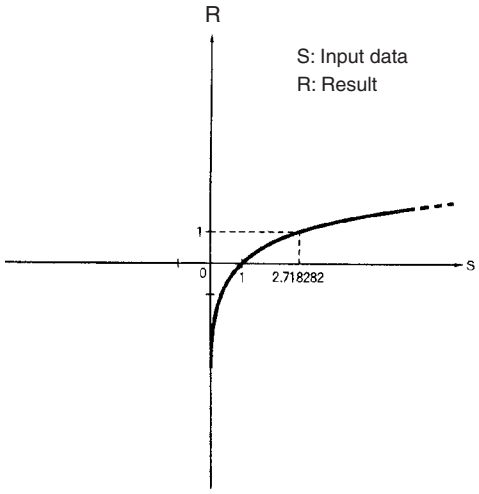
$\log_e$ [S+3 | S+2 | S+1 | S] → [D+3 | D+2 | D+1 | D]

The source data must be positive; if it is negative, an error will occur and the instruction will not be executed.

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON and the result will be output as ±∞.

Note The constant e is 2.718282.

The following diagram shows the relationship between the input data and result.



Flags

Name	Label	Operation
Error Flag	ER	ON if the source data is not recognized as floating-point data. ON if the source data is negative. ON if the source data is not a number (NaN). OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision (64-bit) floating-point value.

Name	Label	Operation
Underflow Flag	UF	Unchanged
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

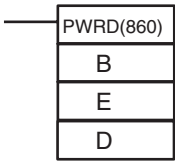
The source data in words S to S+3 must be in IEEE754 floating-point data format.

3-15-20 DOUBLE EXPONENTIAL POWER: PWRD(860)

Purpose

Raises a double-precision (64-bit) floating-point number to the power of another double-precision (64-bit) floating-point number.

Ladder Symbol



B: First base word
E: First exponent word
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	PWRD(860)
	Executed Once for Upward Differentiation	@PWRD(860)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

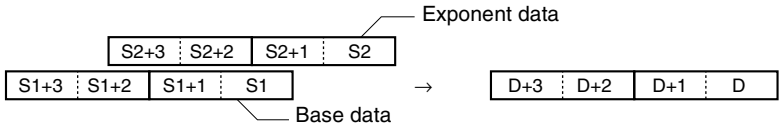
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	B	E	D
CIO Area	CIO 0000 to CIO 6140		
Work Area	W000 to W252		
Auxiliary Bit Area	A000 to A956		A448 to A956
Timer Area	T0000 to T0252		
Counter Area	C0000 to C0252		
DM Area	D00000 to D32764		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---		
Data Registers	---		
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

PWRD(860) raises the double-precision (64-bit) floating-point number in words B to B+3 to the power of the double-precision (64-bit) floating-point number in words E to E+3. In other words, PWR(840) calculates X^Y (X = content of B to B+3; Y = content of E to E+3).



For example, when the base words (B to B+3) contain 3.1 and the exponent words (E to E+3) contain 3, the result is 3.1^3 or 29.791.

If the absolute value of the result is greater than the maximum value that can be expressed as floating-point data, the Overflow Flag will turn ON.

If the absolute value of the result is less than the minimum value that can be expressed as floating-point data, the Underflow Flag will turn ON.

Flags

Name	Label	Operation
Error Flag	ER	ON if the base data (B to B+3) or exponent data (E to E+3) is not recognized as floating-point data. ON if the base data (B to B+3) or exponent data (E to E+3) is not a number (NaN). ON if the base data (B to B+3) is 0 and the exponent data (E to E+3) is less than 0. (Division by 0) ON if the base data (B to B+3) is negative and the exponent data (E to E+3) is non-integer. (Root of a negative number) OFF in all other cases.
Equals Flag	=	ON if both the exponent and mantissa of the result are 0. OFF in all other cases.
Overflow Flag	OF	ON if the absolute value of the result is too large to be expressed as a double-precision floating-point value.
Underflow Flag	UF	ON if the absolute value of the result is too small to be expressed as a double-precision floating-point value.
Negative Flag	N	ON if the result is negative. OFF in all other cases.

Precautions

The base data (B to B+3) and the exponent data (E to E+3) must be in IEEE754 floating-point data format.

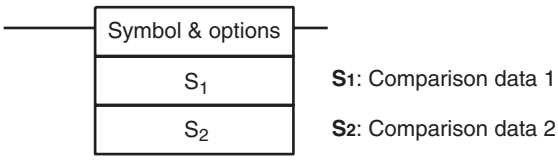
3-15-21 Double-precision Floating-point Input Instructions

Purpose

These input comparison instructions compare two double-precision floating point values (64-bit IEEE754 format) and create an ON execution condition when the comparison condition is true.

Note Refer to *3-6-1 Input Comparison Instructions (300 to 328)* for details on the signed and unsigned binary input comparison instructions and *3-14-21 Single-precision Floating-point Comparison Instructions* for details on single-precision floating-point input comparison instructions.

Ladder Symbol



Variations

Variations	Creates ON Each Cycle Comparison is True	Input comparison instruction

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Area	S_1	S_2
CIO Area	CIO 0000 to CIO 6140	
Work Area	W000 to W252	
Auxiliary Bit Area	A000 to A956	
Timer Area	T0000 to T0252	
Counter Area	C0000 to C0252	
DM Area	D00000 to D32764	
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	---	
Data Registers	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

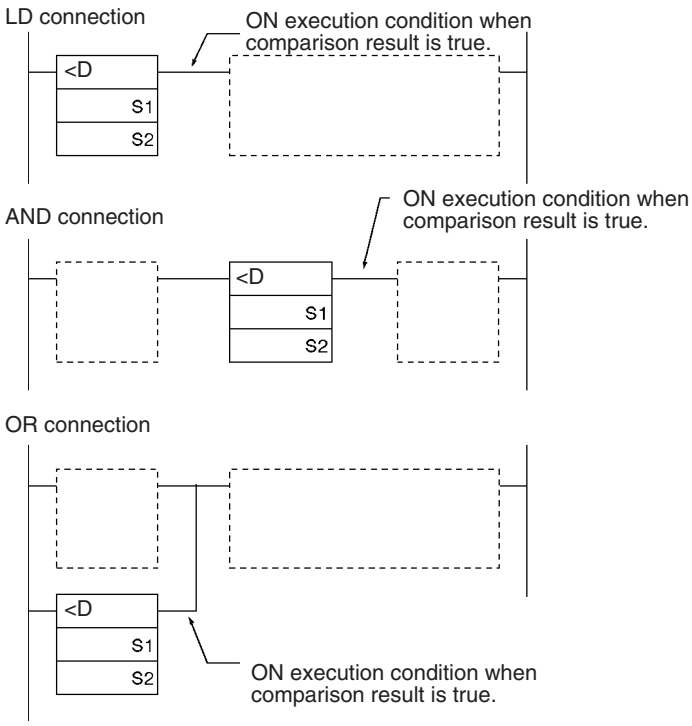
Description

The input comparison instruction compares the data specified in S_1 and S_2 as double-precision floating point values (64-bit IEEE754 data) and creates an ON execution condition when the comparison condition is true. When the data is stored in words, S_1 and S_2 specify the first of four words containing the 64-bit data. The 64-bit floating-point data cannot be input as constants.

Inputting the Instructions

The input comparison instructions are treated just like the LD, AND, and OR instructions to control the execution of subsequent instructions.

Input type	Operation
LD	The instruction can be connected directly to the left bus bar.
AND	The instruction cannot be connected directly to the left bus bar.
OR	The instruction can be connected directly to the left bus bar.



Options

With the three input types and six symbols, there are 18 different possible combinations.

Symbol		Option (data format)
=	(Equal)	D: Double-precision floating-point data
< >	(Not equal)	
<	(Less than)	
<=	(Less than or equal)	
>	(Greater than)	
>=	(Greater than or equal)	

Summary of Input Comparison Instructions

The following table shows the function codes, mnemonics, names, and functions of the 18 single-precision floating-point input comparison instructions. (C1=S_1+3, S_1+2, S_1+1, S_1 and C2=S_2+3, S_2+2, S_2+1, S_2.)

Code	Mnemonic	Name	Function
335	LD=D	LOAD DOUBLE FLOATING EQUAL	True if C1 = C2
	AND=D	AND DOUBLE FLOATING EQUAL	
	OR=D	OR DOUBLE FLOATING EQUAL	
336	LD<>D	LOAD DOUBLE FLOATING NOT EQUAL	True if C1 ≠ C2
	AND<>D	AND DOUBLE FLOATING NOT EQUAL	
	OR<>D	OR DOUBLE FLOATING NOT EQUAL	
337	LD<D	LOAD DOUBLE FLOATING LESS THAN	True if C1 < C2
	AND<D	AND DOUBLE FLOATING LESS THAN	
	OR<D	OR DOUBLE FLOATING LESS THAN	
338	LD<=D	LOAD DOUBLE FLOATING LESS THAN OR EQUAL	True if C1 ≤ C2
	AND<=D	AND DOUBLE FLOATING LESS THAN OR EQUAL	
	OR<=D	OR DOUBLE FLOATING LESS THAN OR EQUAL	

Code	Mnemonic	Name	Function
339	LD>D	LOAD DOUBLE FLOATING GREATER THAN	True if C1 > C2
	AND>D	AND DOUBLE FLOATING GREATER THAN	
	OR>D	OR DOUBLE FLOATING GREATER THAN	
340	LD>=D	LOAD DOUBLE FLOATING GREATER THAN OR EQUAL	True if C1 ≥ C2
	AND>=D	AND DOUBLE FLOATING GREATER THAN OR EQUAL	
	OR>=D	OR DOUBLE FLOATING GREATER THAN OR EQUAL	

Flags

In this table, C1 = content of S1 to S1+3 and C2 = content of S2 to S2+3.

Name	Label	Operation
Error Flag	ER	ON if C1 or C2 is not a valid floating-point number (NaN). ON if C1 or C2 is +∞. ON if C1 or C2 is −∞. OFF in all other cases.
Greater Than Flag	>	ON if C1 > C2. OFF in all other cases.
Greater Than or Equal Flag	> =	ON if C1 ≥ C2. OFF in all other cases.
Equal Flag	=	ON if C1 = C2. OFF in all other cases.
Not Equal Flag	=	ON if C1 ≠ C2. OFF in all other cases.
Less Than Flag	<	ON if C1 < C2. OFF in all other cases.
Less Than or Equal Flag	< =	ON if C1 ≤ C2. OFF in all other cases.
Negative Flag	N	Unchanged

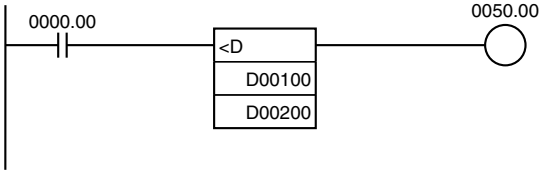
Precautions

Input comparison instructions cannot be used as right-hand instructions, i.e., another instruction must be used between them and the right bus bar.

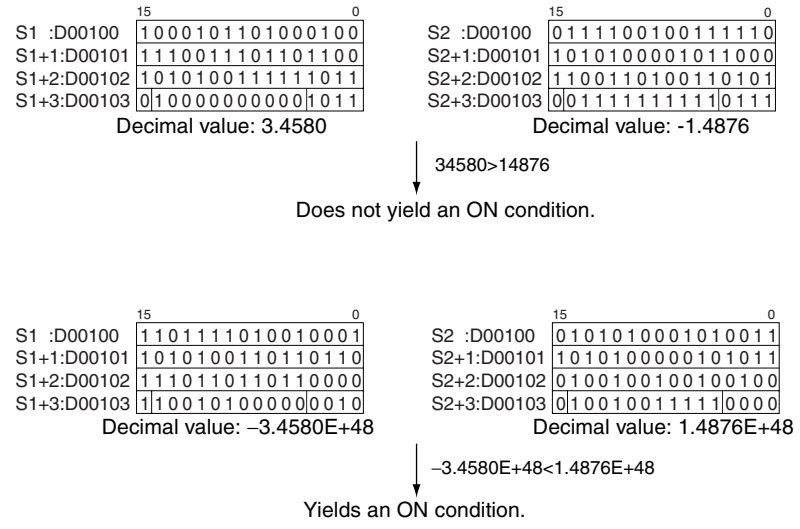
Example

AND DOUBLE FLOATING LESS THAN: AND<D(331)

When CIO 0000.00 is ON in the following example, the floating point data in words D00100 to D00103 is compared to the floating point data in words D00200 to D00203. If the content of D00100 to D00103 is less than that of D00200 to D00203, execution proceeds to the next line and CIO 0050.00 is turned ON. If the content of D00100 to D00103 is not less than that of D00200 to D00203, execution does not proceed to the next instruction line.



DOUBLE FLOATING LESS THAN Comparison (<D)



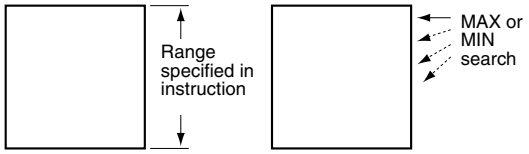
3-16 Table Data Processing Instructions

This section describes instructions used to handle table data.

Instruction	Mnemonic	Function code	Page
FIND MAXIMUM	MAX	182	467
FIND MINIMUM	MIN	183	471

Range Instructions

The range instructions included here act on a specified range of words to find the maximum value (MAX(182)) or minimum value (MIN(183)).

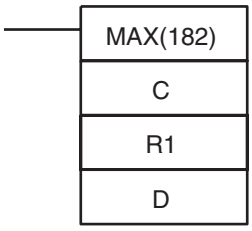


3-16-1 FIND MAXIMUM: MAX(182)

Purpose

Finds the maximum value in a range.

Ladder Symbol



C: First control word

R1: First word in range

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	MAX(182)
	Executed Once for Upward Differentiation	@MAX(182)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

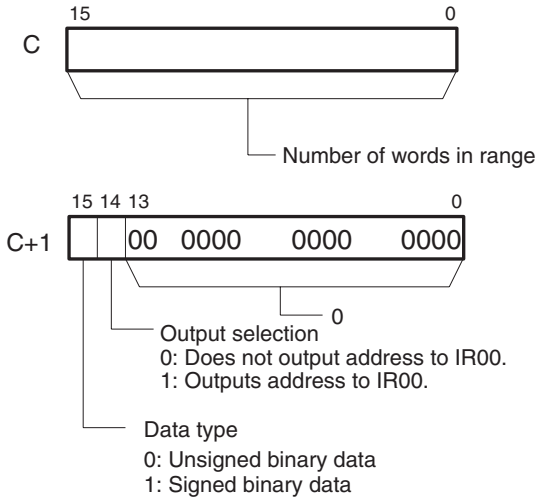
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C and C+1: Control words

C specifies the number of words in the range, bit 15 of C+1 indicates whether the data will be treated as signed binary or unsigned binary, and bit 14 of C+1 indicates whether or not to output the memory address of the word that contains the maximum value to IR00.

Note C and C+1 must be in the same data area.

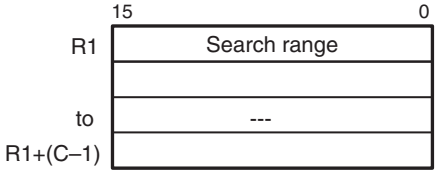


The following table shows the possible values of C+1.

C+1	Data type	Index Register output
0000	Unsigned binary	No
4000	Unsigned binary	Yes
8000	Signed binary	No
C000	Signed binary	Yes

R1: First word in range

R1 specifies the first word in the search range. The words from R1 to R1+(C–1) are searched for the maximum value. (C is the number of words specified in C.)



Note R1 and R1+(C–1) must be in the same data area.

Operand Specifications

Area	C	R1	D
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143	
Work Area	W000 to W254	W000 to W255	
Auxiliary Bit Area	A000 to A958	A000 to A959	A448 to A959
Timer Area	T0000 to T0254	T0000 to T0255	
Counter Area	C0000 to C0254	C0000 to C0255	
DM Area	D00000 to D32766	D00000 to D32767	
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	Specified values only	---	
Data Resisters	---		DR0 to DR15

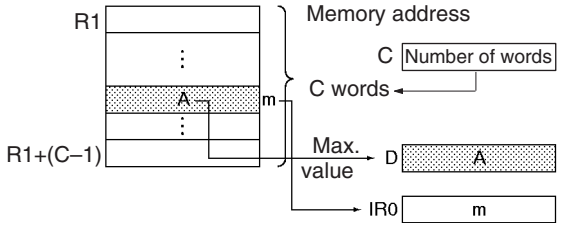
Area	C	R1	D
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

MAX(182) searches the range of memory from R1 to R1+C–1 for the maximum value in the range and outputs that maximum value to D.

When bit 14 of C+1 has been set to 1, MAX(182) writes the memory address of the word containing the maximum value to IR00. (If two or more words within the range contain the maximum value, the address of the first word containing the maximum value is written to IR00.)

When bit 15 of C+1 has been set to 1, MAX(182) treats the data within the range as signed binary data.



Flags

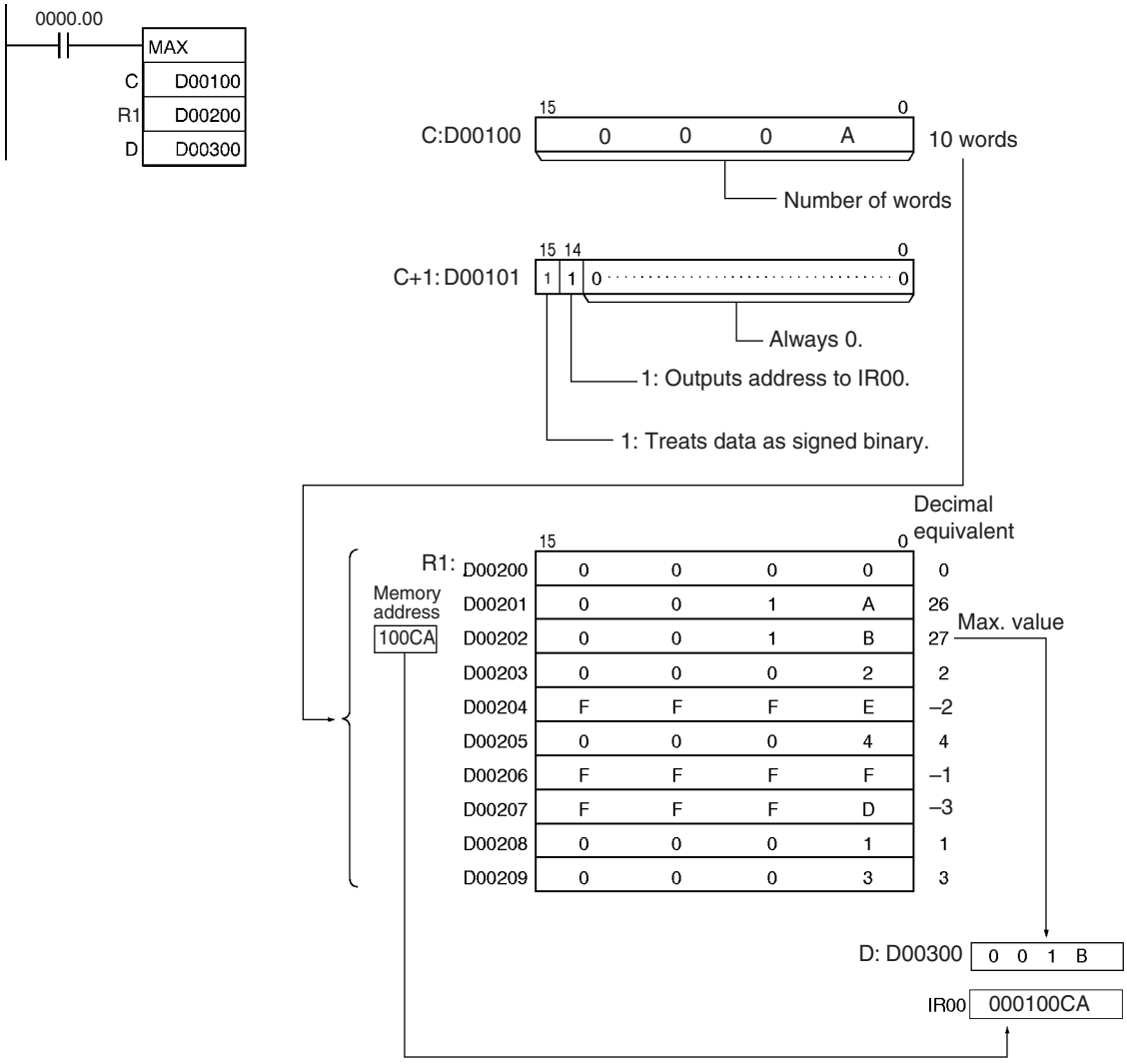
Name	Label	Operation
Error Flag	ER	ON if the content of C is not within the specified range of 0001 through FFFF. OFF in all other cases.
Equals Flag	=	ON if the maximum value is 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 is ON in the word containing the maximum value. OFF in all other cases.

Precautions

When bit 15 of C+1 has been set to 1, the data within the range is treated as signed binary data and hexadecimal values 8000 to FFFF are considered negative. Thus, the results of the search will differ depending on the data-type setting.

Examples

When CIO 0000.00 turns ON in the following example, MAX(182) searches the 10-word range beginning at D00200 for the maximum value. The maximum value is written to D00300 and the memory address of the word containing the maximum value is written to IR00.

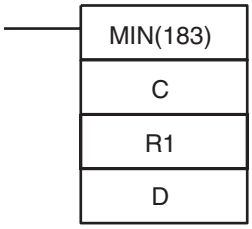

0000.00
MAX
C D00100
R1 D00200
D D00300
C:D00100
15 0
0 0 0 A
10 words
Number of words
C+1: D00101
15 14 0
1 1 0 0
Always 0.
1: Outputs address to IR00.
1: Treats data as signed binary.
Decimal equivalent
R1:
Memory address
100CA
D00200 0 0 0 0 0
D00201 0 0 1 A 26
D00202 0 0 1 B 27
D00203 0 0 0 2 2
D00204 F F F E –2
D00205 0 0 0 4 4
D00206 F F F F –1
D00207 F F F D –3
D00208 0 0 0 1 1
D00209 0 0 0 3 3
Max. value
D: D00300 0 0 1 B
IR00 000100CA

3-16-2 FIND MINIMUM: MIN(183)

Purpose

Finds the minimum value in a range.

Ladder Symbol



C: First control word

R1: First word in range

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	MIN(183)
	Executed Once for Upward Differentiation	@MIN(183)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

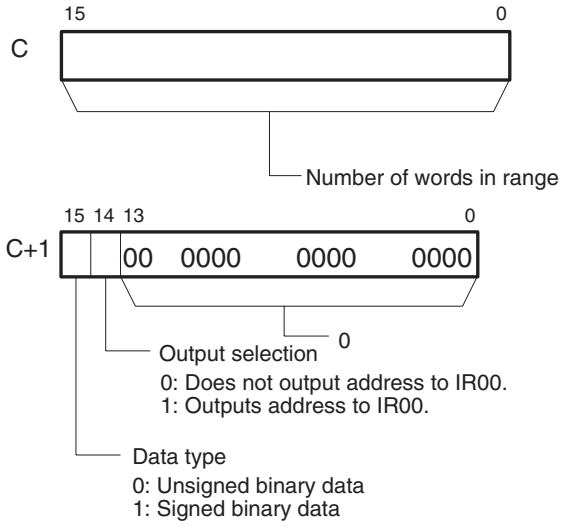
Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C and C+1: Control words

C specifies the number of words in the range, bit 15 of C+1 indicates whether the data will be treated as signed binary or unsigned binary, and bit 14 of C+1 indicates whether or not to output the memory address of the word that contains the minimum value to IR00.

Note C and C+1 must be in the same data area.

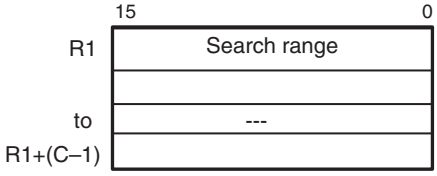


The following table shows the possible values of C+1.

C+1	Data type	Index Register output
0000	Unsigned binary	No
4000	Unsigned binary	Yes
8000	Signed binary	No
C000	Signed binary	Yes

R1: First word in range

R1 specifies the first word in the search range. The words from R1 to R1+(C–1) are searched for the minimum value. (C is the number of words specified in C.)



Note R1 and R1+C–1 must be in the same data area.

Operand Specifications

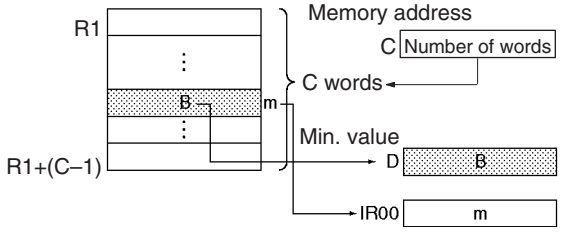
Area	C	R1	D
CIO Area	CIO 0000 to CIO 6142	CIO 0000 to CIO 6143	
Work Area	W000 to W254	W000 to W255	
Auxiliary Bit Area	A000 to A958	A000 to A959	A448 to A959
Timer Area	T0000 to T0254	T0000 to T0255	
Counter Area	C0000 to C0254	C0000 to C0255	
DM Area	D00000 to D32766	D00000 to D32767	
Indirect DM addresses in binary	@ D0000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	Specified values only	---	
Data Resisters	---		DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

MIN(183) searches the range of memory from R1 to R1+C–1 for the minimum value in the range and outputs that minimum value to D.

When bit 14 of C+1 has been set to 1, MIN(183) writes the memory address of the word containing the minimum value to IR00. (If two or more words within the range contain the minimum value, the address of the first word containing the minimum value is written to IR00.)

When bit 15 of C+1 has been set to 1, MIN(183) treats the data within the range as signed binary data.



Flags

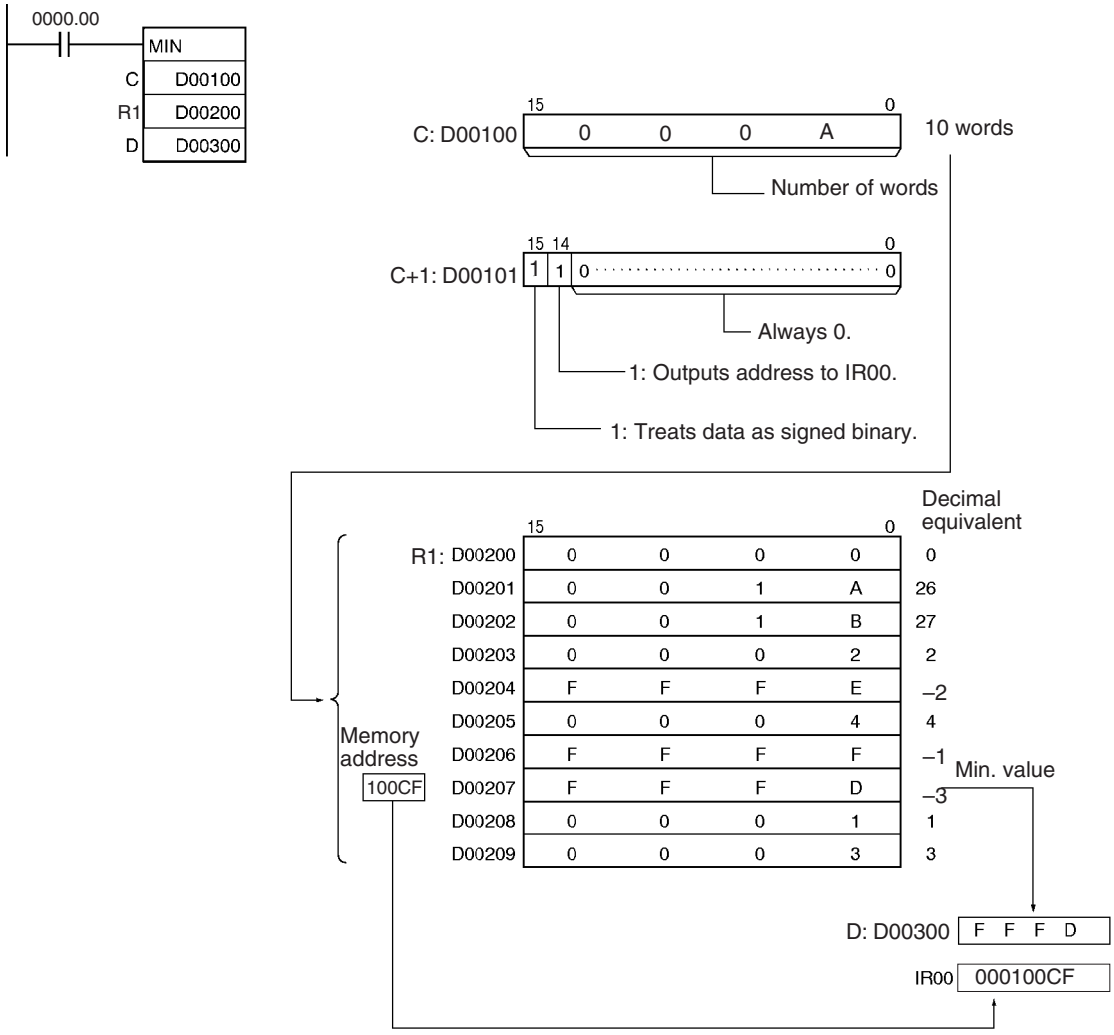
Name	Label	Operation
Error Flag	ER	ON if the content of C is not within the specified range of 0001 through FFFF. OFF in all other cases.
Equals Flag	=	ON if the minimum value is 0000. OFF in all other cases.
Negative Flag	N	ON if bit 15 is ON in the word containing the minimum value. OFF in all other cases.

Precautions

When bit 15 of C+1 has been set to 1, the data within the range is treated as signed binary data and hexadecimal values 8000 to FFFF are considered negative. Thus, the results of the search will differ depending on the data-type setting.

Examples

When CIO 0000.00 turns ON in the following example, MIN(183) searches the 10-word range beginning at D00200 for the minimum value. The minimum value is written to D00300 and the memory address of the word containing the minimum value is written to IR00.


0000.00
MIN
C D00100
R1 D00200
D D00300
C: D00100
15 0
0 0 0 A
10 words
Number of words
C+1: D00101
15 14 0
1 1 0 0
Always 0.
1: Outputs address to IR00.
1: Treats data as signed binary.
Decimal equivalent
R1: D00200 0 0 0 0 0
D00201 0 0 1 A 26
D00202 0 0 1 B 27
D00203 0 0 0 2 2
D00204 F F F E –2
D00205 0 0 0 4 4
D00206 F F F F –1
D00207 F F F D –3
D00208 0 0 0 1 1
D00209 0 0 0 3 3
Memory address
100CF
Min. value
D: D00300 F F F D
IR00 000100CF

3-17 Data Control Instructions

This section describes instructions used to scale and average data.

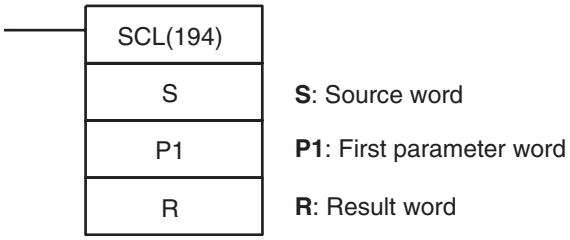
Instruction	Mnemonic	Function code	Page
SCALING	SCL	194	475
SCALING 2	SCL2	486	479
SCALING 3	SCL3	487	483
AVERAGE	AVG	195	486

3-17-1 SCALING: SCL(194)

Purpose

Converts unsigned binary data into unsigned BCD data according to the specified linear function.

Ladder Symbol



Variations

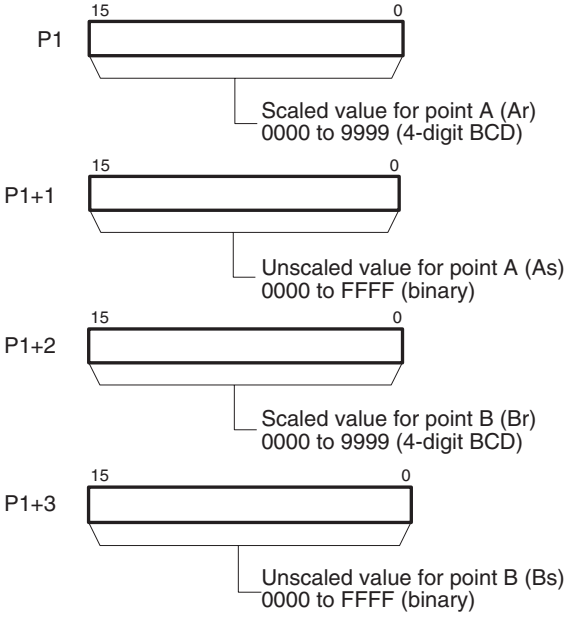
Variations		
	Executed Each Cycle for ON Condition	SCL(194)
	Executed Once for Upward Differentiation	@SCL(194)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

The contents of the four words starting with the first parameter word (P1) are shown in the following diagram.



Note P1 to P1+3 must be in the same area.

Operand Specifications

Area	S	P1	R
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6140	CIO 0000 to CIO 6143
Work Area	W000 to W255	W000 to W252	W000 to W255
Auxiliary Bit Area	A000 to A959	A000 to A956	A448 to A959
Timer Area	T0000 to T0255	T0000 to T0252	T0000 to T0255
Counter Area	C0000 to C0255	C0000 to C0252	C0000 to C0255
DM Area	D00000 to D32767	D00000 to D32764	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---		
Data Resisters	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

SCL(194) is used to convert the unsigned binary data contained in the source word S into unsigned BCD data and place the result in the result word R according to the linear function defined by points (As, Ar) and (Bs, Br). The address of the first word containing the coordinates of points (As, Ar) and (Bs, Br) is specified for the first parameter word P1. These points are defined by 2 values (As and Bs) before scaling and 2 values (Ar and Br) after scaling.

The following equations are used for the conversion.

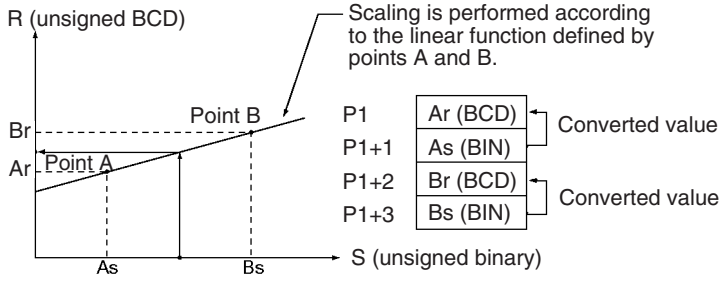
$$R = Br - \frac{(Br - Ar)}{\text{BCD conversion of } (Bs - As)} \times \text{BCD conversion of } (Bs - S)$$

The slope of the line is as follows:

$$R = Br - \frac{(Br - Ar)}{\text{BCD conversion of } (Bs - As)}$$

Points A and B can define a line with either a positive or negative slope. Using a negative slope enables reverse scaling.

The result will be rounded to the nearest integer. If the result is less than 0000, 0000 will be output as the result. If the result is greater than 9999, 9999 will be output.



SCL(194) can be used to scale the results of analog signal conversion values from Motion Control Modules with Analog I/O (FQM1-MMA22) according to

user-defined scale parameters. For example, if a 1 to 5-V input to a Motion Control Module with Analog I/O is input to memory as 0000 to 0FA0 hexadecimal, the value in memory can be scaled to 50 to 200 BCD using SCL(194).

SCL(194) converts unsigned binary to unsigned BCD. To convert a negative value, it will be necessary to first add the maximum negative value in the program before using SCL(194) (see example).

SCL(194) cannot output a negative value to the result word, R. If the result is a negative value, 0000 will be output to R.

Flags

Name	Label	Operation
Error Flag	ER	ON if the contents of P1 (Ar) or P1+2 (Br) is not BCD. ON if the contents of P1+1 (As) and P1+3 (Bs) are equal. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.

Precautions

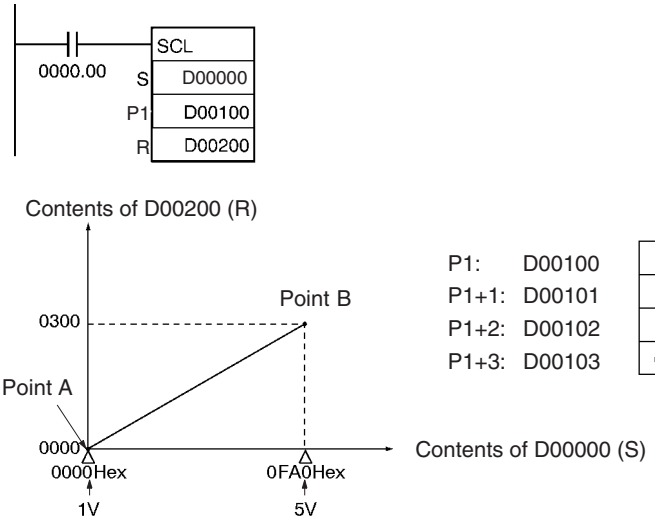
An error will occur and the Error Flag will turn ON if the values for Ar (P1) and Br (P1+2) are not in BCD, or if the values for As (P1+1) and Bs (P1+3) are equal.

The Equals Flag will turn ON when the contents of the result word R is 0000.

Examples

In the following example, it is assume that an analog signal from 1 to 5 V is converted and input to D00000 as 0000 to 0FA0 hexadecimal. SCL(194) is used to convert (scale) the value in D00000 to a value between 0000 and 0300 BCD.

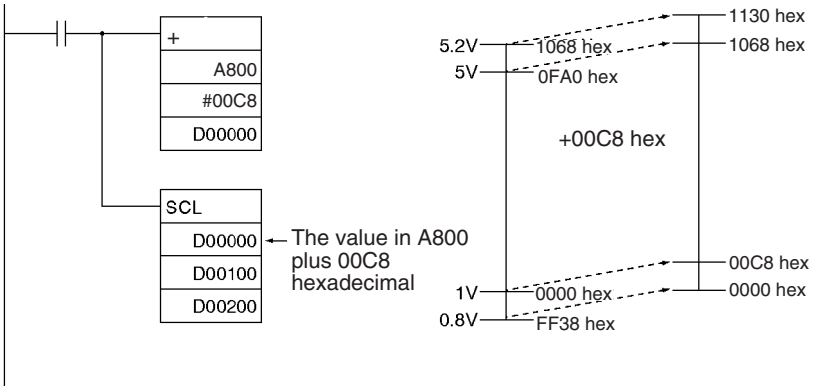
When CIO 0000.00 is ON, the contents of D00000 is scaled using the linear function defined by point A (0000, 0000) and point B (0FA0, 0300). The coordinates of these points are contained in D00100 to D00103, and the result is output to D00200.

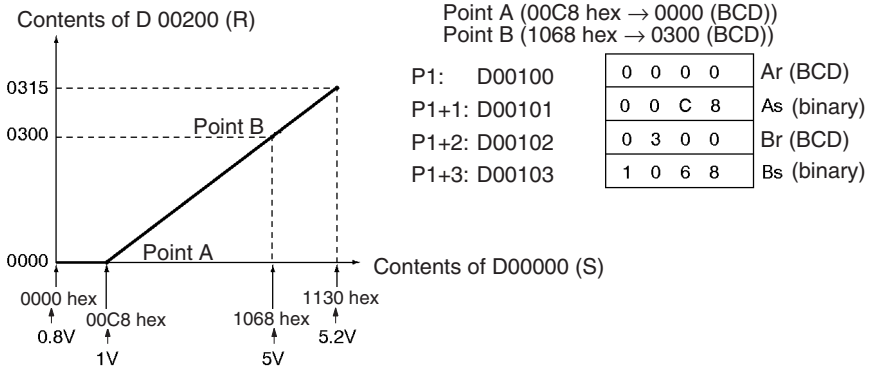


Negative Values

A Motion Control Module actually inputs values from FF38 to 1068 hexadecimal for 0.8 to 5.2 V. SCL(194), however, can handle only unsigned binary values between 0000 and FFFF hexadecimal, making it impossible to use SCL(194) directly to handle signed binary values below 1 V (0000 hexadecimal), i.e., FF38 to FFFF hexadecimal. In an actual application, it is thus necessary to add 00C8 hexadecimal to all values so that FF38 hexadecimal is

represented as 0000 hexadecimal before using SCL(194), as shown in the following example.

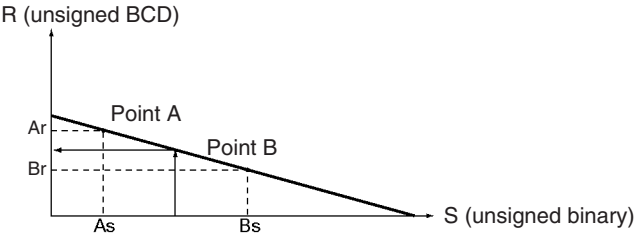




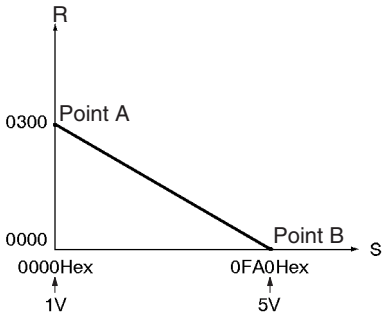
In this example, values from 0000 to 00C8 hexadecimal will be converted to negative values. SCL(194), however, can output only unsigned BCD values from 0000 to 9999, so 0000 BCD will be output whenever the content of D00000 is between 0000 and 00C8 hexadecimal.

Reverse Scaling

Reverse scaling can also be used by setting As < Bs and Ar > Br. The following relationship will result.



Reverse scaling can be used, for example, to convert (reverse scale) 1 to 5 V (0000 to 0FA0 hexadecimal) to 0300 to 0000, respectively, as shown in the following diagram.

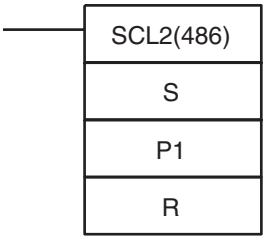


3-17-2 SCALING 2: SCL2(486)

Purpose

Converts signed binary data into signed BCD data according to the specified linear function. An offset can be input in defining the linear function.

Ladder Symbol



S: Source word

P1: First parameter word

R: Result word

Variations

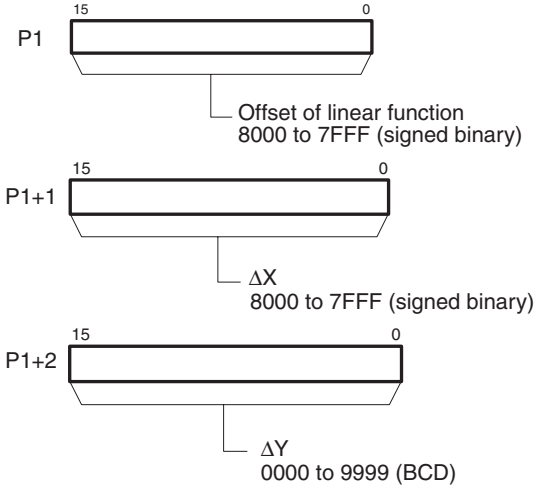
Variations		
	Executed Each Cycle for ON Condition	SCL2(486)
	Executed Once for Upward Differentiation	@SCL2(486)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

The contents of the three words starting with the first parameter word (P1) are shown in the following diagram.



Note P1 to P1+2 must be in the same area.

Operand Specifications

Area	S	P1	R
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6141	CIO 0000 to CIO 6143
Work Area	W000 to W255	W000 to W253	W000 to W255
Auxiliary Bit Area	A000 to A959	A000 to A957	A448 to A959
Timer Area	T0000 to T0255	T0000 to T0253	T0000 to T0255
Counter Area	C0000 to C0255	C0000 to C0253	C0000 to C0255
DM Area	D00000 to D32767	D00000 to D32765	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---		
Data Resisters	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

SCL2(486) is used to convert the signed binary data contained in the source word S into signed BCD data (the BCD data contains the absolute value and the Carry Flag shows the sign) and place the result in the result word R according to the linear function defined by the slope (ΔX, ΔY) and an offset. The address of the first word containing ΔX, ΔY, and the offset is specified for the first parameter word P1. The sign of the result is indicated by the status of the Carry Flag (ON: negative, OFF: positive).

The following equations are used for the conversion.

$$R = \frac{\Delta Y}{\text{BCD conversion of } \Delta X} \times ((\text{BCD conversion of S}) - (\text{BCD conversion of offset})$$

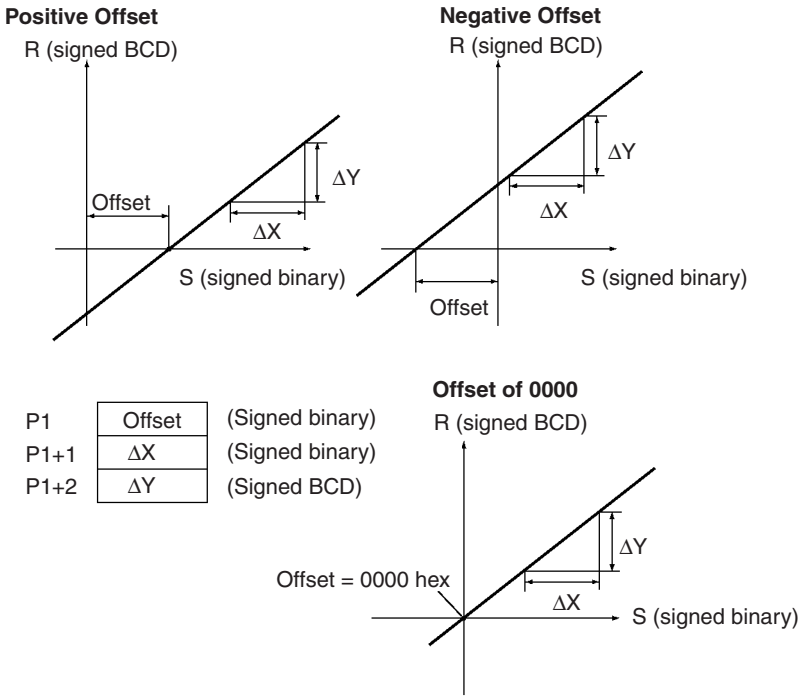
The slope of the line is ΔY/ΔX.

The offset and slope can be a positive value, 0, or a negative value. Using a negative slope enables reverse scaling.

The result will be rounded to the nearest integer.

The result in R will be the absolute BCD conversion value and the sign will be indicated by the Carry Flag. The result can thus be between –9999 and 9999.

If the result is less than –9999, –9999 will be output as the result. If the result is greater than 9999, 9999 will be output.



SCL2(486) can be used to scale the results of analog signal conversion values from Motion Control Modules with Analog I/O (FQM1-MMA22) according to user-defined scale parameters. For example, if a 1 to 5-V input to a Motion Control Module is input to memory as 0000 to 0FA0 hexadecimal, the value in memory can be scaled to –100 to 200 BCD using SCL2(486).

SCL2(486) converts signed binary to signed BCD. Negative values can thus be handled directly for S. The result of scaling in R and the Carry Flag can also be used to output negative values for the scaling result.

Flags

Name	Label	Operation
Error Flag	ER	ON if the contents of P1+1 (ΔX) is 0000. ON if the contents of P1+2 (ΔY) is not BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0. OFF in all other cases.
Carry Flag	CY	ON if the result is negative. OFF if the result is zero or positive.

Precautions

An error will occur and the Error Flag will turn ON if the value for ΔX (P1+1) is 0000 or if the value for ΔY (P1+2) is not BCD.

The Equals Flag will turn ON when the contents of the result word R is 0000.

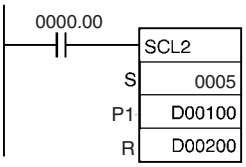
The Carry Flag will turn ON if the value placed in the result word is negative.

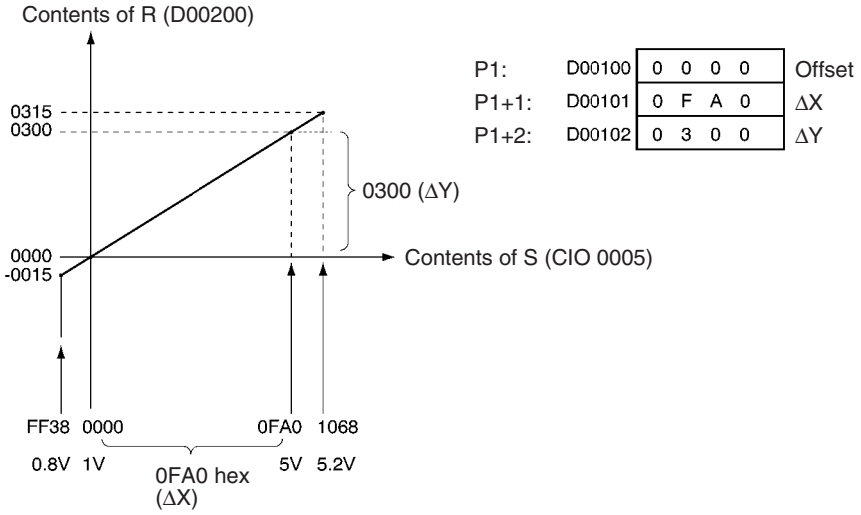
Examples

Scaling 1 to 5-V Analog Input to 0 to 300

In the following example, it is assumed that an analog signal from 1 to 5 V is converted and input to CIO 0005 as 0000 to 0FA0 hexadecimal. SCL2(486) is used to convert (scale) the value in CIO 0005 to a value between 0000 and 0300 BCD.

When CIO 0000.00 is ON, the contents of CIO 0005 is scaled using the linear function defined by ΔX (0FA0), ΔY (0300), and the offset (0). These values are contained in D00100 to D00102, and the result is output to D00200.

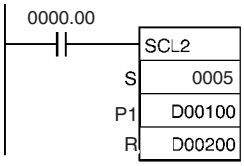


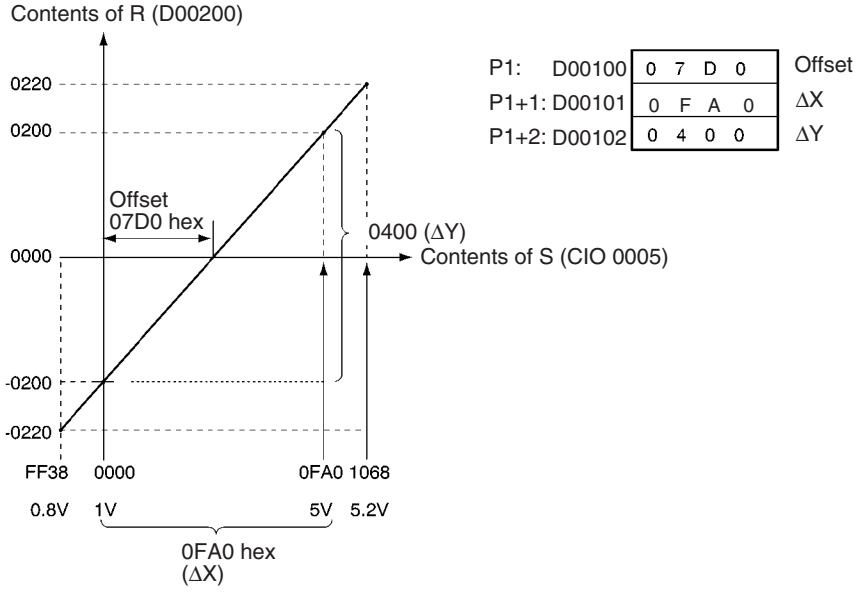


Scaling 1 to 5-V Analog Input to –200 to 200

In the following example, it is assume that an analog signal from 1 to 5 V is converted and input to CIO 0005 as 0000 to 0FA0 hexadecimal. SCL2(486) is used to convert (scale) the value in CIO 0005 to a value between –0200 and 0200 BCD.

When CIO 0000.00 is ON, the contents of CIO 0005 is scaled using the linear function defined by ΔX (0FA0), ΔY (0400), and the offset (07D0). These values are contained in D00100 to D00102, and the result is output to D00200.



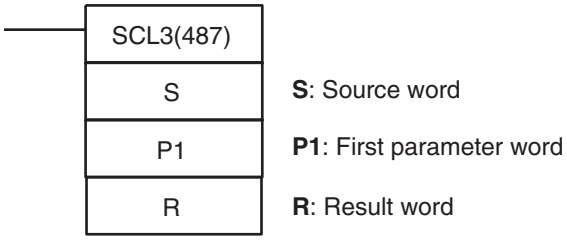


3-17-3 SCALING 3: SCL3(487)

Purpose

Converts signed BCD data into signed binary data according to the specified linear function. An offset can be input in defining the linear function.

Ladder Symbol



Variations

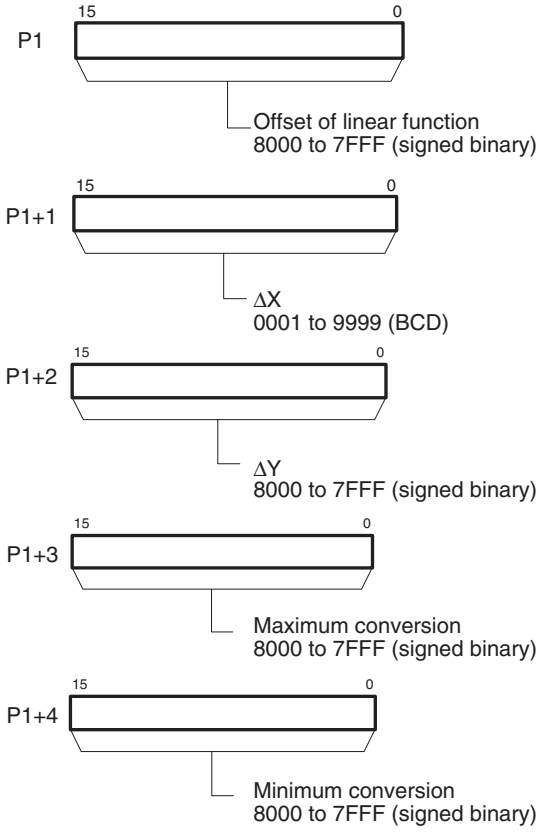
Variations		
	Executed Each Cycle for ON Condition	SCL3(487)
	Executed Once for Upward Differentiation	@SCL3(487)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

The contents of the five words starting with the first parameter word (P1) are shown in the following diagram.



Note P1 to P1+4 must be in the same area.

Operand Specifications

Area	S	P1	R
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6139	CIO 0000 to CIO 6143
Work Area	W000 to W255	W000 to W251	W000 to W255
Auxiliary Bit Area	A000 to A959	A000 to A955	A448 to A959
Timer Area	T0000 to T0255	T0000 to T0251	T0000 to T0255
Counter Area	C0000 to C0255	C0000 to C0251	C0000 to C0255
DM Area	D00000 to D32767	D00000 to D32763	D00000 to D32767
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---		
Data Resisters	DR0 to DR15	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

SCL3(487) is used to convert the signed BCD data (the BCD data contains the absolute value and the Carry Flag shows the sign) contained in the source word S into signed binary data and place the result in the result word R according to the linear function defined by the slope (ΔX, ΔY) and an offset. The maximum and minimum conversion values are also specified. The address of the first word containing ΔX, ΔY, the offset, the maximum conversion, and the minimum conversion is specified for the first parameter word P1.

The sign of the result is indicated by the status of the Carry Flag (ON: negative, OFF: positive). Use STC(040) and CLC(041) to turn the Carry Flag ON and OFF.

The following equations are used for the conversion.

$$R = \frac{\Delta Y}{\text{Binary conversion of } \Delta X} \times ((\text{Binary conversion of S}) + (\text{Offset}))$$

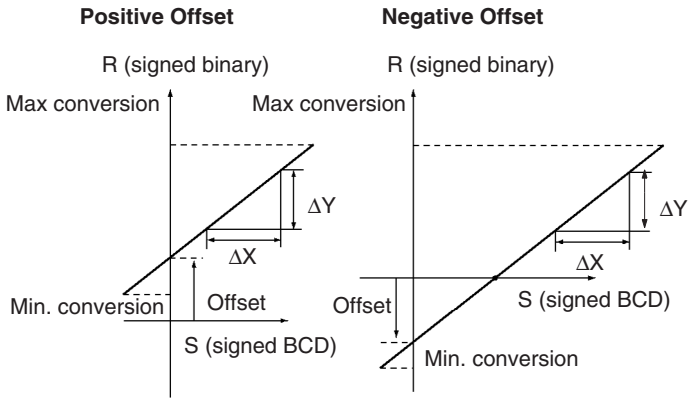
The slope of the line is $\Delta Y/\Delta X$.

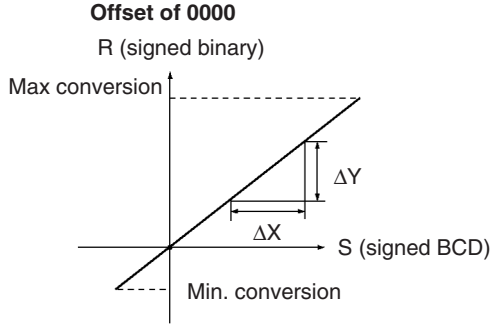
The offset and slope can be a positive value, 0, or a negative value. Using a negative slope enables reverse scaling.

The result will be rounded to the nearest integer.

The source value in S is treated as an absolute BCD value and the sign is indicated by the Carry Flag. The source value can thus be between –9999 and 9999.

If the result is less than the minimum conversion value, the minimum conversion value will be output as the result. If the result is greater than the maximum conversion value, the maximum conversion value will be output.





SCL3(487) is used to convert data using a user-defined scale to signed binary for Analog Output Units. For example, SCL3(487) can convert 0 to 200 °C to 0000 to 0FA0 (hex) and output an analog output signal 1 to 5 V from the Analog Output Unit.

Flags

Name	Label	Operation
Error Flag	ER	ON if the contents of S is not BCD. ON if the contents of P1+1 (ΔX) is not between 0001 and 9999 BCD. OFF in all other cases.
Equals Flag	=	ON if the result is 0000. OFF in all other cases.
Negative Flag	N	ON when the MSB of the R (the result) is 1. OFF in all other cases.

Precautions

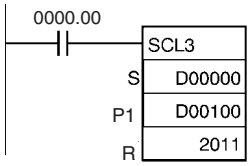
An error will occur and the Error Flag will turn ON if the contents of S is not BCD or if the value for ΔX (P1+1) is not between 0001 and 9999 BCD.

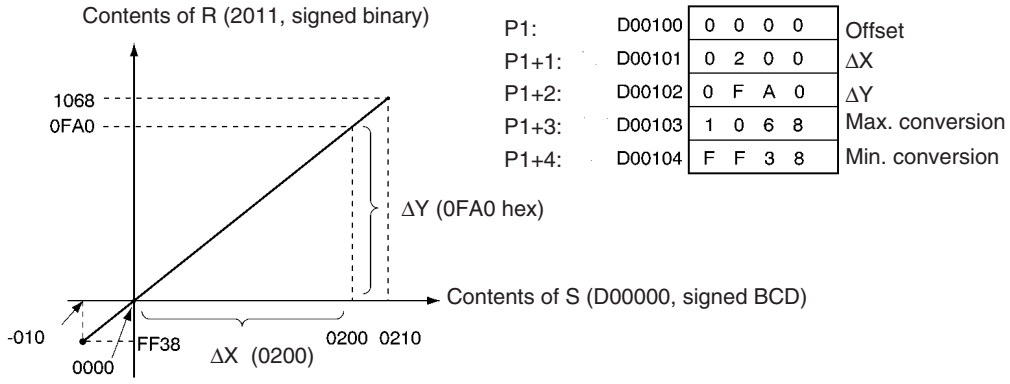
The Equals Flag will turn ON when the contents of the result word R is 0000.

The Negative Flag will turn ON if the MSB of the result in R is 1, i.e., if the result is negative.

Examples

When a value from 0 to 200 is scaled to an analog signal (1 to 5 V, for example), a signed BCD value of 0000 to 0200 is converted (scaled) to signed binary value of 0000 to 0FA0 for a Motion Control Module. When CIO 0000.00 turns ON in the following example, the contents of D00000 is scaled using the linear function defined by ΔX (0200), ΔY (0FA0), and the offset (0). These values are contained in D00100 to D00102. The sign of the BCD value in D00000 is indicated by the Carry Flag. The result is output to CIO 2011.



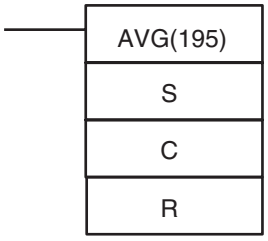


3-17-4 AVERAGE: AVG(195)

Purpose

Calculates the average value of an input word for the specified number of cycles.

Ladder Symbol



S: Source word

C: Control word (Number of cycles N and sign specification)

R: Result word (average)

R+1: First work area word

Variations

Variations		
	Executed Each Cycle for ON Condition	AVG(195)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.

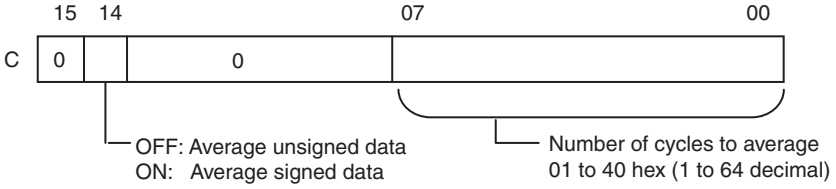
Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	OK	OK

Operands

C: **Number of Cycles and Sign Specification**

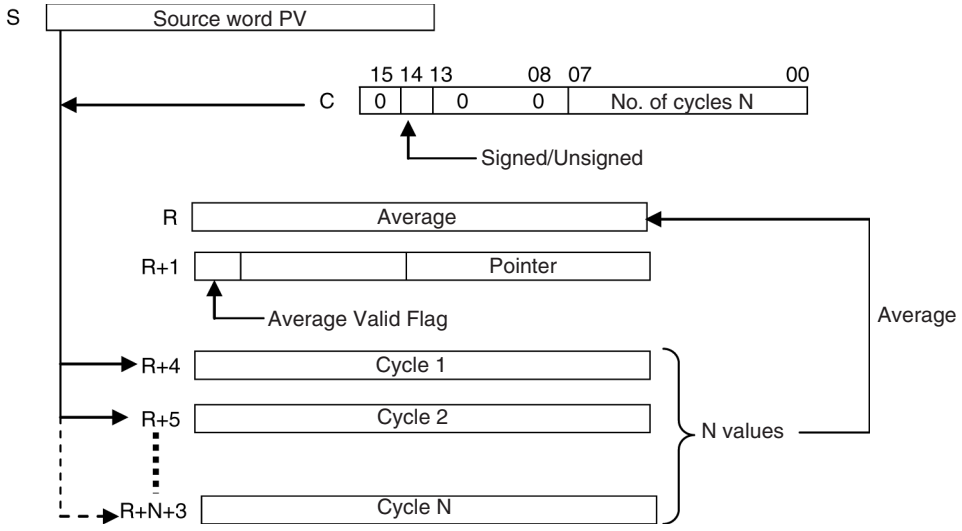
The number of cycles N must be between 01 and 40 hexadecimal (1 to 64 cycles). Bit 14 specifies if signed data or unsigned data is used.



R: Result Word (Average)
R+1: First Work Area Word (Read-only)

R will contain the average value after the specified number of cycles. R+1 provides information on the averaging process and R+2 to R+N+3 contain the previous values of S as shown in the following diagram.

R: Average
R+1: Processing data (read-only)
R+2: Processing data (read-only)
R+3: Processing data (read-only)
R+4: Previous value #1
R+5: Previous value #2
R+N+3: Previous value #N



Note R to R+N+3 must be in the same area.

Operand Specifications

Area	S	C	R
CIO Area	CIO 0000 to CIO 6143		CIO 000 to CIO 6139
Work Area	W000 to W255		W000 to W251
Auxiliary Bit Area	A000 to A959		A448 to A955
Timer Area	T0000 to T0255		T0000 to T0251
Counter Area	C0000 to C0255		C0000 to C0251
DM Area	D00000 to D32767		D00000 to D32763
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	#0000 to #FFFF (binary)	#0001 to #0040 (binary)	---
Data Resisters	DR0 to DR15		---

Area	S	C	R
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

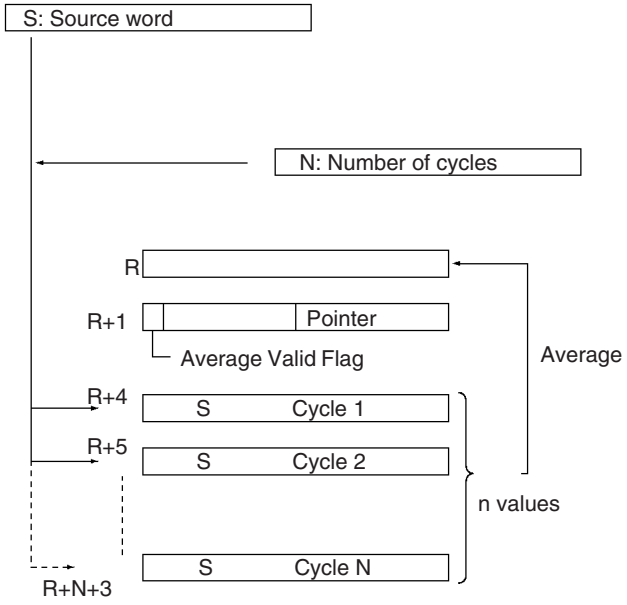
Description

For the first N–1 cycles when the execution condition is ON, AVG(195) writes the values of S (signed or unsigned binary data) in order to words starting with R+4. The Previous Value Pointer (bits 00 to 07 of R+1) is incremented each time a value is written. Until the Nth value is written, the contents of S will be output unchanged to R and the Average Value Flag (bit 15 of R+1) will remain OFF.

When the Nth value is written to R+N+3, the average of all the values that have been stored will be computed, the average will be output to R as a signed or unsigned binary value, and the Average Value Flag (bit 15 of R+1) will be turned ON. For all further cycles, the value in R will be updated for the most current N values of S. The maximum value of N is 64.

The Previous Value Pointer will be reset to 0 after N–1 values have been written.

The average value output to R will be rounded to the nearest integer.



Flags

Name	Label	Operation
Error Flag	ER	ON if the value of N is 0. OFF in all other cases.

Precautions

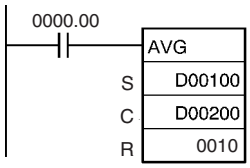
The contents of the First Work Area Word (R+1) is cleared to 0000 each time the execution condition changes from OFF to ON.

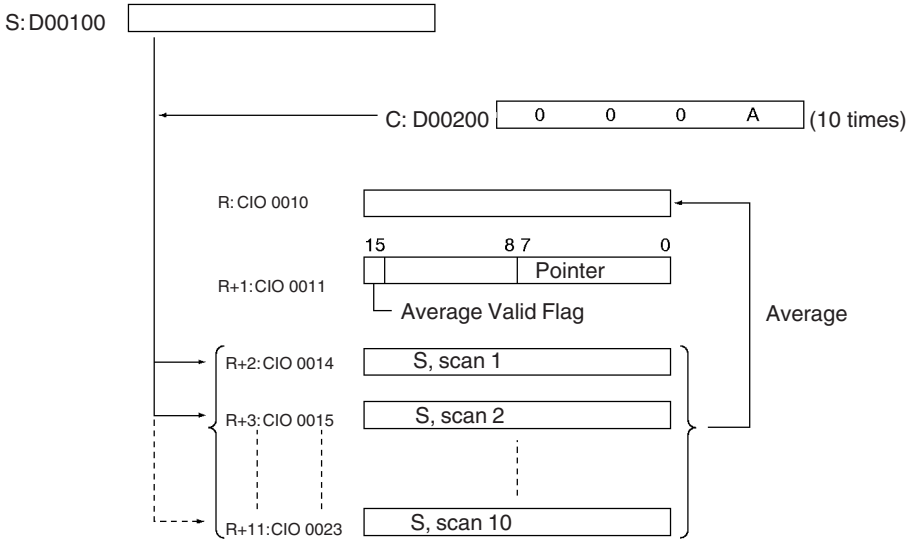
The contents of the First Work Area Word (R+1) will not be cleared to 0000 the first time the program is executed at the start of operation. If AVG(195) is to be executed in the first program scan, clear the First Work Area Word from the program.

If N (Number of Cycles) is 00, an error will occur and the Error Flag will turn ON.

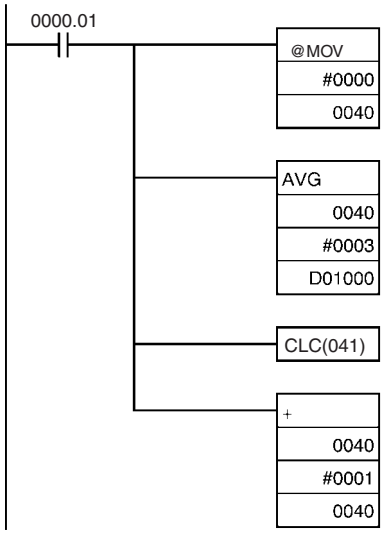
Examples

When CIO 0000.00 is ON in the following example, the contents of D00100 will be stored one time each scan for the number of scans specified in D00200. The contents will be stored in order in the ten words from CIO 0014 to CIO 0023. The average of the contents of these ten words will be placed in CIO 0010 and then bit 15 of CIO 0011 will be turned ON.





In the following example, the content of CIO 0040 is set to #0000 and then incremented by 1 each cycle. For the first two cycles, AVG(195) stores the content of CIO 0040 to D01004 and D01005. The contents of D01001 will also change (which can be used to confirm that the results of AVG(195) has changed). On the third and later cycles, AVG(195) calculates the average value of the contents of D01004 to D01006 and writes that average value to D01000.



	1st cycle	2nd cycle	3rd cycle	4th cycle
CIO 0040	0000	0001	0002	0003

D01000	0000	0001	0001	0002	Average
D01001	0001	0002	8000	8001	Pointer
D01002	---	---	---	---	
D01003	---	---	---	---	
D01004	0000	0000	0000	0003	3 previous values of CIO 0040
D01005	---	0001	0001	0001	
D01006	---	---	0002	0002	

3-18 Subroutines

This section describes instructions used to control subroutines.

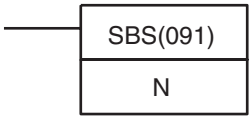
Instruction	Mnemonic	Function code	Page
SUBROUTINE CALL	SBS	091	491
MACRO	MCRO	099	496
SUBROUTINE ENTRY	SBN	092	500
SUBROUTINE RETURN	RET	093	503
JUMP TO SUBROUTINE	JSB	982	503

3-18-1 SUBROUTINE CALL: SBS(091)

Purpose

Calls the subroutine with the specified subroutine number and executes that program.

Ladder Symbol



N: Subroutine number

Variations

Variations		
	Executed Each Cycle for ON Condition	SBS(091)
	Executed Once for Upward Differentiation	@SBS(091)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

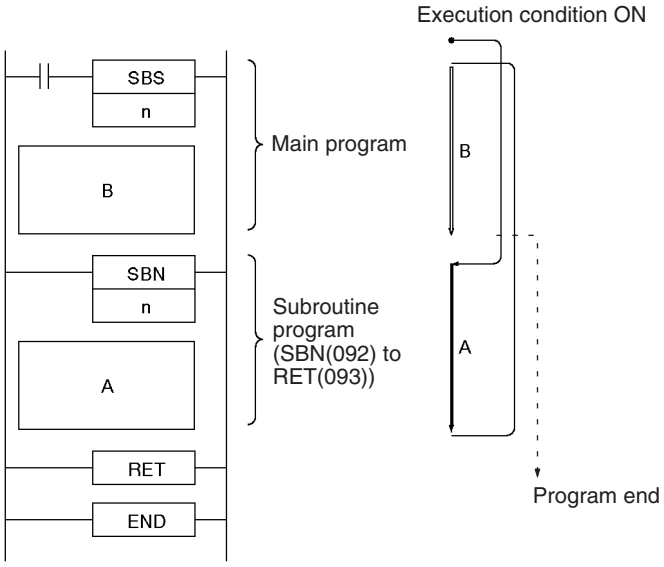
N: Subroutine number
Specifies the subroutine number between 0 and 255 decimal.

Operand Specifications

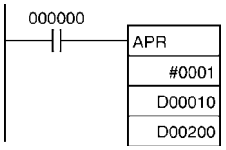
Area	N
CIO Area	---
Work Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	0 to 255 (decimal)
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	---

Description

SBS(091) calls the subroutine with the specified subroutine number. The subroutine is the program section between SBN(092) and RET(093). When the subroutine is completed, program execution continues with the next instruction after SBS(091).



Subroutines can be nested up to 16 levels. Nesting is when another subroutine is called from within a subroutine program, such as shown in the following example, which is nested to 3 levels.



Source data

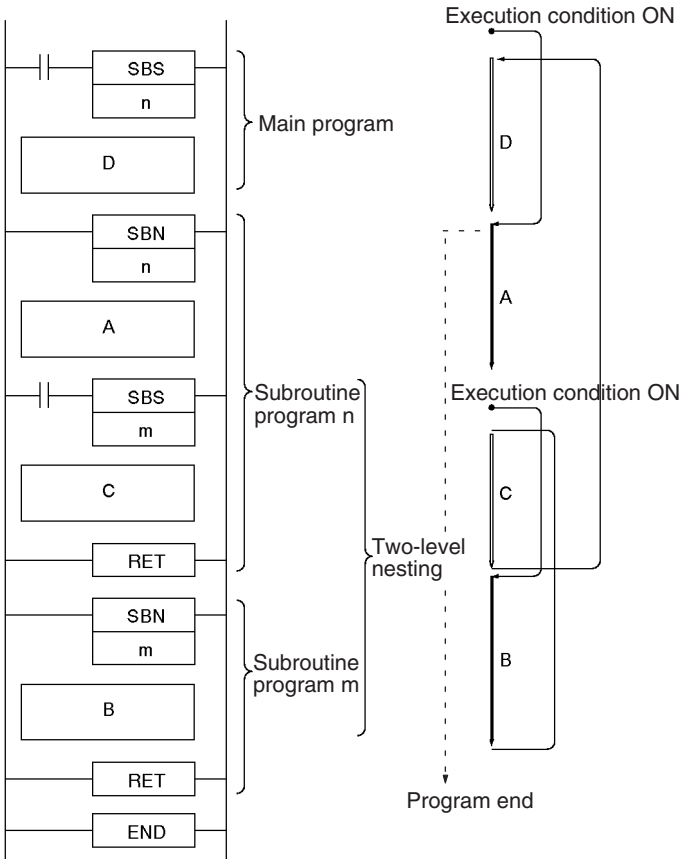
S: D00010			
0	10^{1}	10^{0}	10^{-1}
0	3	0	0

Set the source data in 10^{-1} degrees. (0000 to 0900, BCD)

Result

R: D00200			
10^{-1}	10^{-2}	10^{-3}	10^{-4}
8	6	6	0

Result data has four significant digits, fifth and higher digits are ignored. (0000 to 9999, BCD)

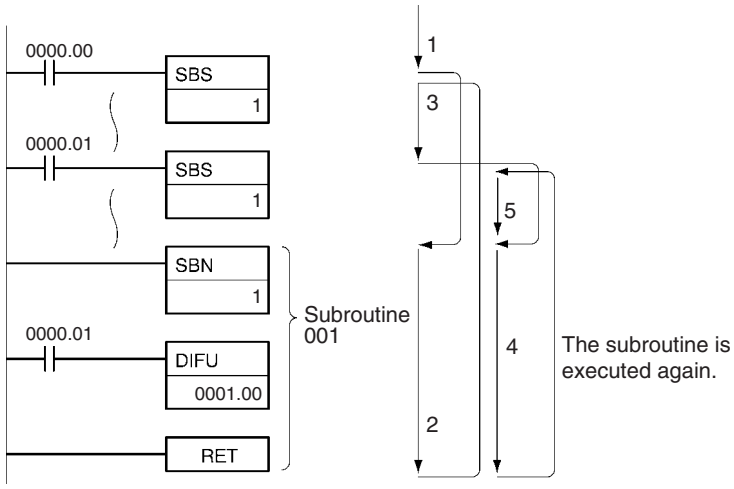


Note A subroutine can be called more than once in a program.

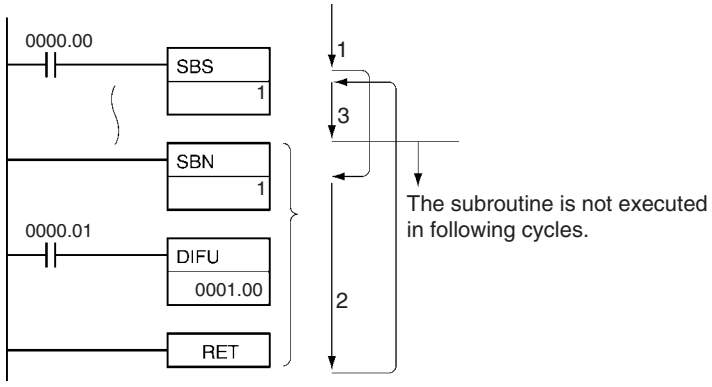
Subroutines and Differentiation

Observe the following precautions when using differentiated instructions (DIFU(013), DIFU(014), or up/down differentiated instructions) in subroutines.

The operation of differentiated instructions in a subroutine is unpredictable if a subroutine is executed more than once in the same cycle. In the following example, subroutine 001 is executed when CIO 0000.00 is ON and CIO 0001.00 is turned ON by DIFU(013) when CIO 0000.01 has gone from OFF to ON. If CIO 0000.01 is ON in the same cycle, subroutine 001 will be executed again but this time DIFU(013) will turn CIO 0001.00 OFF without checking the status of CIO 0000.01.



In contrast, the output of a differentiated instruction (DIFU(013) or DIFD(014)) would remain ON if the instruction was executed and the output was turned ON but the same subroutine was not called a second time.



In the following example, subroutine 001 is executed if CIO 0000.00 is ON. Output CIO 0001.00 is turned ON by DIFU(013) when CIO 0000.01 has gone from OFF to ON. If CIO 0000.00 is OFF in the following cycle, subroutine 001 will not be executed again and output CIO 0001.00 will remain ON.

Flags

Name	Label	Operation
Error Flag	ER	ON if nesting exceeds 16 levels. ON if the specified subroutine number does not exist. ON if a subroutine calls itself. ON if a subroutine being executed is called. ON if the specified subroutine is not defined in the current task. OFF in all other cases.

Precautions

SBS(091) and the corresponding SBN(092) must be programmed in the same task. An error will occur if the corresponding SBN(092) is not in the task.

SBS(091) will be treated as NOP(000) when it is within a program section interlocked by IL(002) and ILC(003).

When SBS(091) is executed in the following cases, the subroutine will not actually be called and the Error Flag will be turned ON:

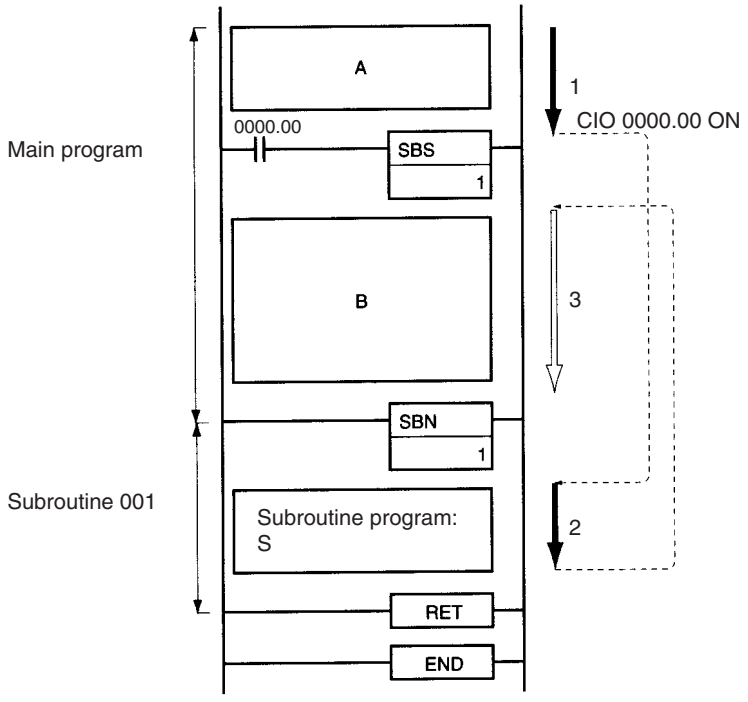
1,2,3...

1. The specified subroutine is not defined within the current task.
2. The subroutine is calling itself.
3. Subroutine nesting exceeds 16 levels.
4. The specified subroutine is being executed.

Examples

Example 1: Sequential (Non-nested) Subroutines

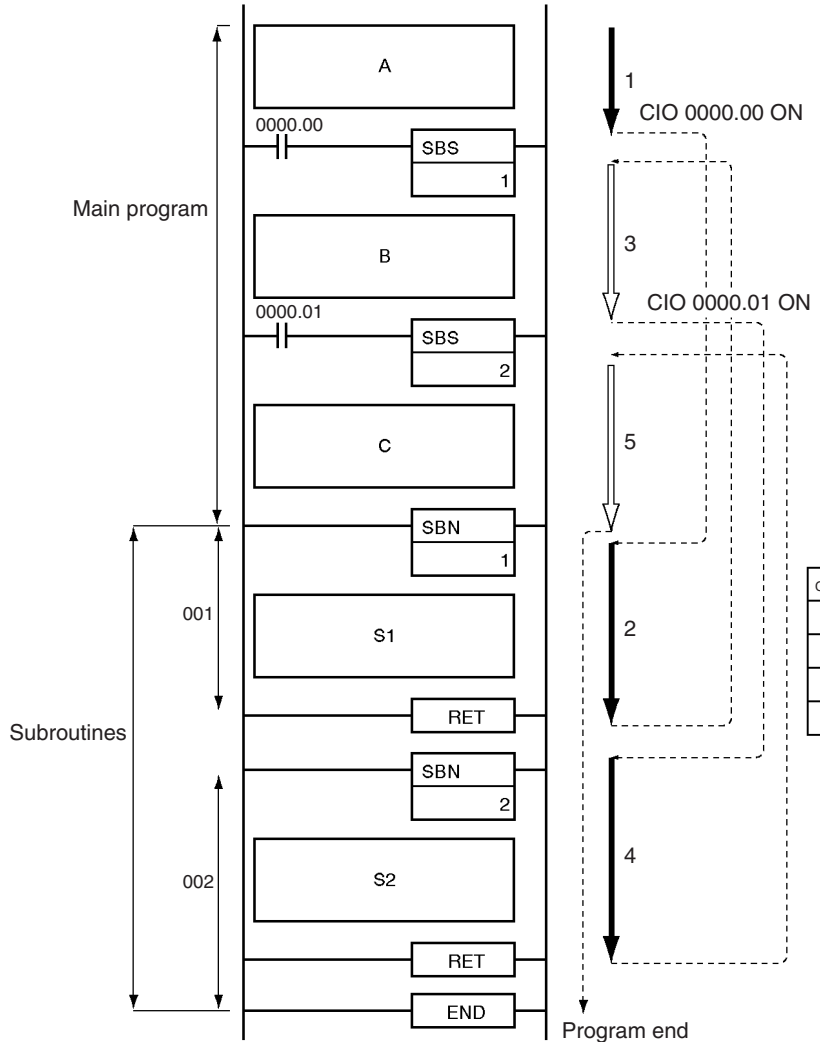
When CIO 0000.00 is ON in the following example, subroutine 001 is executed and program execution returns to the next instruction after SBS(091). The remainder of the main program (through the instruction just before SBN(092) 1) is then executed.



CIO 0000.00	Order of execution
ON	A→S→B
OFF	A→B

Example 2: Sequential (Non-nested) Subroutines

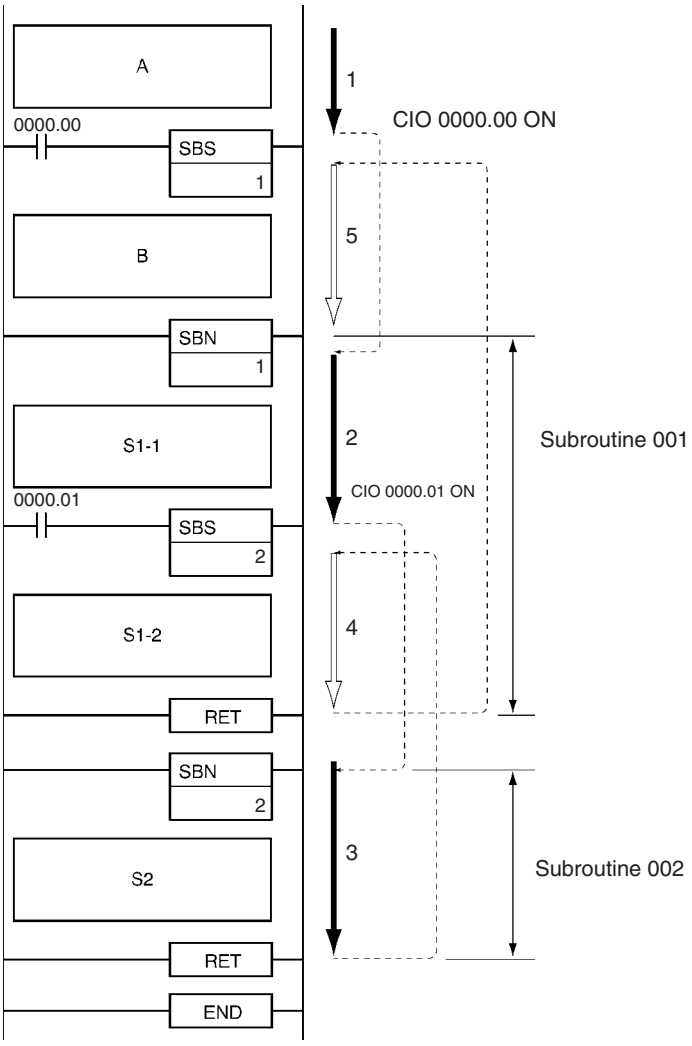
When CIO 0000.00 is ON in the following example, subroutine 001 is executed and program execution returns to the next instruction after SBS(091) 1. When CIO 0000.01 is ON, subroutine 002 is executed and program execution returns to the next instruction after SBS(091) 2.



CIO 0000.00	CIO 0000.01	Order of execution
ON	ON	A→S1→B→S2→C
ON	OFF	A→S1→B→C
OFF	ON	A→B→S2→C
OFF	OFF	A→B→C

Example 3: Nested Subroutines

When CIO 0000.00 is ON in the following example, subroutine 001 is executed. If CIO 0000.01 is ON, subroutine 002 is executed from within subroutine 001 and program execution returns to the next instruction after SBS(091) 2 when subroutine 002 is completed. Execution of subroutine 001 continues and program execution returns to the next instruction after SBS(091) 1 when subroutine 001 is completed.



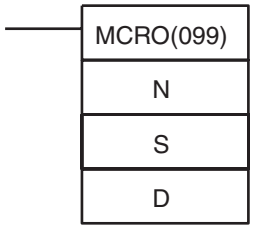
CIO 0000.00	CIO 0000.01	Order of execution
ON	ON	A→S1-1→S2→S1-2→B
ON	OFF	A→S1-1→S1-2→B
OFF	ON	A→B
OFF	OFF	A→B

3-18-2 MACRO: MCRO(099)

Purpose

Calls the subroutine with the specified subroutine number and executes that program using the input parameters in S to S+4 and the output parameters in D to D+4.

Ladder Symbol



N: Subroutine number

S: First input parameter word

D: First output parameter word

Variations

Variations		
	Executed Each Cycle for ON Condition	MCRO(099)
	Executed Once for Upward Differentiation	@MCRO(099)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

N: Subroutine number
Specifies the subroutine number between 0 and 255 decimal.

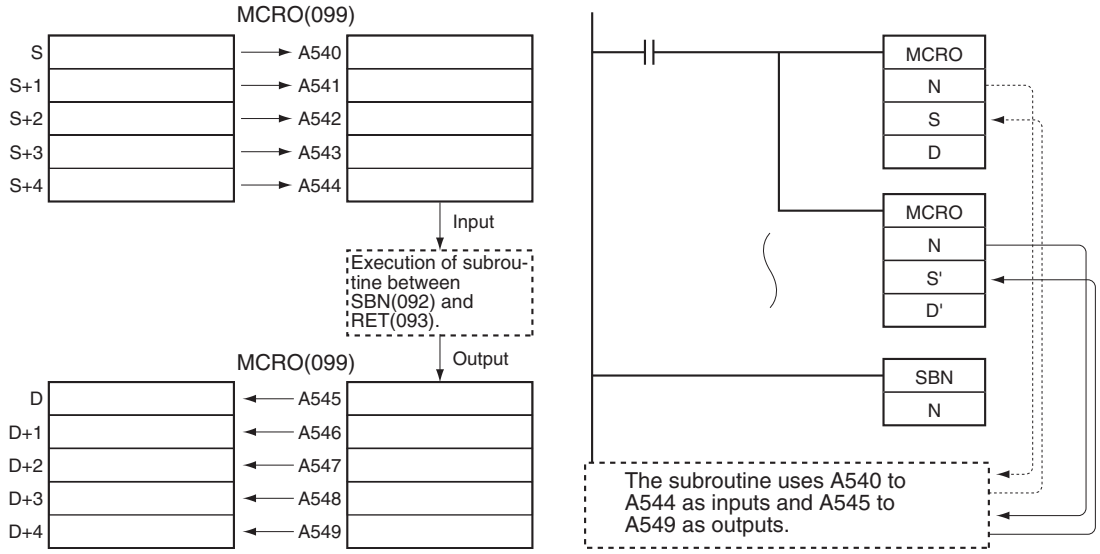
Operand Specifications

Area	N	S	D
CIO Area	---	CIO 0000 to CIO 6139	
Work Area	---	W000 to W251	
Auxiliary Bit Area	---	A000 to A443 A448 to A955	A448 to A955
Timer Area	---	T0000 to T0251	
Counter Area	---	C0000 to C0251	
DM Area	---	D00000 to D32763	
Indirect DM addresses in binary	---	@ D00000 to @ D32767	
Indirect DM addresses in BCD	---	*D00000 to *D32767	
Constants	0 to 255 (decimal)	---	
Data Resisters	---		
Index Registers	---		
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

MCRO(099) calls the subroutine with the specified subroutine number just like SBS(091). Unlike SBS(091), MCRO(099) operands S and D can be used to change bit and word addresses in the subroutine, although the structure of the subroutine is constant.

When MCRO(099) is executed, the contents of S through S+4 are copied to A540 through A544 (macro area inputs) and the specified subroutine is executed. When the subroutine is completed, the contents of A545 through A549 (macro area outputs) are copied to D through D+4 and program execution continues with the next instruction after MCRO(099).



MCRO(099) can be used to consolidate two or more subroutines with the same structure but different input and output addresses into a single subroutine program. When MCRO(099) is executed, the specified input and output data is transferred to the specified subroutine.

Flags

Name	Label	Operation
Error Flag	ER	ON if nesting exceeds 16 levels. ON if the specified subroutine number does not exist. ON if a subroutine calls itself. ON if a subroutine being executed is called. ON if the specified subroutine is not defined in the current task. OFF in all other cases.

The following table shows relevant words in the Auxiliary Area.

Name	Address	Operation
Macro area input words	A540 to A544	When MCRO(099) is executed the five words from S to S+4 are copied to A510 to A514. These input words are passed to the subroutine.
Macro area input words	A545 to A549	After the subroutine specified in MCRO(099) has been executed, the output data in these output words and copied to D to D+4.

Precautions

The five words of input data (words or bits) in A540 to A544 and the five words of output data (words or bits) in A545 to A549 must be used in the subroutine called by MCRO(099). It is not possible to pass more than five words of data.

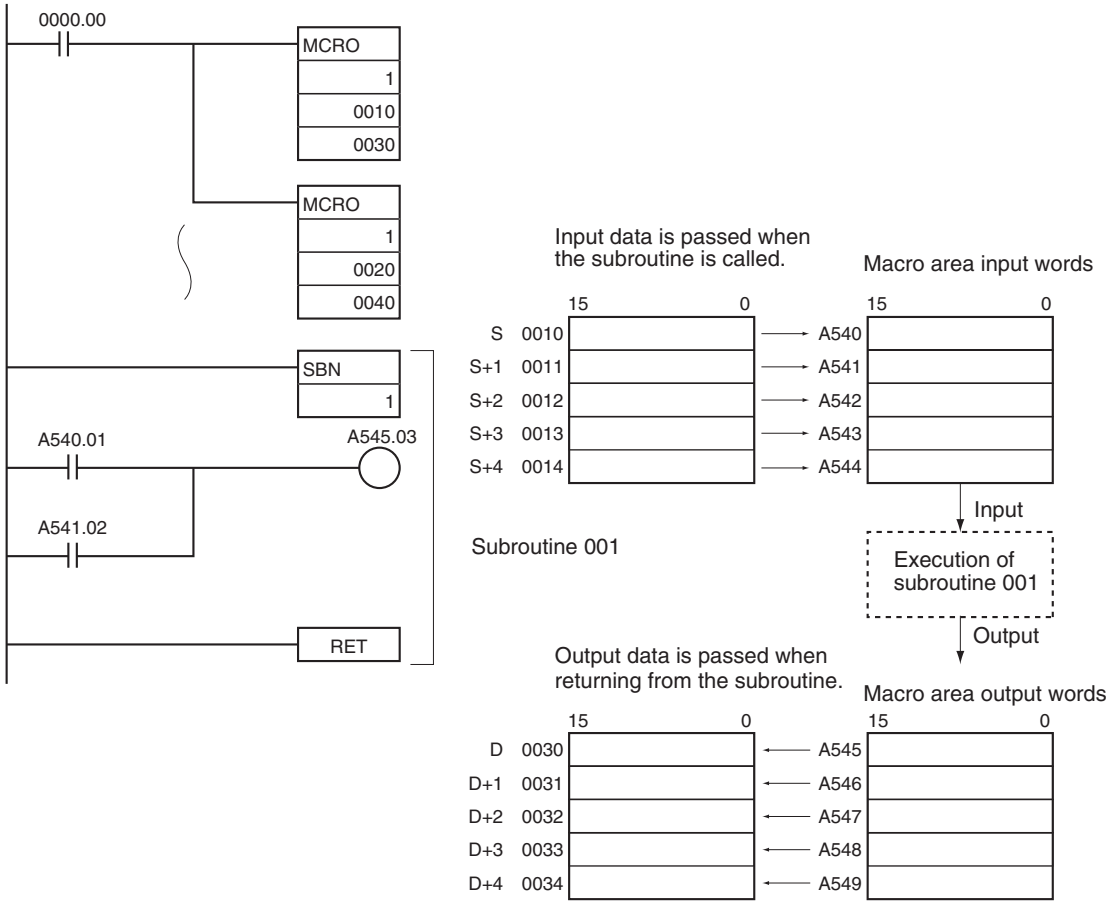
It is possible to nest MCRO(099) instructions, but the data in the macro area input and output words (A540 to A549) must be saved when calling nested subroutine because all MCRO(099) instructions use the same 10 words.

Example

When CIO 0000.00 is ON in the following example, two MCRO(099) instructions pass different input and output data to subroutine 001.

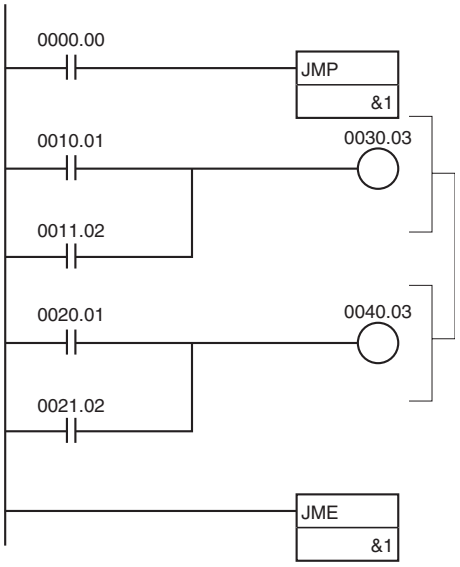
1,2,3...

1. The first MCRO(099) instruction passes the input data in CIO 0010 to CIO 0014 and executes the subroutine. When the subroutine is completed, the output data is stored in CIO 0030 to CIO 0034.
2. The second MCRO(099) instruction passes the input data in CIO 0020 to CIO 0024 and executes the subroutine. When the subroutine is completed, the output data is stored in CIO 0040 to CIO 0044.



The second MCRO(099) instruction operates in the same way, but the input data in CIO 0020 to CIO 0024 is passed to A540 to A544 and the output data in A545 to A549 is passed to CIO 0040 to CIO 0044.

The above programming is equivalent to the following programming.

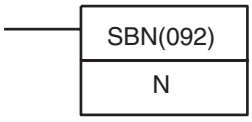


3-18-3 SUBROUTINE ENTRY: SBN(092)

Purpose

Indicates the beginning of the subroutine program with the specified subroutine number. Used in combination with RET(093) to define a subroutine region.

Ladder Symbol



N: Subroutine number

Variations

Variations	Executed Each Cycle for ON Condition	SBN(092)

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	Not allowed	Not allowed	OK

Operands

N: Subroutine number
Specifies the subroutine number between 0 and 255 decimal.

Operand Specifications

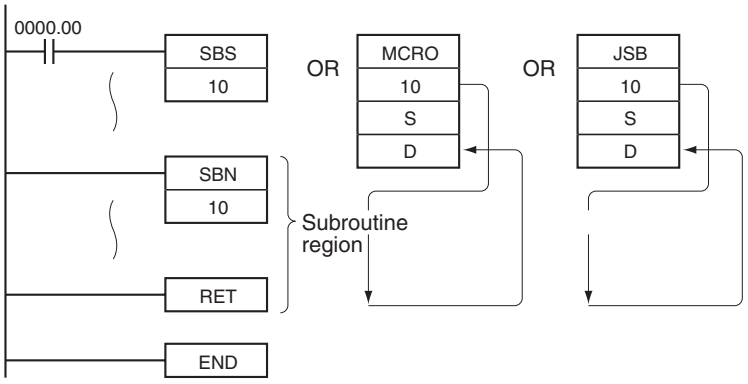
Area	N
CIO Area	---
Work Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	0 to 255 (decimal)
Data Resisters	---

Area	N
Index Registers	---
Indirect addressing using Index Registers	---

Description

SBN(092) indicates the beginning of the subroutine with the specified subroutine number. The end of the subroutine is indicated by RET(093).

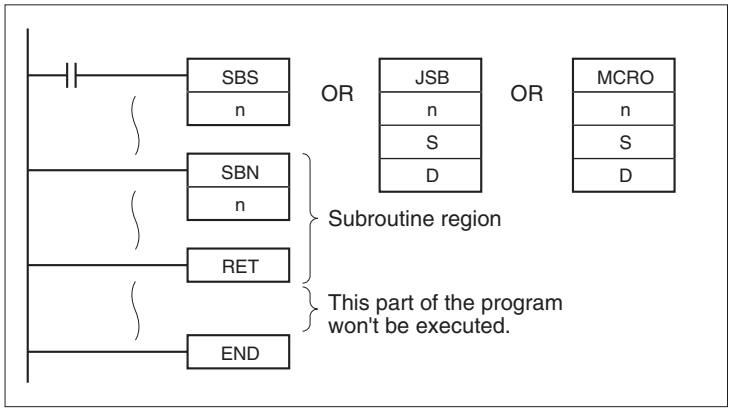
The region of the program beginning at the first SBN(092) instruction is the subroutine region. A subroutine is executed only when it has been called by SBS(091), JSB(982), or MCRO(099).



Precautions

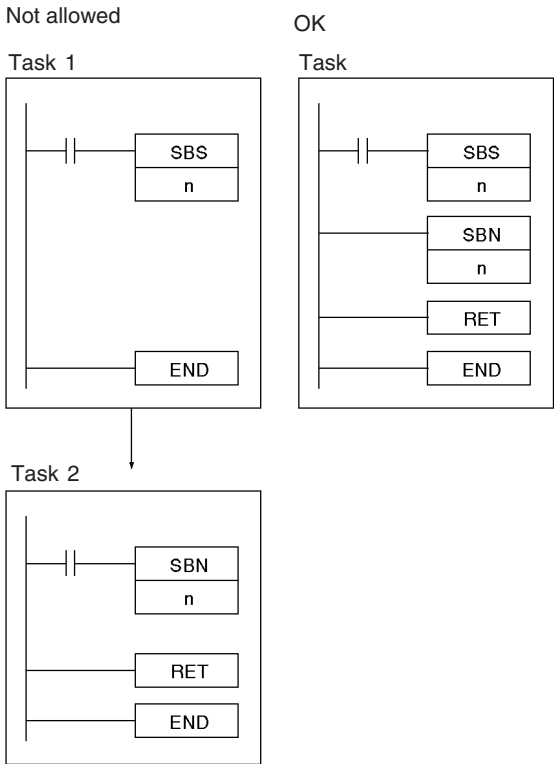
When the subroutine is not being executed, the instructions are treated as NOP(000).

Place the subroutines after the main program and just before the END(001) instruction in the program for each task. If part of the main program is placed after the subroutine region, that program section will be ignored.

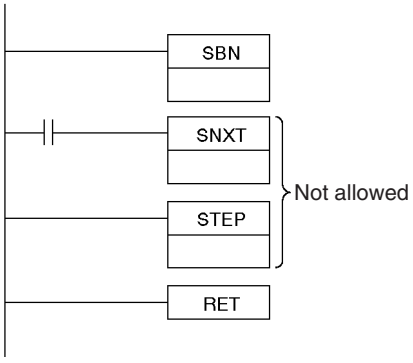


Note Input #0 to #1023 on the CX-Programmer to input the subroutine number, N.

Be sure to place each subroutine in the same program (task) as its corresponding SBS(091), JSB(982), or MCRO(099) instruction. A subroutine in one task cannot be called from another task. It is possible to program a subroutine within an interrupt task.

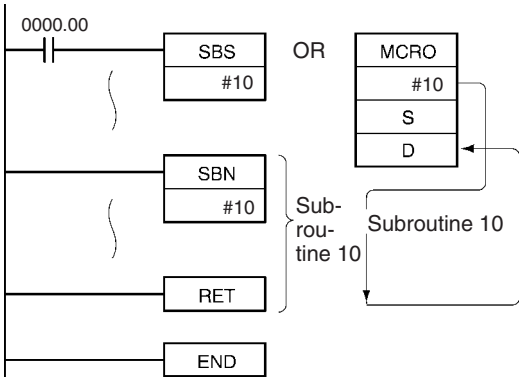


The step instructions, STEP(008) and SNXT(009) cannot be used in subroutines.



Example

When CIO 0000.00 is ON in the following example, subroutine 10 is executed and program execution returns to the next instruction after the SBS(091) or MCRO(099) instruction that called the subroutine.



3-18-4 SUBROUTINE RETURN: RET(093)

Purpose

Indicates the end of a subroutine program. Used in combination with SBN(092) to define a subroutine region.

Ladder Symbol

Variations

Variations	Executed Each Cycle for ON Condition	RET(093)

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	Not allowed	OK	OK

Description

RET(093) indicates the end of a subroutine and SBN(092) indicates the beginning of a subroutine. See *3-18-3 SUBROUTINE ENTRY: SBN(092)* for more details on the operation of subroutines.

When program execution reaches RET(093), it is automatically returned to the next instruction after the SBS(091), JSB(982), or MCRO(099) instruction that called the subroutine. When the subroutine has been called by MCRO(099), the output data in A545 through A549 is written to D through D+4 before program execution is returned.

Precautions

When the subroutine is not being executed, the instructions are treated as NOP(000).

Example

See *3-18-3 SUBROUTINE ENTRY: SBN(092)* for examples of the operation of RET(093).

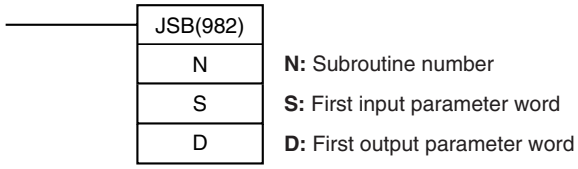
3-18-5 JUMP SUBROUTINE: JSB(982)

Purpose

Always calls the subroutine with the specified subroutine number regardless of the ON/OFF status of the execution condition.

The ON/OFF status of the execution condition is stored in the bit between A019.00 and A034.15 that corresponds to the specified subroutine number (between 0 and 255).

Ladder Symbol



Associated Instructions

- SUBROUTINE ENTRY: SBN(092)
- SUBROUTINE RETURN: RET(093)

Variations

Variations	Executed Each Cycle	JSB(982)

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

N: Subroutine number
Specifies the subroutine number between 0 and 255 decimal.

S: First input parameter word
Specifies the first word of the data to be passed to the subroutine.
The memory address of the specified word is set in Index Register IR0.

D: First output parameter word
Specifies the first word of the data to be passed out of the subroutine.
The memory address of the specified word is set in Index Register IR0.

Operand Specifications

Area	N	S	D
CIO Area	---	CIO 0000 to CIO 6143	
Work Area	---	W000 to W255	
Auxiliary Bit Area	---	A000 to A959	A448 to A959
Timer Area	---	T0000 to T0255	
Counter Area	---	C0000 to C0255	
DM Area	---	D00000 to D32767	
Indirect DM addresses in binary	---	@D00000 to @D32767	
Indirect DM addresses in BCD	---	*D00000 to *D32767	
Constants	0 to 255 (decimal)	---	---
Data Resisters	---	---	---
Index Registers	---	---	
Indirect addressing using Index Registers	---	,IR2 to ,IR15 –2048 to +2047 ,IR2 to –2048 to +2047 ,IR15 DR0 to DR15, IR2 to DR0 to DR15, IR15 ,IR2+(++) to ,IR15+(++) ,–(– –)IR2 to , –(– –)IR15	

Description

The specified subroutine is always started, so the JSB(982) instruction's execution condition does not control execution of the instruction. Instead, the execution condition is automatically stored in a corresponding Auxiliary Area bit (A019.00 and A034.15) as the Powerflow Flag. That Powerflow Flag can be used in the subroutine as a switch to perform different processing.

JSB(982) will be executed even when the execution condition is OFF. JSB(982) stores the status of the execution condition in the corresponding bit (A019.00 and A034.15 correspond to subroutine numbers 0 to 255) and calls the specified subroutine. The execution condition (Powerflow Flag stored in A019.00 and A034.15) can be used in the subroutine to control processing.

Example 1:
When the execution condition is ON, jogging is performed. When the execution condition is OFF, the axis is stopped or decelerated.

Example 2:
When the execution condition goes from OFF to ON, a communications instruction is executed. Even if the execution condition is OFF, the communications instruction will receive the response and communications processing will be monitored until it is completed.

Using Index Registers for General-purpose Subroutines

With JSB(982), the words containing the input data can be specified and the memory address of the beginning word is automatically stored in IR0 so the

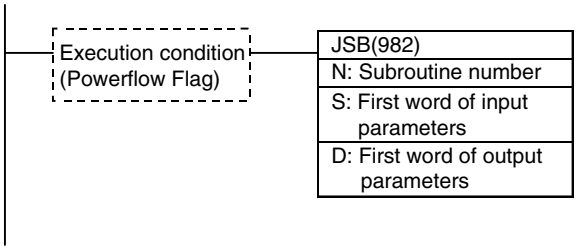
specified subroutine can be used repeatedly. When Index Register IR0 is indirectly addressed in the subroutine, the subroutine accesses the first word containing the desired input data and that input data is read and processed.

The output parameter words can be specified in the same way, and the beginning word for the output data is automatically stored in IR1. Index Register IR1 can be indirectly addressed in the subroutine to write the processing results to the desired words as output data.

Note

1. With SBS(091), the subroutine is completely skipped when the execution condition (Powerflow Flag) is OFF. Therefore, if SBS(091) is used to call the subroutine, any processing required when the execution condition is OFF cannot be included in the subroutine. (For example, a subroutine that performs jogging when the execution condition is ON cannot contain processing such as stopping or decelerating an axis required when the execution condition is OFF).
2. With SBS(091), there is no function in the subroutine to indicate whether or not the subroutine is being executed for the first time. Therefore, if SBS(091) is used to call the subroutine, it is not possible to divide processing over several cycles in that subroutine.

Operation of JSB(982)



Note JSB(982) will be executed even when the execution condition (Powerflow Flag) is OFF.

When JSB(982) is executed, it performs the following operations:

1,2,3...

1. The subroutine is started and the corresponding Subroutine Powerflow Flag (A019.00 and A034.15) is turned ON.

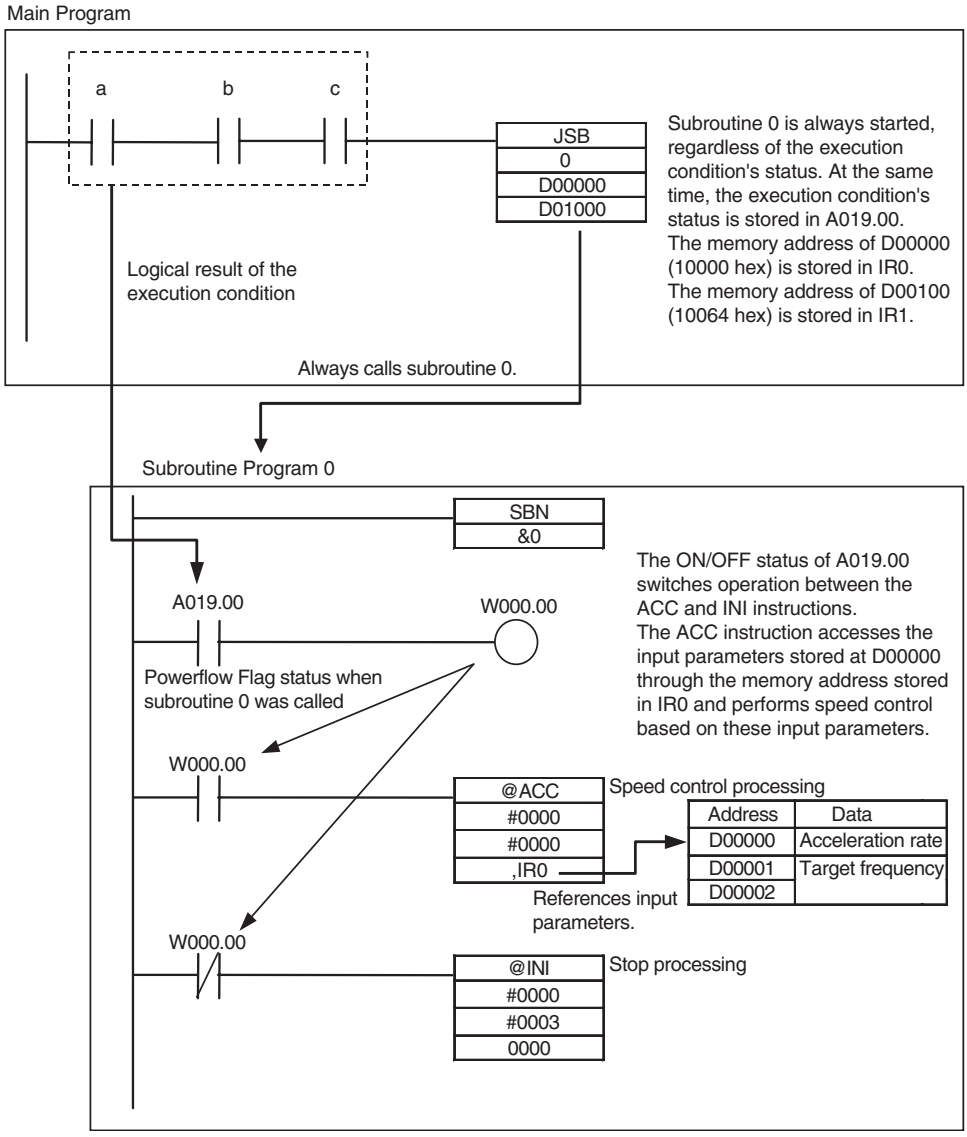
Address		Corresponding subroutine number
Word	**Bit**	
A019	00 to 15	SBN000 to SBN015
A020	00 to 15	SBN016 to SBN031
A021	00 to 15	SBN032 to SBN047
to		to
A034	00 to 15	SBN240 to SBN255

2. The memory addresses of the first input parameter word (S) and first output parameter word (D) are stored in Index Registers IR0 and IR1, respectively.
3. The specified subroutine is executed up to RET(093), the SUBROUTINE RETURN instruction.
4. The JSB(982) instruction ends.

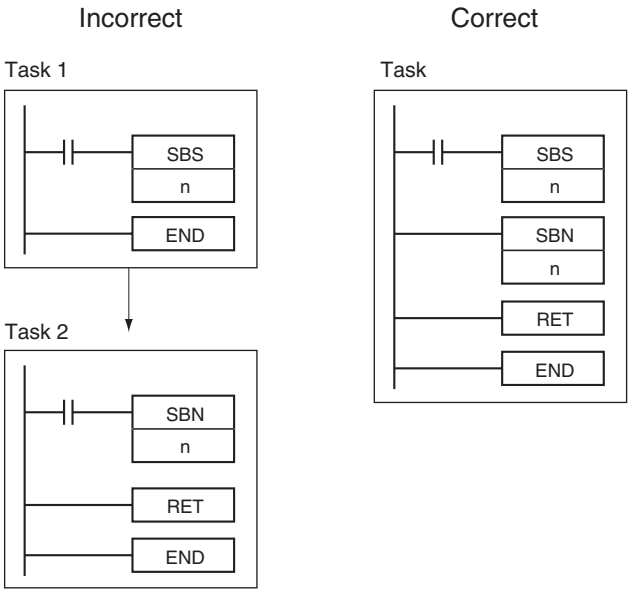
Note If JSB(982) is placed in a program section between IL(002) and ILC(003), JSB(982) will execute the subroutine even when the program section is interlocked. The contents of the destination subroutine will be interlocked.

Flags None of the Condition Flags are affected by this instruction.

Example: Executing JSB(982) with a Powerflow Flag



Example Task Structure



3-19 Interrupt Control Instructions

This section describes instructions used to control interrupts and interrupt timers.

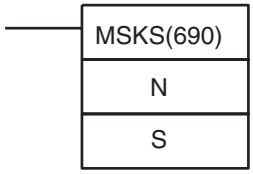
Instruction	Mnemonic	Function code	Page
SET INTERRUPT MASK	MSKS	690	508
READ INTERRUPT MASK	MSKR	692	510
CLEAR INTERRUPT	CLI	691	512
DISABLE INTERRUPTS	DI	693	513
ENABLE INTERRUPTS	EI	694	514
INTERVAL TIMER	STIM	980	516

3-19-1 SET INTERRUPT MASK: MSKS(690)

Purpose

Both I/O interrupt tasks and scheduled interrupt tasks are masked (disabled) when the Module enters RUN mode. MSKS(690) can be used to unmask or mask I/O interrupts. MSKS(690) can be used only for FQM1-MMP22 and FQM1-MMA22 Motion Control Modules.

Ladder Symbol



N: Interrupt identifier

S: Interrupt data

Variations

Variations		
	Executed Each Cycle for ON Condition	MSKS(690)
	Executed Once for Upward Differentiation	@MSKS(690)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Specifying I/O Interrupt Processing and Mask Processing

Operand	Contents
N	Specify the interrupt input number. #0000: Interrupt input 0 #0001: Interrupt input 1 #0002: Interrupt input 2 #0003: Interrupt input 3 #0004: Phase-Z input counter clear interrupt for counter 1 (See note 1.) #0005: Phase-Z input counter clear interrupt for counter 2 (See note 1.)
S	Interrupt mask. 0000 hex: Interrupt enabled 0001 hex: Interrupt masked 0002 hex: Decrementing counter started and interrupts enabled (See note 2.)

Note

1. These settings can be used only in CPU Units with unit version 3.2 or later.
2. S cannot be set to #0002 if N has been set to #0004 or #0005.

The relationship between interrupt input numbers and interrupt task numbers is shown in the following table.

Interrupt input number	Interrupt task numbers	
Interrupt input 0	000	CIO 2960.00
Interrupt input 1	001	CIO 2960.01
Interrupt input 2	002	CIO 2960.02
Interrupt input 3	003	CIO 2960.03
Phase-Z input counter clear interrupt for counter 1	004	If the reset method is set to *Phase-Z signal + software reset*, the task starts when a counter reset is performed by the phase-Z signal.
Phase-Z input counter clear interrupt for counter 2	005	

Operand Specifications

Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W255
Auxiliary Bit Area	---	A000 to A959
Timer Area	---	T0000 to T0255
Counter Area	---	C0000 to C0255
DM Area	---	D00000 to D32767
Indirect DM addresses in binary	---	@ D00000 to @ 32767
Indirect DM addresses in BCD	---	*D00000 to *D32767
Constants	Specified values only	#0 to #2
Data Resisters	---	DR0 to DR15
Index Registers	---	
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Flags

Name	Label	Operation
Error Flag	ER	ON if the specified range exceeds an area boundary. ON if the specified value exceeds the allowable range. ON if N is not within the specified range of 0 to 5 hex. ON if S is not within the specified range of 0 to 2 hex. OFF in all other cases.
Equals Flag	=	OFF
Negative Flag	N	OFF

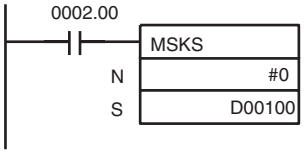
System Setup Settings

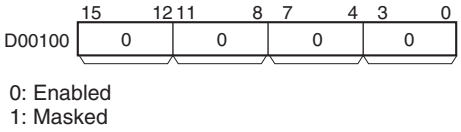
Name	Description	Settings
Interrupt Input Settings	Specify whether built-in inputs are to be used as normal inputs or as interrupt inputs.	00 hex: Normal (default) 01 hex: *Interruption* (up) (interrupt when input turns ON) 02 hex: *Interruption* (down) (interrupt when input turns OFF) 03 hex: *Interruption* (both edge) (interrupt when input turns ON or OFF)
Pulse Input Setting	Specify the counter's reset method. An interrupt task can be started only when this value is set to 1 hex.	0 hex: Software reset (default setting) 1 hex: Phase-Z signal + software reset

Examples

Examples

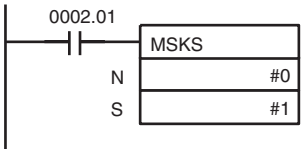
When CIO 0002.00 turns ON in the following example, MSKS(690) unmasks (enables) interrupt input 0.





0: Enabled
1: Masked

When CIO 0002.01 turns ON in the following example, MSKS(690) disables interrupt input 0.

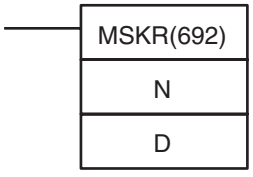


3-19-2 READ INTERRUPT MASK: MSKR(692)

Purpose

Reads the current interrupt processing settings that were set with MSKS(690).

Ladder Symbol



N: Interrupt identifier

D: Destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	MSKR(692)
	Executed Once for Upward Differentiation	@MSKR(692)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Reading Masks

Operand	Contents
N	Specify the interrupt input number. #0000: Interrupt input 0 #0001: Interrupt input 1 #0002: Interrupt input 2 #0003: Interrupt input 3 #0004: Phase-Z input counter clear interrupt for counter 1 (See note.) #0005: Phase-Z input counter clear interrupt for counter 2 (See note.)

Note These settings can be used only in CPU Units with unit version 3.2 or later.

The relationship between interrupt input numbers and interrupt task numbers is shown in the following table.

Interrupt input number	Interrupt task numbers	
Interrupt input 0	000	CIO 2960.00
Interrupt input 1	001	CIO 2960.01
Interrupt input 2	002	CIO 2960.02
Interrupt input 3	003	CIO 2960.03
Phase-Z input counter clear interrupt for counter 1	004	If the reset method is set to *Phase-Z signal + software reset*, the task starts when a counter reset is performed by the phase-Z signal.
Phase-Z input counter clear interrupt for counter 2	005	

Operand Specifications

Area	N	D
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W255
Auxiliary Bit Area	---	A448 to A959
Timer Area	---	T0000 to T0255
Counter Area	---	C0000 to C0255
DM Area	---	D00000 to D32767
Indirect DM addresses in binary	---	@ D00000 to @ D32767
Indirect DM addresses in BCD	---	*D00000 to *D32767
Constants	Specified values only	---
Data Resisters	---	DR0 to DR15
Index Registers	---	
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

MSKR(692) reads the interrupt input settings specified with MSKS(690). The status (disabled/enabled) of the input interrupt specified with N is output to D.

Flags

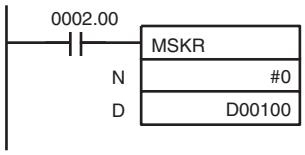
Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0 to 5. OFF in all other cases.

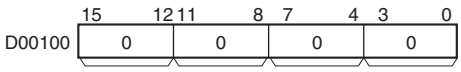
Precautions

MSKR(692) can be executed in the main program or in interrupt tasks.

Examples

When CIO 0002.00 turns ON in the following example, MSKR(692) reads the current mask status of Interrupt Input 0 and stores it in D00100.





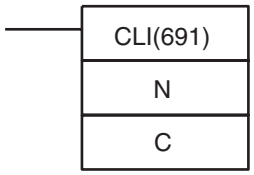
0000: Interrupt enabled
0001: Interrupt masked

3-19-3 CLEAR INTERRUPT: CLI(691)

Purpose

Clears or retains recorded interrupt inputs.

Ladder Symbol



N: Interrupt number

C: Control data

Variations

Variations		
	Executed Each Cycle for ON Condition	CLI(691)
	Executed Once for Upward Differentiation	@CLI(691)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operand Specifications

Operand	Contents
N	Specify the interrupt input number. #0000: Interrupt input 0 #0001: Interrupt input 1 #0002: Interrupt input 2 #0003: Interrupt input 3 #0004: Phase-Z input counter clear interrupt for counter 1 (See note.) #0005: Phase-Z input counter clear interrupt for counter 2 (See note.)
C	Control data: Interrupt mask clear specification. 0000 hex: Recorded interrupt input retained. 0001 hex: Recorded interrupt input cleared.

Note These settings can be used only in CPU Units with unit version 3.2 or later.

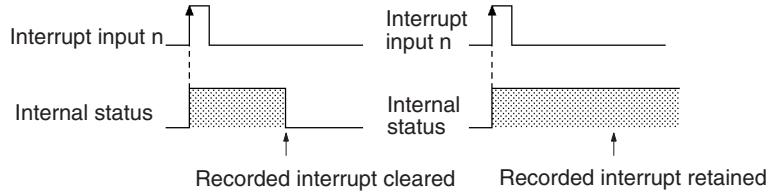
Area	N	C
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W255
Auxiliary Bit Area	---	A000 to A959
Timer Area	---	T0000 to T0255
Counter Area	---	C0000 to C0255
DM Area	---	D00000 to D32767
Indirect DM addresses in binary	---	@ D00000 to @ D32767

Area	N	C
Indirect DM addresses in BCD	---	*D00000 to *D32767
Constants	Refer to the previous table.	
Data Resisters	---	DR0 to DR15
Index Registers	---	
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

CLI(691) clears or retains the specified recorded input interrupt.

Values #0000 to #0003 correspond to interrupt inputs 0 to 3.
CLI(691) clears a recorded interrupt input when the corresponding bit of C is ON and retains the recorded interrupt input when the corresponding bit is OFF.



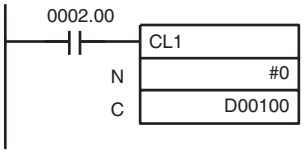
If an interrupt task is being executed and an interrupt input with a different interrupt number is received, that interrupt number is recorded internally. The recorded interrupts are executed later in order of their priority (from the lowest number to the highest). CLI(691) can be used to clear these recorded interrupts before they are executed.

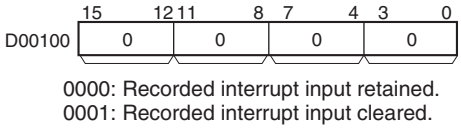
Flags

Name	Label	Operation
Error Flag	ER	ON if N is not between 0 to 5. ON if C is not between 0000 and 0001 hex. OFF in all other cases.

Examples

When CIO 0002.00 turns ON in the following example, CLI(691) clears the recorded interrupts for interrupt input 0.



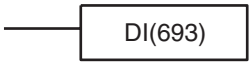


3-19-4 DISABLE INTERRUPTS: DI(693)

Purpose

Disables execution of all interrupt tasks.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	DI(693)
	Executed Once for Upward Differentiation	@DI(693)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	Not allowed

Description

DI(693) is executed from the main program to temporarily disable all interrupt tasks (input interrupt tasks, interval timer interrupt tasks, pulse output interrupt tasks, and high-speed counter interrupt tasks). All interrupt tasks will be disabled until they are enabled again by execution of EI(694).

Note

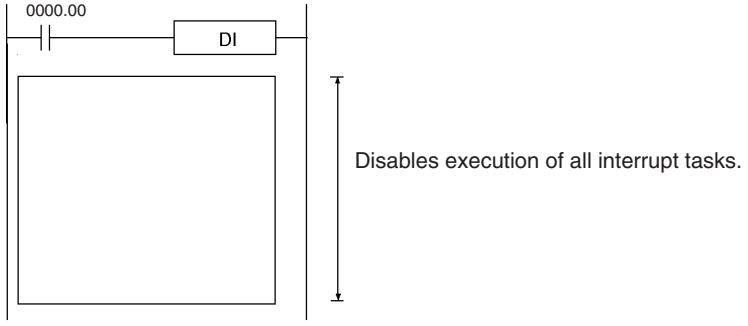
1. Use EI(694) to enable interrupts again.
2. DI(693) cannot be executed in an interrupt task. Attempting to do so will cause an error and the Error Flag will turn ON.

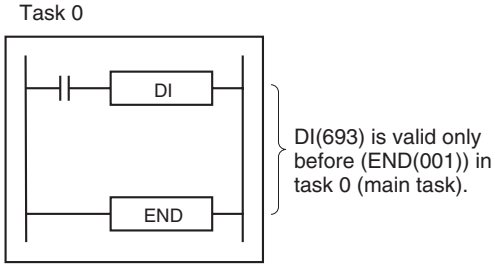
Flags

Name	Label	Operation
Error Flag	ER	ON if DI(693) is executed from an interrupt task. OFF in all other cases.

Examples

When CIO 0000.00 is ON in the following example, DI(693) disables all interrupt tasks.



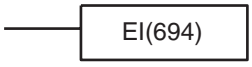


3-19-5 ENABLE INTERRUPTS: EI(694)

Purpose

Enables execution of all interrupt tasks.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for Normally ON Condition	EI(694)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	Not allowed

Description

EI(694) is executed from the main program to enable all interrupt tasks (input interrupt tasks, interval timer interrupt tasks, pulse output interrupt tasks, and high-speed counter interrupt tasks) that were disabled by DI(693).

Note

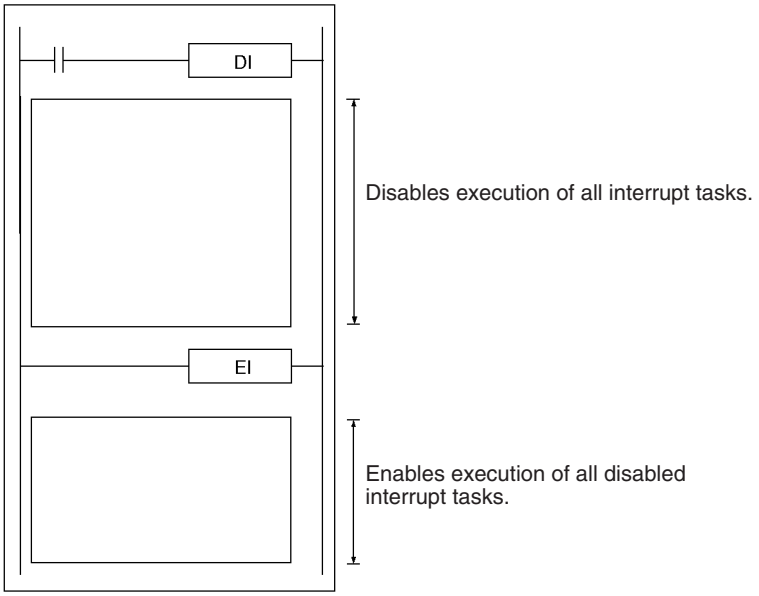
1. EI(694) does not require an execution condition.
2. EI(694) will not enable interrupt tasks for which MSKS(690) has disabled interrupt inputs, STIM(980) has disabled interval timer interrupts, or CTBL(882) has disabled target value comparisons.
3. EI(694) cannot be executed in an interrupt task. Attempting to do so will cause an error and the Error Flag will turn ON.

Flags

Name	Label	Operation
Error Flag	ER	ON if EI(694) is executed from an interrupt task. OFF in all other cases.

Examples

In the following example, EI(694) enables all interrupt tasks that were disabled by DI(693).



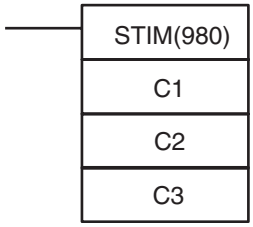
3-19-6 INTERVAL TIMER: STIM(980)

Purpose

STIM(980) is used to control the interval timers and pulse outputs with the following functions.

Function	Description
Basic functions	Starting the one-shot interrupt timer, starting the scheduled interrupt timer, reading a timer PV, and stopping an interrupt timer
Pulse output functions (FQM1-MMP22 only)	Starting the one-shot pulse output and starting/stopping the pulse counter timer

Ladder Symbol



C1: Control data #1

C2: Control data #2

C3: Control data #3

Variations

Variations		
	Executed Each Cycle for ON Condition	STIM(980)
	Executed Once for Upward Differentiation	@STIM(980)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

C1: Control Word 1

The contents of control word #1 are shown below.

Value of C1	Control function
#0000	Start one-shot timer.
#0001	One-shot pulse output 1 (FQM1-MMP22 only)
#0002	One-shot pulse output 2 (FQM1-MMP22 only)
#0003	Start scheduled interrupt timer.
#0006	Read timer PV.
#000A	Stop timer.
#000B	Start or stop pulse counter timer 1 (FQM1-MMP22 only).
#000C	Start or stop pulse counter timer 2 (FQM1-MMP22 only).

C2 and C3: Control Words 2 and 3

The functions of C2 and C3 depend upon the control function setting in C1.

- C1 = #0000 or #0003 (Start one-shot timer or scheduled interrupt timer.)

Operand	Contents
C2	Timer SV (first word)
C3	Interrupt task number 0000 to 0031 hexadecimal (0 to 49 decimal)

The following table shows the settings for C2 when specifying a word address or constant.

C2	Settings
Word address	C2: Initial value of decrementing counter 0001 to 270F Hex C2+1: Decrementing time interval in 0.1 ms units 0005 to 0064 Hex (0.5 to 10.0 ms)
Constant	Decrementing time interval in 0.1 ms units 0005 to 0064 Hex (0.5 to 10.0 ms)

Note a) The total time from the execution of STIM(980) to time-up is:
(Content of C2) × (Content of C2+1) × 0.1 ms)
= 0.5 to 99,990 ms

b) If a constant is set for C2, C2 contains the decrementing time interval between 0.5 and 10.0 ms. (The SV is set directly in 0.1 ms units.)

c) A timer interrupt function (one-shot timer or scheduled interrupt timer) executed with STIM(980) can be used simultaneously with the one-shot pulse output or pulse counter time measurement function.

- C1 = #0001 or #0002 (One-shot pulse output 1 or 2)

Operand	Contents
C2	ON time setting: 0001 to 270F Hex
C3	Time units 0000 Hex: 0.1 ms 0001 Hex: 0.01 ms 0002 Hex: 0.1 ms 0003 Hex: 1 ms 0004 Hex: 0.001 ms

- C1 = #0006 (Read timer PV.)

Operand	Contents
C2	First word containing PV parameter 1
C3	Word containing PV parameter 2

- Contents of PV parameter 1 words

Word	Settings
C2	Number of times that the decrementing timer value has been decremented (4-digit hexadecimal)
C2+1	Decrementing time interval (0.1 ms units, 4-digit hexadecimal)

- Content of PV parameter 2 word:

Word	Settings
C3	Elapsed time since last decrement operation (0.1 ms units, 4-digit hexadecimal)

The elapsed time since the interval timer started is:
((Content of C2) × (Content of C2+1)) + ((Content of C3) × 0.1 ms)

- C1 = #000A (Stop timer.)

In this case, set both C2 and C3 to 0000.

• C1 = #000B or #000C (Pulse counter timer 1 or 2)

Operand	Contents
C2	Start or stop counter. #0000: Start #0001: Stop
C3	Time units 0000 hex: 0.1 ms 0001 hex: 0.01 ms 0002 hex: 0.1 ms 0003 hex: 1 ms 0004 hex: 0.001 ms

Operand Specifications

Area	C1	C2	C3
CIO Area	---	CIO 0000 to CIO 6143	
Work Area	---	W000 to W255	
Auxiliary Bit Area	---	A000 to A959	
Timer Area	---	T0000 to T0255	
Counter Area	---	C0000 to C0255	
DM Area	---	D00000 to D32767	
Indirect DM addresses in binary	---	@D00000 to @D32767	
Indirect DM addresses in BCD	---	*D00000 to *D32767	
Constants	#0 to #C	#0000 to #270F	#0 to #31
Data Resisters	---	---	DR0 to DR15
Index Registers	---		
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

STIM(980) is used to control the interval timers and pulse outputs according to the parameters specified in C1 (control data 1), C2 (control data 2), and C3 (control data 3).

Note To use STIM(980) to control one-shot pulse outputs, the pulse output operation mode must be set to *1 shot* in advance in the System Setup.
To use STIM(980) for high-precision time measurement, the pulse output operation mode must be set to *Calculation (time measurement)* in advance in the System Setup.

Operation

1. Starting a One-shot Timer

   Set C1 to #0000, set C2 to a constant or the word containing the timer SV, and set C3 to the interrupt task number.

2. Outputting a One-shot Pulse Output

   Set C1 to #0001 or #0002, set C2 to the ON time, and set C3 to the interrupt task number.

3. Starting a Scheduled Interrupt Timer

   Set C1 to #0003, set C2 to a constant or the word containing the timer SV, and set C3 to the interrupt task number.

4. Reading the Timer PV

   Set C1 to #0006 and set C2 and C3 to the words that will receive the Timer PV parameters.

5. Starting/Stopping Pulse Counter Time Measurement

   Set C1 to #000B or #000C, set C2 to #0000 (start) or #0001 (stop), and set C3 to the time units.

6. Stopping the Timer (One-shot or Scheduled Interrupt Timer Only)

   Set C1 to #000A and set C2 and C3 to #0000.

Flags

Name	Label	Operation
Error Flag	ER	ON if a value that is out of range is specified for an SV. (For example, an error will occur if C1 is set to a value other than 0000, 0001, 0002, 0003, 0006, 000A, 000B, or 000C.) ON if the pulse output operation mode set in the System Setup does not match the control mode specified in C1.
Equals Flag	=	---
Negative Flag	N	---

System Setup Settings

Tab	Name	Setting
Pulse Output Tab	Pulse output operation mode	1 shot

Precautions

Read Timer PV and Stop Timer

- The Read Timer PV function (C1 = #0006) and Stop Timer function (C1 = #000A) are valid only on a one-shot timer or scheduled interrupt timer started with STIM(980).

  The Read Timer and Stop Timer functions cannot be used on the one-shot pulse output or pulse counter time measurement.

One-shot Pulse Outputs

- When STIM(980) is executed just once to start a one-shot pulse output (C1 = #0001 or #0002), a pulse output with the specified pulse width is started from the corresponding pulse output port.

  In general, start the pulse output with either the up-differentiated version of the instruction (@ prefix) or an execution condition that is ON for just one cycle. The later STIM(890) instruction will be ignored if a one-shot pulse output is being generated and another instruction is executed to start a one-shot pulse output.

  The Pulse Output Flag (A874.07 or A875.07) will be ON while a one-shot pulse is being output by STIM(890).

- The pulse output operation mode must be set to *1 shot* in the System Setup in order to use STIM(980) to control one-shot pulse outputs. If STIM(980) is executed to start a one-shot pulse output from a port that is set for another output mode (such as relative pulse output), the instruction will not be executed and the ER Flag will be turned ON

Pulse Counters

- When STIM(980) is executed just once to start a counter (C1 = #000B or #000C, and C2 = #0000), the pulse count is started and must be stopped by executing the same instruction with C2 = #0001.

In general, start the pulse counter with either the up-differentiated version of the instruction (@ prefix) or an execution condition that is ON for just one cycle. If another STIM(890) instruction is executed to start the pulse counter while pulses are already being counted, the pulse counter will restart.

- The pulse output operation mode must be set to *Calculation (time measurement)* in the System Setup in order to use STIM(980) to count pulses. If STIM(980) is executed to start a pulse counter at a port that is set for another output mode (such as relative pulse output), the instruction will not be executed and the ER Flag will be turned ON.

3-20 High-speed Counter/Pulse Output Instructions

This section describes instructions used to control the high-speed counters and pulse outputs.

Instruction	Mnemonic	Function code	Page
MODE CONTROL	INI	880	521
HIGH-SPEED COUNTER PV READ	PRV	881	527
REGISTER COMPARISON TABLE	CTBL	882	530
SPEED OUTPUT	SPED	885	537
SET PULSES	PULS	886	543
PULSE OUTPUT	PLS2	887	550
ACCELERATION CONTROL	ACC	888	555

3-20-1 MODE CONTROL: INI(880)

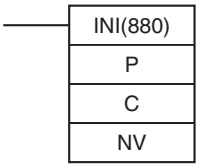
Purpose

INI(880) can be used to execute the following operations:

- To start comparison with the high-speed counter or pulse output counter comparison table
- To stop comparison with the high-speed counter or pulse output counter comparison table
- To change the PV of the high-speed counter
- To change the maximum circular value of the high-speed counter or pulse output counter
- To change the PV of the pulse output
- To stop pulse output
- To stop comparison of the sampling counter
- To change the PV of the sampling counter
- To change the circular value of the sampling counter

This instruction is supported by the FQM1-MMP22 and FQM1-MMA22 Motion Control Modules only.

Ladder Symbol



P: Port specifier

C: Control data

NV: First word with new PV

Variations

Variations		
	Executed Each Cycle for ON Condition	INI(880)
	Executed Once for Upward Differentiation	@INI(880)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

P: Port Specifier

P specifies the port to which the operation applies.

P	FQM1-MMP22	FQM1-MMA22
#0001	Counter input port 1	Counter input port 1
#0002	Counter input port 2	Counter input port 2
#0003	Pulse output port 1	Sampling counter
#0004	Pulse output port 2	---

C: Control Data

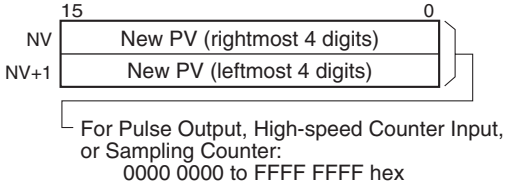
The function of INI(880) is determined by the control data, C.

C	INI(880) function
#0000	Start comparison. (Use only in target value comparison mode.)
#0001	Stop comparison. (Use only in target value comparison mode.)
#0002	Change pulse output PV, high-speed counter PV, or sampling counter PV.
#0003	Stop pulse output. (Use only with the FQM1-MMP22.)
#0004	Change pulse output counter circular value, high-speed counter circular value, or sampling counter circular value.

NV: First Word with New PV

NV and NV+1 contain the new PV when changing the PV (when C = #0002).

If the PV is not being changed, this operand is not used and NV should be set to #0000.



Operand Specifications

Area	P	C	NV	
			C = #0002 or #0004	C = Other value
CIO Area	---	---	CIO 0000 to CIO 6142	CIO 0000 (Cannot be used.)
Work Area	---	---	W000 to W254	---
Auxiliary Bit Area	---	---	A000 to A958	---
Timer Area	---	---	T0000 to T0254	---
Counter Area	---	---	C0000 to C0254	---
DM Area	---	---	D00000 to D32766	---
Indirect DM addresses in binary	---	---	@ D00000 to @ D32767	---
Indirect DM addresses in BCD	---	---	*D00000 to *D32767	---
Constants	See operand description.	See operand description.	---	---
Data Resisters	---	---	---	---

Area	P	C	NV	
			C = #0002 or #0004	C = Other value
Index Registers	---	---	---	---
Indirect addressing using Index Registers	---	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	---

Description

INI(880) performs the operation specified in C for the port specified in P. The possible combinations of operations and ports are shown in the following table.

P: Port specifier	C: Control data				
	0000 hex: Start comparison	0001 hex: Stop comparison	0002 hex: Change PV	0003 hex: Stop pulse output	0004 hex: Change circular value
#0001 or #0002: High-speed counter input	OK	OK	OK	Not allowed.	OK
#0003 or #0004: Pulse output	OK	OK	OK	OK	OK
#0003 Sampling counter	Not allowed.	OK	OK	Not allowed.	OK

■ **Starting Comparison (C = 0000 hex)**

If C is 0000 hex, INI(880) starts comparison of a high-speed counter's PV or pulse output counter's PV to the comparison table registered with CTBL(882).

Note If INI(880) is executed without registering a table, the Error Flag will turn ON and INI(880) will not be executed.

■ **Stopping Comparison (C = 0001 hex)**

If C is 0001 hex, INI(880) stops comparison of a high-speed counter's PV or pulse output counter's PV to the comparison table registered with CTBL(882).

■ **Changing a PV (C = 0002 hex)**

High-speed Counter Input (P = #0001, #0002, or #0003)

P = #0001 or #0002: Change high-speed counter PV

P = #0003 (FQM1-MMA22): Change sampling counter PV

High-speed counter's counting mode		Control function	PV setting range
Linear mode	Differential inputs, increment/decrement pulses, or pulse + direction inputs	Changes high-speed counter PV. The new value is specified in NV and NV+1.	8000 0000 to 7FFF FFFF hex (-2,147,483,648 to 2,147,483,647)
Circular mode or sampling counter operation		**Note:** An error will occur if the specified port is not set for high-speed counter operation.	0000 0000 to circular set value (hex)

Pulse Output (P = #0003 or #0004)

High-speed counter's counting mode	Control function	PV setting range
Absolute pulse output linear mode	Changes the pulse output PV. The new value is specified in NV and NV+1. **Note:** This instruction can be executed only while the pulse output is stopped. An error will occur if it is executed while pulses are being output.	8000 0000 to 7FFF FFFF hex (-2,147,483,648 to 2,147,483,647)
Absolute pulse output circular mode		0000 0000 to circular set value (hex)

■ **Stopping the Pulse Output (P = #0003 or #0004, C = #0003)**

This function immediately stops the pulse output from the specified port. If this instruction is executed when pulse output is already stopped, then the pulse amount setting will be cleared. When pulse output is stopped, the Pulse Output Direction Flags (A874.08 and A875.08) will be turned OFF.

■ **Changing the Counter Circular Value (C = #0004)**

High-speed Counter Input (P = #0001, #0002, or #0003)

P = #0001 or #0002: Change high-speed counter PV

P = #0003 (FQM1-MMA22): Change sampling counter PV

High-speed counter's counting mode	Control function	PV setting range
Circular mode	Changes the high-speed counter's circular value. The new value is specified in NV and NV+1. **Note:** An error will occur if the specified port is not set for high-speed counter operation.	0000 0000 to FFFF FFFF hex (0 to 4,294,967,295)
Sampling counter operation		0000 0000 to FFFF FFFF hex (0 to 4,294,967,295)

Pulse Output (P = #0003 or #0004)

Control function	PV setting range
Changes the pulse output counter's circular value. The new value is specified in NV and NV+1.	0000 0000 to FFFF FFFF hex (0 to 4,294,967,295)

Maximum Circular Value in System Setup

INI(880) does not change the maximum circular value setting in the System Setup. This instruction changes the counter's register directly, so the counter's circular value is initialized to the System Setup circular value when the power is turned ON. The high-speed counter or pulse output counter circular value can be changed while the counter is counting, so check the status of the counter before changing the circular value. If the circular value is set below the PV, the new circular value won't be effective until the counter has counted all the way to FFFF FFFF and started counting again from 0.

Execution Conditions for each Function

1. Starting Comparison between the Target Value Comparison Table and the High-speed Counter PV

   To start the comparison, execute INI(880) with P set to #0001 or #0002, C set to #0000, and NV set to 0000. Since the comparison operation will continue after INI(880) is executed one time with C = #0000, start the comparison with either the up-differentiated version of the instruction (@ prefix) or an execution condition that is ON for just one cycle.

2. Stopping Comparison between the Target Value Comparison Table and the High-speed Counter PV

To stop the comparison, execute INI(880) with P set to #0001 or #0002, C set to #0001, and NV set to 0000.

3. Starting Comparison between the Target Value Comparison Table and the Pulse Output PV

   To start the comparison, execute INI(880) with P set to #0003 or #0004, C set to #0000, and NV set to 0000. Since the comparison operation will continue after INI(880) is executed one time with C = #0000, start the comparison with either the up-differentiated version of the instruction (@ prefix) or an execution condition that is ON for just one cycle.

   Note If a target value comparison is executed for the pulse output PV under certain conditions, pulse outputs will stop at completion of the comparison with no mismatch even if the specified pulse width or target position has not been reached. Do not use the target value comparison for pulse output PV under the following conditions:

   (1) When the pulse output operation mode is either *1 shot* or *Electronic cam control*.

   (2) When the pulse output operation mode is either *Relative pulse* or *Absolute pulse* for independent-mode positioning.

4. Stopping Comparison between the Target Value Comparison Table and the Pulse Output PV

   To stop the comparison, execute INI(880) with P set to #0003 or #0004, C set to #0001, and NV set to 0000.

5. Changing the High-speed Counter PV

   To change the high-speed counter's PV, execute INI(880) with P set to #0001 or #0002, C set to #0002, and the new PV set in NV and NV+1. If the counter is operating in circular mode, the PV must not be higher than the maximum circular counter value. If the specified PV is higher than the maximum circular counter value, an error will occur and the PV will not be changed.

6. Changing the Pulse Output PV

   To change the pulse output PV, execute INI(880) with P set to #0003 or #0004, C set to #0002, and the new PV set in NV and NV+1.

   Note a) INI(880) can be executed to change the pulse output PV when the pulse output mode is set to relative pulse output, one-shot pulse output, or pulse counter time measurement. However, these pulse output modes initialize the operation counter to 0 before starting operation, so changing the pulse output PV has no real effect.

   b) In circular mode, the pulse output PV cannot be changed to a value higher than the maximum circular counter value.

7. Stopping the Pulse Output

   To stop the pulse output, execute INI(880) with P set to #0003 or #0004, C set to #0003, and NV set to 0000.

   Note Set the System Setup's pulse output mode to relative pulse output, absolute pulse output (linear mode), or absolute pulse output (circular mode). An error will occur and INI(880) won't be executed if the pulse output mode is set to one-shot mode or pulse counter time measurement mode.

8. Changing the High-speed Counter Circular Value

   To change the high-speed counter's circular value, execute INI(880) with P set to #0001 or #0002, C set to #0004, and the new circular value set in NV and NV+1.

Note a) This function does not change the maximum circular value setting in the System Setup; it changes the counter's register directly.

b) The circular value can be changed while the counter is counting. Check the status of the counter before changing the circular value.

9. Changing the Pulse Output Counter Circular Value

   To change the pulse output counter's circular value, execute INI(880) with P set to #0003 or #0004, C set to #0004, and the new circular value set in NV and NV+1.

   Note a) This function does not change the maximum circular value setting in the System Setup; it changes the counter's register directly.

   b) The circular value can be changed while the counter is counting. Check the status of the counter before changing the circular value.

10. Stopping (Interrupting) Sampling by the Sampling Counter

    To stop sampling, execute INI(880) with P set to #0003, C set to #0001, and NV set to 0000.

11. Changing the Sampling Counter PV

    To change the sampling counter's PV, execute INI(880) with P set to #0003, C set to #0002, and the new PV set in NV and NV+1. The PV can not be higher than the maximum circular counter value.

    This function can be used to clear or adjust the sampling counter.

12. Changing the Sampling Counter Circular Value

    To change the sampling counter's circular value, execute INI(880) with P set to #0003, C set to #0004, and the new circular value set in NV and NV+1.

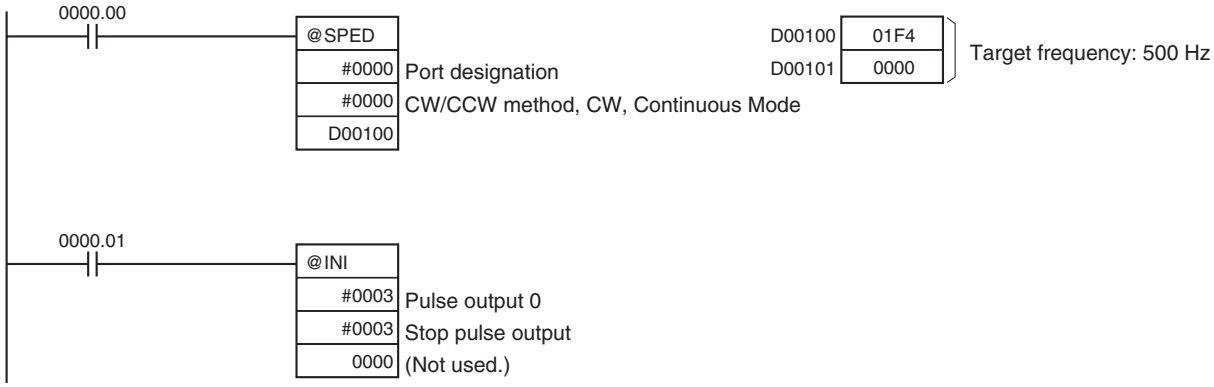
    This function can be used to set the sampling period as a circular value.

Flags

Name	Label	Operation
Error Flag	ER	ON if P is set to a value other than #0001, #0002, #0003, or #0004. (With the FQM1-MMA22, #0004 is not allowed either.) ON if the PV in NV and NV+1 exceeds the maximum circular counter value during relative pulse output circular mode or high-speed counter circular mode operation. ON if a new PV is specified for a port that is currently outputting pulses. ON if INI(880) is executed to stop pulses while the System Setup is set for one-shot pulse output. ON if an instruction controlling pulse I/O or a high-speed counter is being executed in the main program, an interrupt occurs, and INI(880) is executed in the interrupt task. OFF in all other cases.

Example

When CIO 0000.00 goes from OFF to ON in the following example, SPED(885) starts outputting pulses from pulse output 0 in Continuous Mode at 500 Hz. When CIO 0000.01 goes from OFF to ON, INI(880) stops the pulse output.

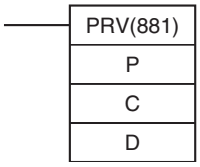


3-20-2 HIGH-SPEED COUNTER PV READ: PRV(881)

Purpose

Use PRV(881) to read the high-speed counter PV, high-speed counter latch value, high-speed counter rate-of-change (counter movements) or frequency, pulse output PV, pulse counter PV (time measurement), or elapsed time of one-shot pulse output.

Ladder Symbol



P: Port specifier
C: Control data
D: First destination word

Variations

Variations		
	Executed Each Cycle for ON Condition	PRV(881)
	Executed Once for Upward Differentiation	@PRV(881)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

P: Port Specifier

P specifies the port to which the operation applies.

P	Port
#1	High-speed counter 1
#2	High-speed counter 2
#3	Pulse output 1 or Analog input (FQM1-MMA22 only)
#4	Pulse output 2

C: Control Data

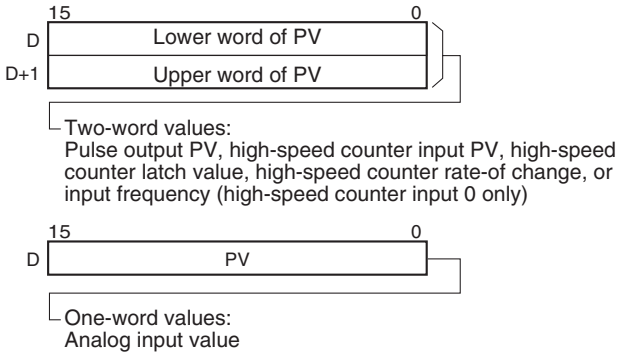
The function of INI(880) is determined by the control data, C.

C	Function
#0	Reads one of the following values: • High-speed counter PV • Pulse output PV • Pulse output counter PV (time measurement) • Analog input value

C	Function
#1	Reads the high-speed counter rate-of-change or measured frequency.
#2	Reads the high-speed counter latch value.

D: First Destination Word

The PV is output to D or to D and D+1.



Operand Specifications

Area	P	C	D
CIO Area	---	---	CIO 0000 to CIO 6142
Work Area	---	---	W000 to W254
Auxiliary Bit Area	---	---	A448 to A958
Timer Area	---	---	T0000 to T0254
Counter Area	---	---	C0000 to C0254
DM Area	---	---	D00000 to D32766
Indirect DM addresses in binary	---	---	@ D00000 to @ D32767
Indirect DM addresses in BCD	---	---	*D00000 to *D32767
Constants	See operand description.	See operand description.	---
Data Resisters	---	---	---
Index Registers	---	---	---
Indirect addressing using Index Registers	---	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

The function performed by PRV(881) depends on the setting of operand C (the control data word).

■ **Reading a PV (C = #0000)**

Port and mode		Operation	Setting range
Absolute pulse output	Linear Mode	Reads the pulse output PV and stores it in D and D+1.	8000 0000 to 7FFF FFFF hex (-2,147,483,648 to 2,147,483,647)
	Circular Mode		0000 0000 to FFFF FFFF hex (0 to 4,294,967,295)
Relative pulse output			
High-speed counter input	Linear Mode	Reads the high-speed counter PV and stores it in D and D+1.	8000 0000 to 7FFF FFFF hex (-2,147,483,648 to 2,147,483,647)
	Circular Mode		0000 0000 to FFFF FFFF hex (0 to 4,294,967,295)

■ **Reading High-speed Counter Rate-of-change or Frequency (C = #0001)**

Reads the high-speed counter's rate-of-change (counter movements) or measured frequency value and stores it in D and D+1.

D: Rightmost 4 digits (hexadecimal or BCD)
D+1: Leftmost 4 digits (hexadecimal or BCD)

The possible range of the high-speed counter's rate-of-change (counter movements) or measured frequency value is 0000 0000 to FFFF FFFF hex.

Note When reading the high-speed counter rate-of-change (counter movements) with the sampling time set to the cycle time, use an instruction such as MOV(021) to read the value directly from A854 and A855 (counter 1) or A856 and A857 (counter 2). If PRV(881) is used to read the rate-of-change value, a value of 0 will be output.

■ **Reading the Analog Input Value (C = #0001, FQM1-MMA22 Only)**

Reads the most recent analog input value and stores it in D.

The Motion Control Module converts the analog input data when PRV(881) is executed, so the most recent value is stored. The analog input method must be set to *Immediate refresh* in the System Setup in order to read the analog input value with PRV(881).

■ **Reading High-speed Counter Latch Value (C = #0002)**

Reads the high-speed counter's PV and stores it in D and D+1 as an 8-digit hexadecimal value.

D: Rightmost 4 digits (hexadecimal)
D+1: Leftmost 4 digits (hexadecimal)

Range in linear mode: 8000 0000 to 7FFF FFFF hex.
Range in circular mode: 0000 0000 to FFFF FFFF hex

Execution Conditions

Reading the High-speed counter PV, High-speed Counter Latch Value, Pulse Output PV, Pulse Counter PV (Time measurement), Elapsed Time of the One-shot Pulse Output, or Analog Input Value

To read the desired value, execute PRV(881) with P set to #0001, #0002, #0003, or #0004, C set to #0000, and D set to the first word address where the value will be stored.

Note

1. The high-speed counter PVs read with PRV(881) are the same as the values stored in A850 and A851 (port 1 PV) and A852 and A853 (port 2 PV), but those Auxiliary Area words are refreshed just once each cycle whereas the value read with PRV(881) always provides the most recent data.

2. The pulse output PVs and pulse output counter PV (time measurement PVs) read with PRV(881) are the same as the values stored in A870 and A871 (port 1 PV) and A872 and A873 (port 2 PV), but those Auxiliary Area words are refreshed just once each cycle whereas the value read with PRV(881) always provides the most recent data.

Flags

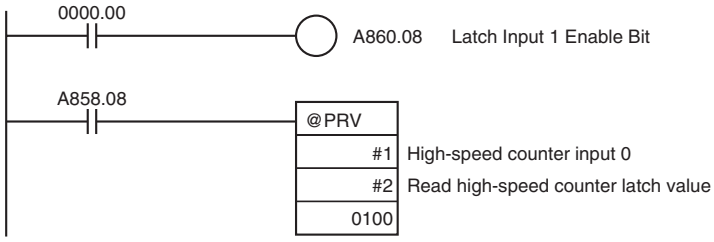
Name	Label	Operation
Error Flag	ER	ON if the combination of P and C is not allowed. (For example, ON if P = #0003 and C = #0001.) ON if P is set to a value other than #0001, #0002, #0003, or #0004. ON if C is set to a value other than #0000 or #0001. ON if an instruction controlling pulse I/O or a high-speed counter is being executed in the main program, an interrupt occurs, and PRV(881) is executed in the interrupt task. OFF in all other cases.

Examples

■ **Example 1**

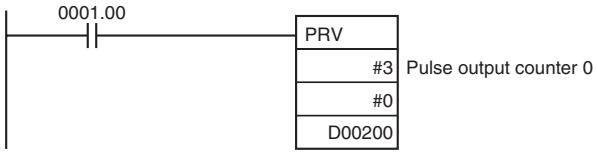
When CIO 0000.00 goes from OFF to ON in the following programming example, the latch input is enabled and the high-speed counter PV is latched if the latch input is ON.

When the Count Latched Flag goes from OFF to ON, PRV(881) reads the latched high-speed counter 0 PV and stores it in words CIO 0101 and CIO 0100.



■ **Example 2**

When CIO 0001.00 goes from OFF to ON in the following programming example, PRV(881) reads the most recent pulse output counter PV and stores it as a hexadecimal value in D00200 and D00201.



3-20-3 REGISTER COMPARISON TABLE: CTBL(882)

Purpose

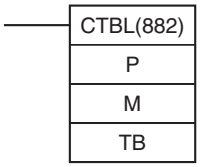
Use CTBL(882) to register a comparison table and compare the table values with a high-speed counter PV or pulse output counter PV. Either target value or range comparisons are possible. An interrupt task is executed when a specified condition is met.

When performing range comparisons, a bit pattern is output internally when the PV is within a specified range.

It is also possible to register a comparison table without starting the comparison. Once the table is registered, the comparison operation can be started/stopped with INI(880).

When performing high-speed analog sampling, CTBL(882) starts the sampling operation.

Ladder Symbol



P: Port specifier
M: Mode specifier
TB: First comparison table word

Variations

Variations		
	Executed Each Cycle for ON Condition	CTBL(882)
	Executed Once for Upward Differentiation	@CTBL(882)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

P: Port Specifier
P specifies the port for which pulses are to be counted as shown in the following table.

P	Port
#1	High-speed counter 1
#2	High-speed counter 2
#3	Pulse output 1 or Sampling counter
#4	Pulse output 2

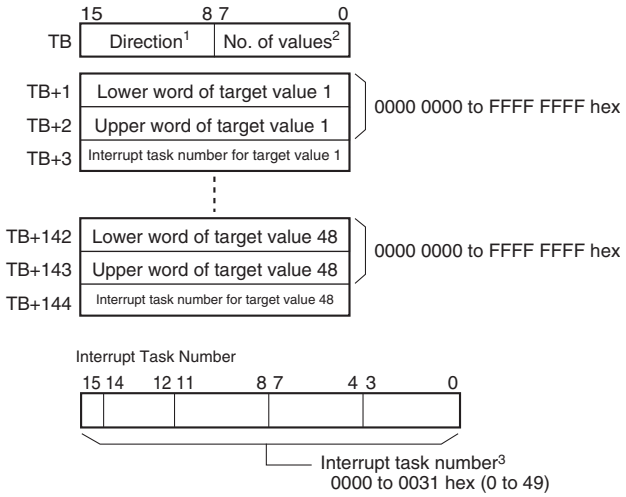
M: Mode Specifier
The function of CTBL(882) is determined by the mode specifier, M, as shown in the following table.

M	CTBL(882) function
#0	For a counter or pulse output, this mode registers a target value comparison table and starts comparison. For a sampling counter, this mode registers a target value comparison table, clears the sampling counter to 0, and starts sampling.
#1	For a counter or pulse output, this mode starts range comparison. For a sampling counter, this mode starts sampling without clearing the sampling counter to 0.
#2	Registers a target value comparison table.

TB: First Comparison Table Word
TB is the first word of the comparison table. The structure of the comparison table depends on the type of comparison being performed.

- Target Value Comparison

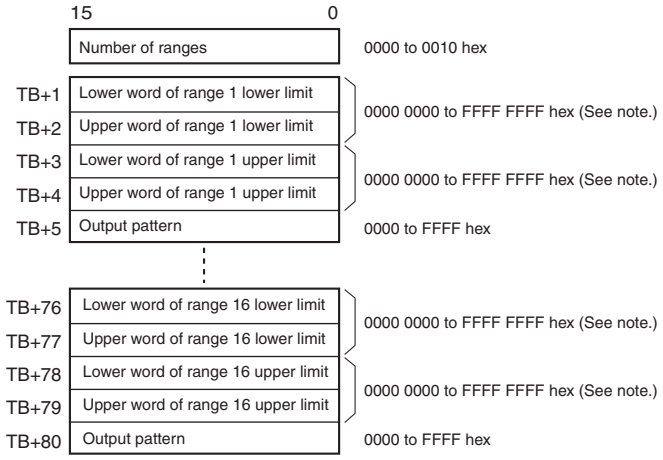
  For target value comparison, the length of the comparison table is determined by the number of target values specified in TB. The table can be between 4 and 145 words long, as shown below.



Note 1. Set 00 for incrementing or F0 for decrementing.
2. Setting range: 01 to 30 hex (1 to 48 target values)
3. Set the interrupt task number to FFFF hex to disable comparison.

- Range Comparison

  For range comparison, the length of the comparison table is determined by the number of ranges specified in TB. The table can be between 6 and 81 words long, as shown below.



Operand Specifications

Area	P	M	TB
CIO Area	---	---	CIO 0000 to CIO 6140
Work Area	---	---	W000 to W252
Auxiliary Bit Area	---	---	A448 to A956
Timer Area	---	---	T0000 to T0252
Counter Area	---	---	C0000 to C0252
DM Area	---	---	D00000 to D32764
Indirect DM addresses in binary	---	---	@ D00000 to @ D32767
Indirect DM addresses in BCD	---	---	*D00000 to *D32767
Constants	See operand description.	See operand description.	---
Data Resisters	---	---	---

Area	P	M	TB
Index Registers	---	---	---
Indirect addressing using Index Registers	---	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

CTBL(882) registers a comparison table and starts comparison (with a high-speed counter PV or pulse output counter PV) for the port specified in P and the method specified in M. Once a comparison table is registered, it is valid until a different table is registered or until the Motion Control Module is switched to PROGRAM mode.

Each time CTBL(882) is executed, comparison is started under the specified conditions. When using CTBL(882) to start comparison, it is normally sufficient to use the differentiated variation (@CTBL(882)) of the instruction or an execution condition that is turned ON only for one cycle.

Note If the comparison table specifies an interrupt task number that is not in the program, a program error (fatal error) will occur when that interrupt is called.

■ Registering a Target Value Comparison Table (M = #0002)

If M is set to #0002, CTBL(882) registers a comparison table to compare the high-speed counter PV or pulse output counter PV, but does not start comparison. In this case, start comparison separately with INI(880).

■ Registering a Target Value Comparison Table and Starting Comparison (M = #0000)

If M is set to #0000, CTBL(882) registers a comparison table to compare the high-speed counter PV or pulse output counter PV, and starts the comparison.

■ Stopping Comparison

Use INI(880) to stop comparison operations started with either CTBL(882) or INI(880).

■ Target Value Comparison Operation

Target value comparison compares the PV with a list of preset target values. Up to 48 target values can be registered in the table and the target values are compared in the order in which they appear in the table. When the PV matches a target value, CTBL(882) performs the following operations and then moves to the next target value in the table.

Execution of the Interrupt Task (FQM1-MMP22 and FQM1-MMA22)

The first comparison target value is determined by the direction setting (in the leftmost byte of TB) and the high-speed counter's PV, as follows.

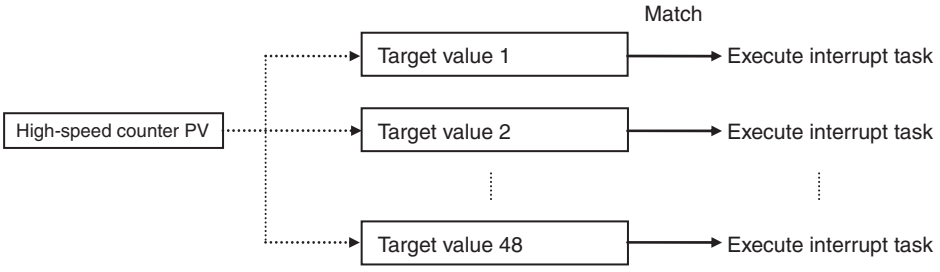
- Direction = 00 hex: Incrementing

  Compares target values with the PV from the beginning of the table. The first target value in the table that is greater than the PV is used as the first comparison target value.

- Direction = F0 hex: Decrementing

  Compares target values with the PV from the beginning of the table. The first target value in the table that is less than the PV is used as the first comparison target value.

If the direction setting is inappropriate (incrementing specified but all target values are less than the PV or decrementing specified but all target values are greater than the PV), the first target value registered in the table will be used as the first comparison target value.



Target Value Comparison Table Structure (FQM1-MMP22 and FQM1-MMA22)

Word	Leftmost byte (bits 08 to 15)	Rightmost byte (bits 08 to 15)	Function
TB	Direction (See note 1.)	Number of target values (See note 2.)	Table definition
TB+1	Target value (rightmost 4 digits, hexadecimal)		One target value entry (Set the number of entries specified in TB.)
TB+2	Target value (leftmost 4 digits, hexadecimal)		
TB+3	Interrupt task number (0000 to 0031, hexadecimal)		

Note

1. Set the direction to 00 hex when incrementing or F0 hex when decrementing.
2. Set the number of target value entries between 01 and 30 hex (1 to 48).

Set interrupt task numbers between 0000 and 0031 hex. The same interrupt task number can be used for multiple target value entries.

When interrupt processing is not required, set the interrupt task number to FFFF hex to disable interrupt processing for that target value entry.

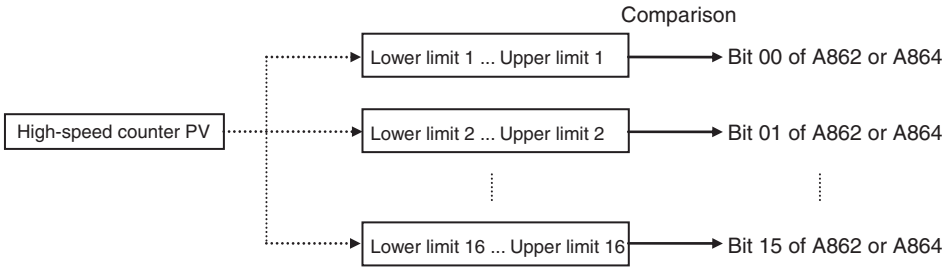
The comparison operation can be stopped with INI(880). Once a comparison table is registered, it is valid until a different table is registered or until the Motion Control Module stops operating.

■ Range Comparison Operation (M = #0001)

Range comparison compares the PV with target ranges defined by a lower limit and an upper limit. Up to 16 ranges (lower and upper limit pairs) can be set in the table.

The comparison operation is performed each time CTBL(882) is executed, and all of the entries in the comparison table are evaluated starting from the beginning of the table. The comparison results are output as flags (bits 0 to 15 correspond to ranges 1 to 16) with the corresponding bit ON when the comparison result is true. The flags are output to the Auxiliary Area word allocated to the port (words A862 to A865 for the high-speed counters and A880 to A883 for the pulse outputs).

At the same time, the specified output bit pattern is stored in the allocated word (A863 or A865 for the high-speed counters and A881 or A883 for the pulse outputs). When several comparison conditions are met at the same time, the output patterns are all ORed and the OR result is stored as the result.



High-speed Counter 1: Range comparison result = A862
Output pattern destination = A863
High-speed Counter 2: Range comparison result = A864
Output pattern destination = A865
Pulse Output 1: Range comparison result = A880
Output pattern destination = A881
Pulse Output 2: Range comparison result = A882
Output pattern destination = A883

Range Comparison Table Structure

Word	Content	Function
TB	Number of ranges in table (0001 to 0010 hex)	Table definition
TB+1	Lower limit 1 (rightmost 4 digits, hexadecimal)	One range entry (Set the number of range entries specified in TB.)
TB+2	Lower limit 1 (leftmost 4 digits, hexadecimal)	
TB+3	Upper limit 1 (rightmost 4 digits, hexadecimal)	
TB+4	Upper limit 1 (leftmost 4 digits, hexadecimal)	
TB+5	Output pattern	

- Overlapping ranges can be specified.
- If the lower and upper limit values are reversed in linear mode, the comparison operation will not function properly, but an error will not occur. In linear mode, always set the upper limit ≥ lower limit.

■ High-speed Analog Sampling (M = #0000, FQM1-MMA22 Only)

This function stores the analog input data in the specified DM words of an FQM1-MMA22 Motion Control Module.

The high-speed analog sampling function stores the analog input value in the specified DM Area location (starting at TB+2) each time that the sampling counter PV matches the target value specified in TB and TB+1. Sampling stops automatically when the desired number of analog input samples (specified in TB+3) have been stored.

In order to use the high-speed analog sampling function, the input method must be set to *Immediate refresh* in the System Setup's **Analog Input/Output** Tab.

Table Structure

The following table shows the function of the table words

Word	Content
TB	Target value (rightmost 4 digits, hexadecimal)
TB+1	Target value (leftmost 4 digits, hexadecimal)
TB+2	First DM word for storage of analog input sample (Set a DM Area offset address between 0000 and 7FFF hex.)
TB+3	Number of samples (Specifies the number of samples to store. Set between 0001 and 8000 hex.)

Name	Label	Operation
Error Flag	ER	ON if P is set to a value other than #0001, #0002, #0003, or #0004. ON if M is set to a value other than #0000, #0001, or #0002. ON if one of the following errors is detected in a target value comparison table. • ON if TB is not 01 to 30 hex. • In circular mode, ON if the content of TB+1 and TB+2 (target value) exceeds the maximum circular counter value. • ON if TB+3 (interrupt task number) contains a value other than #0000 to #0031 or #FFFF. ON if all of the interrupt task numbers in the table are set to #FFFF (disabled). ON if TB is not #0001 to #0010 hex in a range comparison table. ON if an instruction controlling a high-speed counter is being executed in the main program, an interrupt occurs, and CTBL(882) is executed in the interrupt task. OFF in all other cases.

Precautions

Do not change the maximum circular counter value when performing a comparison in circular mode.

When using target value comparisons, set target values that allow an interval greater than the "interrupt overhead time + interrupt task processing time" after the interrupt is generated.

Do not perform target value comparisons on the pulse output counter when using one of the following operation modes for pulse outputs. The pulse output will not operate properly if target value comparisons are performed.

- Independent mode (positioning)
- Electronic Cam mode
- One-shot pulse output mode

Counting will start when the count start bit goes ON or the pulse output begins, but interrupt tasks will not be called until the comparison operation is started.

Use INI(880) to stop the target value comparison.

Once a target value comparison table is registered, it is valid until a different table is registered or until the Motion Control Module stops operating. The cycle time can be reduced by executing the up-differentiated variation of CTBL(882) only when necessary.

Execution Conditions for each Function

1. Registering the Target Value Comparison Table and Starting Comparison

   To register the target value comparison table and start the comparison, execute CTBL(882) with P set to #0001, M set to #0000, and TB set to the address of the first word of the comparison table. Since the comparison operation will continue after CTBL(882) is executed one time, use either the up-differentiated variation of the instruction (@CTBL(882)) or an execution condition that is ON for just one cycle.

2. Starting Range Comparison

   To perform range comparison, execute CTBL(882) with P set to #0001, M set to #0001, and TB set to the address of the first word of the comparison table. When CTBL(882) is executed with these conditions, the PV is com-

pared just once to the ranges specified in the table. The comparison operation does not continue.

3. Registering the Target Value Comparison Table Only

   To just register the target value comparison table, execute CTBL(882) with P set to #0001, M set to #0002, and TB set to the address of the first word of the comparison table.

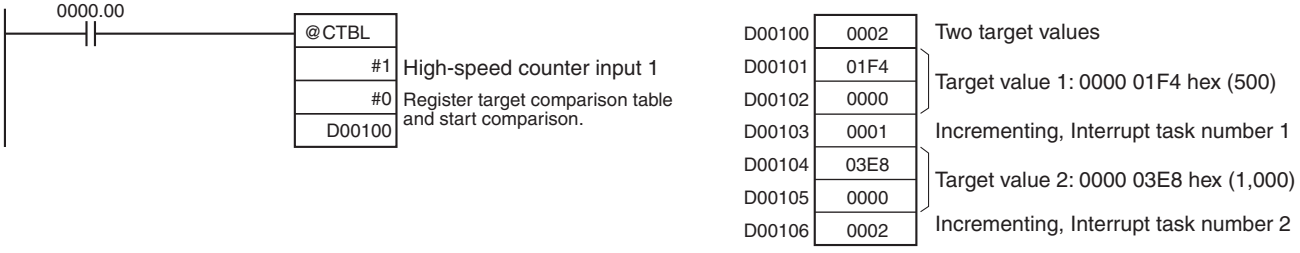
4. Starting High-speed Analog Sampling

   To start high-speed analog sampling, execute CTBL(882) with P set to #0003, M set to #0000 or #0001, and TB set to the address of the first word of the comparison settings table. Once CTBL(882) is executed, the comparison operation will continue until the FQM1-MMA22 Motion Control Module has stored the number of samples specified in TB+3. Since CTBL(882) needs to be executed just one time, use either the up-differentiated variation of the instruction (@CTBL(882)) or an execution condition that is ON for just one cycle.

   If M is set to #0000, the sampling counter will be cleared to 0 when sampling starts. If M is set to #0001, the sampling counter will not be cleared to 0.

Example

When CIO 0000.00 goes from OFF to ON in the following programming example, CTBL(882) registers a target value comparison table and starts comparison for high-speed counter 1. The PV of the high-speed counter is counted incrementally and when it reaches 500, it equals target value 1 and interrupt task 1 is executed. When the PV is incremented to 1,000, it equals target value 2 and interrupt task 2 is executed.



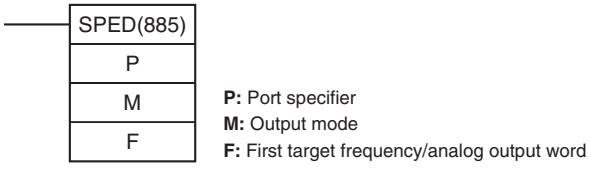
3-20-4 SPEED OUTPUT: SPED(885)

Purpose

SPED(885) is used to set the output pulse frequency for a specific port and start pulse output without acceleration or deceleration. Either independent mode positioning or continuous mode speed control is possible. For independent mode positioning, the number of pulses is set using PULS(886).

SPED(885) can also be executed during pulse output to change the target frequency of the current pulse outputs, creating stepped speed changes. This instruction is supported by the FQM1-MMP22 and FQM1-MMA22 only.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	SPED(885)
	Executed Once for Upward Differentiation	@SPED(885)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

P: Port Specifier

The port specifier specifies the port where the pulses will be output.

P	Port
#1	FQM1-MMP22: Pulse output 1 FQM1-MMA22: Analog output 1
#2	FQM1-MMP22: Pulse output 2 FQM1-MMA22: Analog output 2

M: Output Mode

The value of M determines the output mode.

Output	Value of M	Mode
Pulse output	#0000	Continuous, CW
	#0001	Continuous, CCW
	#0002	Independent, CW
	#0003	Independent, CCW
Analog output	#0000 (fixed)	---

F: First Frequency Word or First Analog Output Word

- Pulse Outputs (F = First Frequency Word)

  The allowed frequency setting range is slightly different when F is specified as a constant or word address.

Setting method	Setting
Constant	Specify the frequency in 1-Hz units. Range: #0000 0000 to #000F 4240 (0 to 1 MHz)
Word address	Specify the frequency in 1-Hz units. Range: #0000 0000 or #0000 0001 to #000F 4240 (1 to 1 MHz) F: Rightmost 4 digits F+1: Leftmost 4 digits

  The target frequency can be set between 1 Hz and 1 MHz, but the frequency that can actually be output depends on the clock frequency. Refer to *7-6-4 Pulse Output Specifications* in the *FQM1 Series Flexible Motion Controller Operation Manual* (Cat. No. O012) to verify the allowed output range. An error will occur and the instruction will not be executed if the specified frequency exceeds the allowed output range. If the specified frequency is below the allowed output range, the lower limit frequency will be output.

  The output frequencies are obtained by dividing the Motion Control Module's clock pulse with an integer dividing ratio, meaning the actual output frequency can be different from the set frequency. (Refer to *Precautions when Using Pulse Outputs* in the *FQM1 Series Flexible Motion Controller Operation Manual* (Cat. No. O012) for details.)

The output frequency will not be changed unless a minimum of one pulse is output. For example, if 1 Hz is output when 20 MHz (1 Hz to 1 MHz) is being used, execution will not be enabled for 1 s while the 1-pulse output is being completed. The instruction can be executed, but a 1-pulse output wait time will be required until the frequency is actually changed. For instructions with automatic acceleration/deceleration, such as PLS2(887) or ACC(888), the frequency will be changed automatically according to the acceleration/deceleration rate, but for either the start frequency or the acceleration/deceleration rate, a 1-pulse output wait time will be required. When using low frequencies, therefore, allow for delays in speed changes.

- Analog Outputs (F = First Analog Output Word)

  Sets the value that will be output from the analog output port. Specify the value in 4-digit hexadecimal.

  - –10 to +10 V Range:
    EC78 to 1388 hex (–5,000 to 5,000 decimal) (resolution: 10,000) corresponding to 0% to 100% voltage (–10 to 10 V)
    The possible setting range is actually EA84 to 157C (–5,500 to 5,500 decimal) corresponding to –5% to 105% voltage (–11 to 11 V)
  - 0 to 10 V, 0 to 5 V, and 1 to 5 V Ranges:
    0000 to 0FA0 hex (0000 to 4,000 decimal) (resolution: 4,000) corresponding to 0% to 100% of the FS range. (Actually, the setting range is FF38 to 1068 (–200 to 4,200 decimal) corresponding to –5% to 105% voltage (–0.5 to 10.5 V, –0.25 to 5.25 V, or 0.8 to 5.2 V).)

  Note An error will occur and the ER Flag will be turned ON if the settings are outside of the ranges listed above.

Operand Specifications

Area	P	M	F
CIO Area	---	---	CIO 0000 to CIO 6143
Work Area	---	---	W000 to W255
Auxiliary Bit Area	---	---	A000 to A959
Timer Area	---	---	T0000 to T0255
Counter Area	---	---	C0000 to C0255
DM Area	---	---	D00000 to D32767
Indirect DM addresses in binary	---	---	@ D00000 to @ D32767
Indirect DM addresses in BCD	---	---	*D00000 to *D32767
Constants	See operand description.	See operand description.	See operand description.
Data Resisters	---	---	---
Index Registers	---	---	---
Indirect addressing using Index Registers	---	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Execution Conditions for Pulse Output Functions (FQM1-MMP22)

1. Pulse Output in Independent Mode (Positioning) in the CW Direction: (Outputs just the Number of Pulses specified with PULS(886).)

Execute SPED(885) with P set to #0001 or #0002, M set to #0002, and F set to the target frequency.

2. Pulse Output in Independent Mode (Positioning) in the CCW Direction: (Outputs just the Number of Pulses specified with PULS(886).)

   Execute SPED(885) with P set to #0001 or #0002, M set to #0003, and F set to the target frequency.

3. Pulse Output in Continuous Mode (Speed Control) in the CW Direction

   Execute SPED(885) with P set to #0001 or #0002, M set to #0000, and F set to the target frequency.

4. Pulse Output in Continuous Mode (Speed Control) in the CCW Direction

   Execute SPED(885) with P set to #0001 or #0002, M set to #0001, and F set to the target frequency.

5. Since SPED(885) starts the pulse output with the parameters specified in operands P, M, and F, it needs to be executed just one time. Use either the up-differentiated variation of the instruction (@SPED(885)) or an execution condition that is ON for just one cycle.

6. Pulse outputs can be output independently and simultaneously from two output ports.

   Set the pulse frequency in F in 1-Hz units between 0000 0000 and 000F 4240 hex (0 Hz to 1 MHz). The pulse frequency can be set to 0 to stop the pulse output.

   The pulse output direction can be specified with operand M.

7. To generate pulse outputs with SPED(885), the pulse output operation mode must be set to *Relative pulse*, *Absolute pulse (Linear mode)*, or *Absolute pulse (Circular mode)* in advance in the System Setup. Pulses will not be output and an error will occur if the pulse output operation mode is set to *Electronic cam control*, *1 shot*, or *Calculation (time measurement)*.

8. Any of the following methods can be used to stop pulses being output by SPED(885).
   a) Execute SPED(885) again with a target frequency of 0.
   b) In independent mode (positioning), the pulse output will stop when the number of output pulses reaches the SV set with PULS(886).
   c) Execute INI(880) with C set to #0003.
   d) Switch the Motion Control Module to PROGRAM mode.
   e) Execute ACC(888) with M set to #0008 or #0009 and the target frequency set to 0 Hz. Pulse output will decelerated to a stop. (If number of output pulses set for PULS(886) in independent mode is reached, an immediate stop will be performed.)

Function Description

Pulses can be output in independent mode or continuous mode.

- Independent Mode (Positioning)
  In independent mode, only a preset number of pulses are output. Set the number of output pulses in advance with PULS(886).
- Continuous Mode (Speed Control)
  In continuous mode, pulses are output continuously until stopped by executing SPED(885) again with a target frequency of 0, executing INI(880) with C = #0003, or switching the Motion Control Module to PROGRAM mode.

In independent mode (positioning), the number of output pulses must be specified in advance with PULS(886). (No pulses will be output if the number

of output pulses is not specified before executing SPED(885).) If the pulse output has been stopped, it is necessary to set the number of output pulses again with PULS(886).

Note When pulses are being output from pulse output 1 or 2 in independent mode (positioning), the number of pulses that have been output can be monitored in the following Auxiliary Area words:
Pulse output 1: A870 (leftmost 4 digits) and A871 (rightmost 4 digits)
Pulse output 2: A872 (leftmost 4 digits) and A873 (rightmost 4 digits)

Pulses can be output from two ports simultaneously.

- When pulses are already being output by SPED(885), SPED(885) can be executed again to change the output frequency. Even though the frequency is changed, pulses are still output in independent mode (positioning), so the number of output pulses does not change.

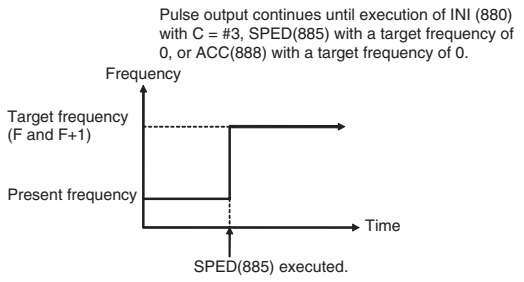
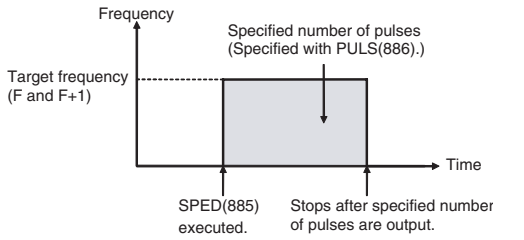
  The frequency cannot be changed if pulses are already being output and SPED(885) is executed again with a pulse output in the opposite direction. Similarly, the frequency cannot be changed if pulses are already being output in independent mode, and SPED(885) is executed again in continuous mode.

- It may not be possible to change the frequency with SPED(885) if pulses are being output by another instruction (such as a pulse output with PLS2(887).) If SPED(885) is executed to change the frequency when it cannot be changed, an error will occur.

  For details on the conditions when the frequency can be changed with SPED(885), refer to *7-6-15 Pulse Output Starting Conditions* in the *FQM1 Series Flexible Motion Controller Operation Manual* (Cat. No. O012).

- When independent mode positioning is performed with SPED(885) but the CW/CCW direction is not correct for the present position and target position, pulses will still be output in the specified direction. When *Absolute pulse (Linear mode)* operation is being used in this case, the target position will never be reached within the 8000 0000 to 7FFF FFFF range. With the pulse output counter, outputs will continue without producing an overflow or underflow even when the counter exceeds the range above. The target value will be reached after exceeding the range.

- When pulse output is started, the direction of rotation (CW or CCW) will be reflected in the Rotation Direction Flags (A874.08 and A875.08).

Mode	Description	Frequency changes
Continuous (Speed control)	Mode #0 or #1 (Continuous mode) The frequency is changed from the present frequency to the target frequency in one step and the pulse output continues at the target frequency. The output will continue until stopped with INI(880) or by executing SPED(885) with a target frequency of 0.	
Independent (Positioning)	Mode #2 or #3 (Independent mode) The frequency is changed to the target frequency in one step and the pulse output continues until the number of pulses specified with PULS(886) have been output. (With independent mode, the number of pulses must be set in advance with PULS(886) and the output operates according to that setting.)	

Execution Conditions for Analog Output Functions (FQM1-MMA22)

1. Generating an Analog Output

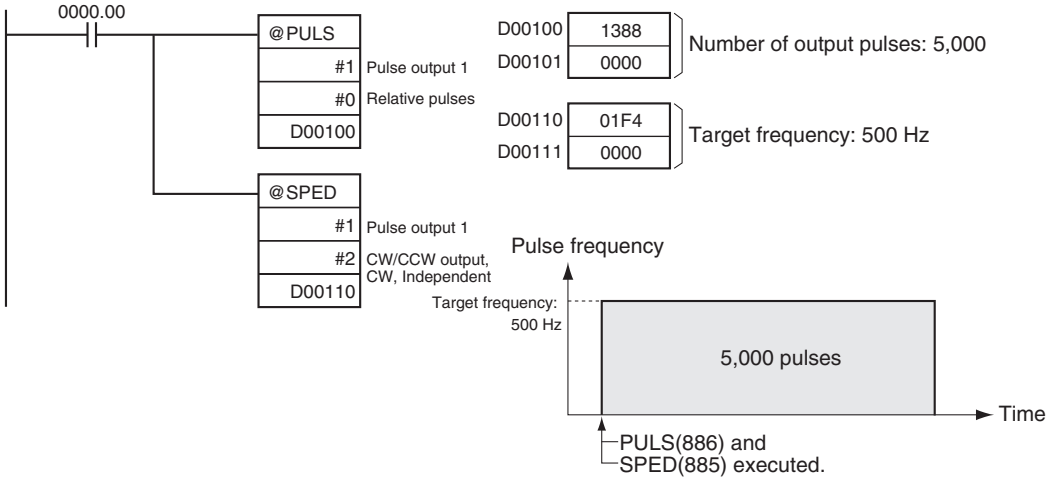
   Execute SPED(885) with P set to #0001 or #0002, M set to #0000, and F set to the desired analog output value.
2. When SPED(885) is executed, the analog output is generated according to the values specified by P, M, and F.
3. Analog outputs can be output independently and simultaneously from two output ports.
4. Analog outputs are not distinguished as continuous or independent outputs. The analog output generated by SPED(885) will maintain its output value until one of the following events occurs:
   a) SPED(885) or ACC(888) is executed with a different target value.
   b) The *Output stop function* is set to a setting other than *Hold* and the Motion Control Module is switched to PROGRAM mode or the Analog Output Conversion Enable Bit (A814.00 or A815.00) is reset to 0.
5. The present analog output value will be changed if SPED(885) is executed again while an analog output value is being output or the analog output target value has been reached after execution of ACC(888).
6. The present analog output value will not be changed and an error will occur if SPED(885) is executed by interrupting an existing SPED(885) analog output or ACC(888) has been executed to generate an analog output but the target value has not been reached.
7. In order to generate analog outputs with SPED(885) or ACC(888), the output method must be set to *Immediate refresh* in the System Setup's **Analog Input/Output** Tab. If the output method is set to *End refresh*, an error will occur and an analog output will not be generated by SPED(885) or ACC(888).

Flags

Name	Label	Operation
Error Flag	ER	ON if P is set to a value other than #0001 or #0002. For the FQM1-MMP22, ON if F is not set between #0 and #F4240. For the FQM1-MMA22, ON if F is not in range. • –10 to +10 V Range: ON if F is not between EA84 and 157C. • 0 to 10 V, 0 to 5 V, and 1 to 5 V Ranges: ON if F is not between FF38 and 1068. For the FQM1-MMP22, ON if the specified frequency is not supported at the selected Motion Control Module clock frequency. For the FQM1-MMP22, ON if SPED(885) is executed when the present pulse output cannot be changed. (For example, the pulse output cannot be changed when pulses are being output by PLS2(887) and the target value has not been reached.) ON if an instruction controlling pulse or analog I/O is being executed in the main program, an interrupt occurs, and SPED(885) is executed in the interrupt task. OFF in all other cases.

Example

When CIO 0000.00 goes from OFF to ON, PULS(886) sets the number of output pulses for pulse output 1 to a relative value of 5,000 pulses. SPED(885) is executed next to start pulse output using the CW/CCW method in the clockwise direction in independent mode at a target frequency of 500 Hz.



3-20-5 SET PULSES: PULS(886)

Purpose

PULS(886) sets the number of output pulses for independent mode pulse outputs that are started later in the program. PULS(886) can also be used to set the number of output pulses and frequency for electronic cam control.

- Independent Mode

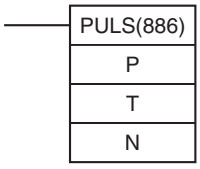
  PULS(886) sets the number of output pulses to be output in an independent mode (positioning) operation. The specified number of output pulses will be output when SPED(885) or ACC(888) is executed in independent mode.

- Electronic Cam Control

PULS(886) sets the number of output pulses and frequency, and outputs pulses. The following functions have been added to CPU Units with unit version 3.2 or later.

- In ring mode, the pulse output can be set to pass through 0. For example, if the ring value is 359, the cam can be moved from 350 past 0 to 10 using a CW pulse output.
- The pulse output frequency can be calculated automatically in both ring mode and linear mode.

Ladder Symbol



P: Port specifier
T: Pulse type
N: First number of pulses word

Variations

Variations		
	Executed Each Cycle for ON Condition	PULS(886)
	Executed Once for Upward Differentiation	@PULS(886)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

P: Port Specifier

P	Port
#1	Pulse output 1
#2	Pulse output 2

T: Pulse Type

T specifies the type of pulses that are output as follows:

T	Pulse type	Notes
#0	Relative pulse output	See note 1.
#1	Absolute pulse output	See note 2.
#2	Pulse output with absolute position specified	See note 3.
#3	Pulse output with absolute position specified (A point past zero can be specified.)	See note 4.
#4	Pulse output with relative position specified (A point past zero can be specified and the pulse output frequency can be calculated automatically.)	See note 4.

Note

1. To specify a relative pulse output, the pulse output operation mode must be set to *Relative pulse* in the System Setup.
2. To specify an absolute pulse output, the pulse output operation mode must be set to *Absolute pulse (Linear mode)* or *Absolute pulse (Circular mode)* in the System Setup.
3. To perform electronic cam control, the pulse output operation mode must be set to *Electronic cam control* in the System Setup.
4. Settings #3 and #4 can be used only in CPU Units with unit version 3.2 or later. A point past the zero point can be specified only when using ring mode, so settings #2 and #3 are identical when using linear mode.

N: First Number of Pulses Word

For a relative pulse output, specify the number of output pulses. For an absolute pulse output, specify the target position.

The number of pulses actually output depends on the pulse type, as shown below.

- Relative pulses:
  Actual number of pulses = Number of output pulses SV
- Absolute pulses:
  Actual number of pulses = Number of output pulses SV − PV

Words	Function	Setting range	Conditions
N+1, N	Number of output pulses or specified position (8-digit hexadecimal)	0000 0000 to FFFF FFFF	Specifies the number of output pulses when relative pulse output is selected.
		8000 0000 to 7FFF FFFF	Specifies the target position when absolute (linear mode) pulse output is selected.
		0000 0000 to Maximum circular counter value	Specifies the target position when absolute (circular mode) pulse output is selected.
		8000 0000 to 7FFF FFFF	Specifies the target position for a pulse output (linear mode) with absolute position specified.
		0000 0000 to 7FFF FFFF	Specifies the target position for a pulse output (ring mode) with absolute position specified, D = #2.
		8000 0000 to 7FFF FFFF (Set a ring value up to 3FFF FFFF.)	When specifying the target position for a pulse output in ring mode with an absolute position specified (D = #3 or #4), the allowed setting range is the ((ring value + 1) + target position) or (target position − (ring value + 1)).
N+3, N+2	Pulse output frequency (8-digit hexadecimal), only when T = #2 or #3	0000 0001 to 000F 4240 hex (1 Hz to 1 MHz)	
N+2	Pulse output command cycle (4-digit hex), only when D = #4.	0001 to 01F4 hex (0.1 to 50.0 ms) (Set word N+3 to 0000 hex.)	

The output frequency can be set between 1 Hz and 1 MHz, but the frequency that can actually be output depends on the clock frequency. Refer to *7-6-4 Pulse Output Specifications* in the *FQM1 Series Flexible Motion Controller Operation Manual* (Cat. No. O012) to verify the allowed output range. An error will occur and the instruction will not be executed if the specified frequency exceeds the allowed output range. If the specified frequency is below the allowed output range (except 0000 0000), the lower limit frequency will be output. If the output frequency is set to 0000 0000, the instruction will be treated as NOP(000) and the output status will be maintained.

The output frequency will not be changed unless a minimum of one pulse is output. For example, if 1 Hz is output when 20 MHz (1 Hz to 1 MHz) is being used, execution will not be enabled for 1 s while the 1-pulse output is being completed. The instruction can be executed, but a 1-pulse output wait time will be required until the frequency is actually changed. For instructions with automatic acceleration/deceleration, such as PLS2(887) or ACC(888), the frequency will be changed automatically according to the acceleration/

deceleration rate, but for either the start frequency or the acceleration/deceleration rate, a 1-pulse output wait time will be required. When using low frequencies, therefore, allow for delays in speed changes.

Operand Specifications

Area	P	T	N
CIO Area	---	---	CIO 0000 to CIO 6142
Work Area	---	---	W000 to W254
Auxiliary Bit Area	---	---	A448 to A958
Timer Area	---	---	T0000 to T0254
Counter Area	---	---	C0000 to C0254
DM Area	---	---	D00000 to D32766
Indirect DM addresses in binary	---	---	@ D00000 to @ D32767
Indirect DM addresses in BCD	---	---	*D00000 to *D32767
Constants	See operand description.	See operand description.	See operand description.
Data Resisters	---	---	---
Index Registers	---	---	---
Indirect addressing using Index Registers	---	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

PULS(886) sets the pulse type and number of pulses specified in T and N for the port specified in P. Actual output of the pulses is started later in the program using SPED(885) or ACC(888) in independent mode.

An error will occur if PULS(886) is executed while pulses are being output, so the number of output pulses cannot be changed. Execute this instruction with either the up-differentiated variation of the instruction (@PULS(886)) or an execution condition that is ON for just one cycle.

Once the number of pulses is determined by executing PULS(886), the calculated number of pulses will not be changed even if INI(880) is executed to change the pulse output PV.

It is also possible to specify values to move outside of the number of output pulses PV range (–2,147,483,648 to +2,147,483,647).

When pulse output is started in Electronic Cam Mode, the direction of rotation (CW or CCW) will be reflected in the Rotation Direction Flags (A874.08 and A875.08).

Execution Conditions

1. Setting the Number of Output Pulses for Relative Pulse Output

   To set the number of output pulses for a relative pulse output, execute PULS(886) with P set to #1 or #2, T set to #0, and the number of output pulses set in N and N+1.

2. Setting the Target Position for Absolute Pulse Output

   To set the target position for an absolute pulse output, execute PULS(886) with P set to #1 or #2, T set to #1, and the target position set in N and N+1.

3. Setting the Target Position for Electronic Cam Control and Starting the Output

To set the target position for electronic cam control and start the pulse output, execute PULS(886) with P set to #1 or #2, D set to #2 to #4. Set the specified position in the number of pulses word and the pulse output frequency (when D is #2 or #3) or the pulse output command cycle (when D is #4). The pulse output command cycle is the time until the specified pulses will be cut off. Normally, this interval lasts until the next PULS(886) instruction is executed for pulse output with an absolute position specified.

4. When PULS(886) was executed to start electronic cam control and the target position has been reached, the pulse output will stop automatically. The pulse output direction and number of pulses are determined by the following formulae:

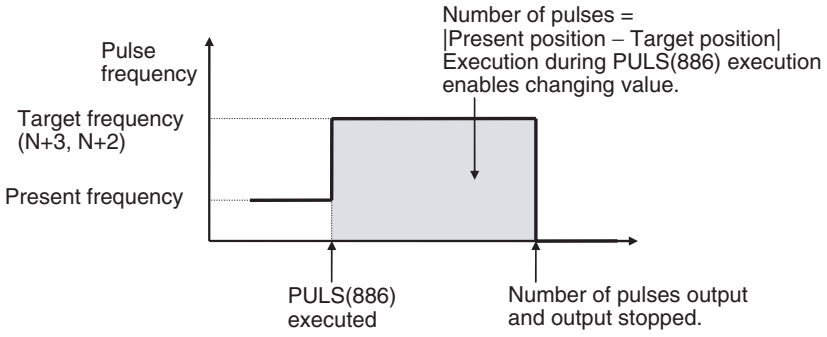
Value	Formula
Direction	PV < Target: CW direction, PV > Target: CCW direction PV = Target: Maintain status
Number of pulses	|PV of pulse output – target pulse amount|

5. The number of pulses actually output depends on the pulse type, as shown below.

   Relative pulses:
   Actual number of pulses = Specified number of output pulses

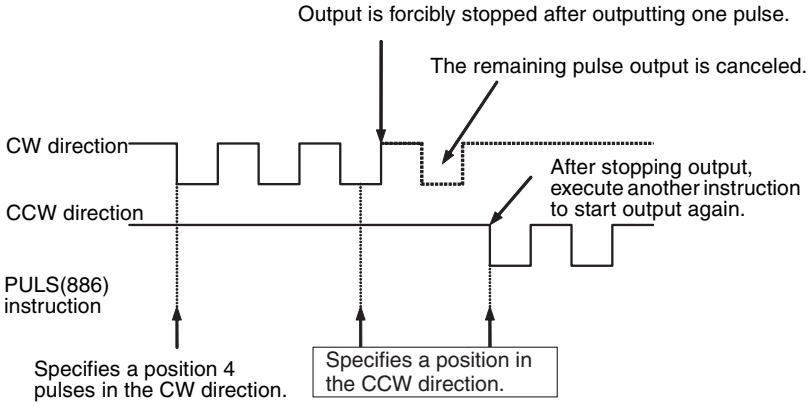
   Absolute pulses:
   Actual number of pulses = |Present position – target position|

6. If PULS(886) was executed to start a pulse output in electronic cam control mode and the pulses are still being output, PULS(886) can be executed in electronic cam control mode again to change the target position and pulse output frequency.



Note a) When the pulse output direction is reversed by the absolute position specified in electronic cam control, the pulse output will be forcibly stopped after completing one output pulse. The pulse wave will not be interrupted abruptly, but the remaining pulses will not be output. After the output is stopped, pulses will not be output automatically in the opposite direction. The output will start when another pulse output instruction is executed.

If another instruction is executed in the opposite direction of the current movement before the axis is forcibly stopped, the instruction will not be executed. It may take some time to forcibly stop movement when the pulse output frequency is low.

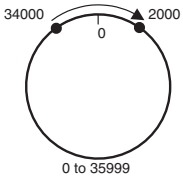


b) If a previously executed PULS(886) instruction reaches its target position while another PULS(886) instruction is being executed, the pulse output will be stopped and the later instruction will not be executed. In this case, execute PULS(886) again. The EQ Flag will go OFF in this case.

7. If a PULS(886) instruction is executed during an independent mode (positioning) pulse output, the number of output pulses or target position will not be set again. (The number of output pulses and target position cannot be changed during an independent mode pulse output.) Execute this instruction with either the up-differentiated version of the instruction (@PULS(886)) or an execution condition that is ON for just one cycle.
8. The pulse output operation mode must be set to *Electronic cam control* in order to perform electronic cam control. If a different operation mode is selected, PULS(886) will not be executed and an error will occur.
9. Any of the following methods can be used to stop pulses being output in electronic cam control mode by PULS(886).
   a) Execute INI(880) with C set to #0003 (immediate stop).
   b) The pulse output will stop when the target position is reached (immediate stop).
   c) Switch the Motion Control Module to PROGRAM mode.
10. Pulse outputs can be output independently and simultaneously from two output ports.
11. When the pulse output operation mode is set to *Absolute pulse (Linear mode)* and the specified target position is the same as the present position, PULS(886) will not be executed and the target position will not be set. The EQ Flag will go OFF in this case.
12. When D is set to #3 or #4 for absolute pulse output in ring mode, it is possible to send a command that moves through the 0 point.
    To move through 0 in the CW direction, calculate the target position to be set in N+1 and N as shown below.

    SV in N+1 and N = (ring value + 1) + target position

    For example, if the ring value range is 0 to 35,999 and you want to move from 34,000 to 2,000 through 0, use the following command.

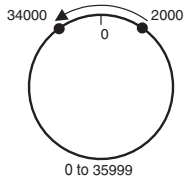


    SV in N+1 and N = (35,999 + 1) + 2,000 = 38,000 (9470 hex)

To move through 0 in the CCW direction, calculate the target position to be set in N+1 and N as shown below.

SV in N+1 and N = 0 − target position

For example, if the ring value range is 0 to 35,999 and you want to move from 2,000 to 34,000 through 0, use the following command.



SV in N+1 and N = 34,000 − 36,000 = −2,000 (FFFF F830 hex)

Note Do not set an SV in N+1 and N that would move around the ring more than once. The correct position may not be calculated (including moving from the present position through 0 and back to the present position).

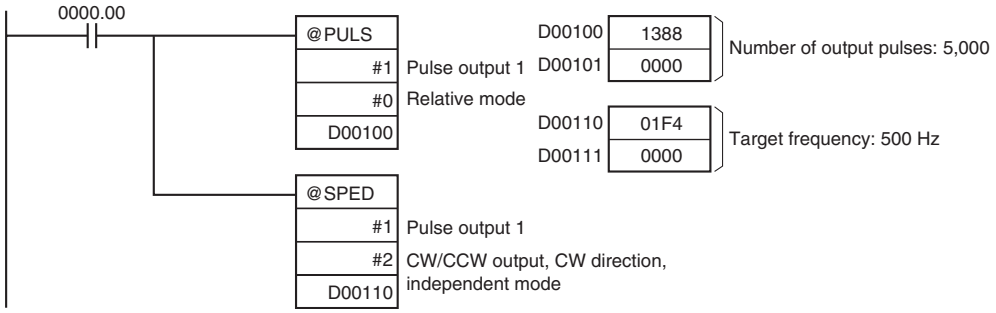
13. If D is set to #3 or #4 when using absolute pulse outputs in Ring Mode and a new target position is being specified after starting execution of a movement that passes through 0, use PRV(881) to read the present value of the pulse output to see whether 0 has been passed. If the new target position will cause movement to pass through 0 again, refer to item 12, above, to set the value in N+1 and N. If N+1 and N are not set correctly, the pulse output may be in the wrong direction. Or, in this case, set D to #3 (and not #4) so that the pulse output frequency is not automatically calculated. The calculated frequency may not always be correct.

Flags

Name	Label	Operation
Error Flag	ER	ON if P is set to a value other than #1 or #2. ON if D is set to a value other than #0 to #4. ON if the pulse output operation mode is set to *1 shot* or *Calculation (time measurement)* mode in the System Setup. ON if an instruction controlling pulse output is being executed in the main program, an interrupt occurs, and PULS(886) is executed in the interrupt task. OFF in all other cases.
Equal Flag	=	Operation when T is set to #0 or #1: • ON when the target position is set with PULS(886). • OFF when the target position could not be set with PULS(886). Operation when T is set to #2, #3, #4: • ON when the pulse output is started by PULS(886). • OFF when the pulse output could not be started by PULS(886).

Example

When CIO 0000.00 goes from OFF to ON in the following programming example, PULS(886) sets the number of output pulses for pulse output 1. A relative value of 5,000 pulses is set. SPED(885) is executed next to start pulse output using the CW/CCW method in the clockwise direction in independent mode at a target frequency of 500 Hz.

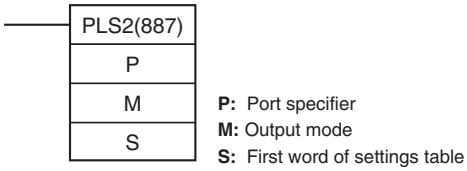


3-20-6 PULSE OUTPUT: PLS2(887)

Purpose

PLS2(887) outputs pulses to the specified port based on the specified target position, target frequency, startup frequency, acceleration rate, and deceleration rate. The acceleration rate and deceleration rate can be set separately. Only independent mode positioning is supported.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	PLS2(887)
	Executed Once for Upward Differentiation	@PLS2(887)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

P: Port Specifier

The port specifier indicates the port.

P	Port
#1	Pulse output 1
#2	Pulse output 2

M: Output Mode

M	Mode
#0	CW direction
#1	CCW direction

S: First Word of Settings Table

Word	Content
T	Target position (8-digit hexadecimal)
T+1	Relative pulse output: 0000 0000 to FFFF FFFF Absolute pulse output (linear mode): 8000 0000 to 7FFF FFFF
T+2	Target frequency (8-digit hexadecimal)
T+3	0000 0001 to 000F 4240
T+4	Starting frequency (8-digit hexadecimal)
T+5	0000 0000 to 000F 4240

Word	Content
T+6	Acceleration rate (4-digit hexadecimal) 0001 to 270F
T+7	Deceleration rate (4-digit hexadecimal) 0001 to 270F

The acceleration rate and deceleration rate specify the amount that the frequency will be changed each 2 ms or 1 ms. Set the rates in 1-Hz units. The 0001 to 270F hex setting range corresponds to a 1 Hz to 9.999 kHz range.)

The target frequency specifies the frequency reached after acceleration. Set the frequency in 1-Hz units. (The 0000 0001 to 000F 4240 hex setting range corresponds to a 1 Hz to 1 MHz range.)

The starting frequency specifies the frequency at which the output starts. Set the frequency in 1-Hz units. (The 0000 0000 to 000F 4240 hex setting range corresponds to a 0 Hz to 1 MHz range.)

The target frequency can be set between 1 Hz and 1 MHz and the starting frequency can be set between 0 Hz and 1 MHz. But the frequency that can actually be output depends on the clock frequency. Refer to *7-6-4 Pulse Output Specifications* in the *FQM1 Series Flexible Motion Controller Operation Manual* (Cat. No. O012) to verify the allowed output range. An error will occur and the instruction will not be executed if the specified frequency exceeds the allowed output range. If the specified frequency is below the allowed output range, the lower limit frequency will be output.

The output frequencies are obtained by dividing the Motion Control Module's clock pulse with an integer dividing ratio, meaning the actual output frequency can be different from the set frequency.
Also, the pulse frequencies actually output during acceleration/deceleration are frequencies that can actually be output with the integer dividing ratio. If the acceleration or deceleration rate is low, there may not be an change in the frequency in every 2 ms interval.
Refer to *Precautions when Using Pulse Outputs* in the *FQM1 Series Flexible Motion Controller Operation Manual* (Cat. No. O012) for details.

The output frequency will not be changed unless a minimum of one pulse is output. For example, if 1 Hz is output when 20 MHz (1 Hz to 1 MHz) is being used, execution will not be enabled for 1 s while the 1-pulse output is being completed. The instruction can be executed, but a 1-pulse output wait time will be required until the frequency is actually changed. For instructions with automatic acceleration/deceleration, such as PLS2(887) or ACC(888), the frequency will be changed automatically according to the acceleration/deceleration rate, but for either the start frequency or the acceleration/deceleration rate, a 1-pulse output wait time will be required. When using low frequencies, therefore, allow for delays in speed changes.

Operand Specifications

Area	P	M	S
CIO Area	---	---	CIO 0000 to CIO 6136
Work Area	---	---	W000 to W248
Auxiliary Bit Area	---	---	A448 to A952
Timer Area	---	---	T0000 to T0248
Counter Area	---	---	C0000 to C0248
DM Area	---	---	D00000 to D32760
Indirect DM addresses in binary	---	---	@ D00000 to @ D32767

Area	P	M	S
Indirect DM addresses in BCD	---	---	*D00000 to *D32767
Constants	See operand description.	See operand description.	DR0 to DR15
Data Resisters	---	---	---
Index Registers	---	---	---
Indirect addressing using Index Registers	---	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

PLS2(887) operation proceeds as described below.

1. PLS2(887) starts a pulse output at the specified starting frequency and accelerates the output each 1 ms or 2 ms by the specified frequency step.
2. When the target frequency is reached, the frequency acceleration stops and the pulse output continues at a constant frequency.
3. When the distance to the target position (calculated number of pulses from the target position) reaches the deceleration point calculated from the deceleration rate and frequency, PLS2(887) decelerates the output each 1 ms or 2 ms by the specified frequency step. PLS2(887) stops the pulse output at the target position.

If the specified number of pulses is too low for the acceleration and deceleration stages, PLS2(887) will not perform trapezoidal acceleration/deceleration. The pulse output will start decelerating before the target frequency is reached or the pulse output may not even accelerate from the starting frequency. In this case, the PLS2(887) Target Frequency Not Reached Flag (A874.02 or A875.02) will turn ON.

The PLS2(887) Target Frequency Not Reached Flag is turned ON during acceleration when the deceleration point is reached and turned OFF when deceleration is completed.

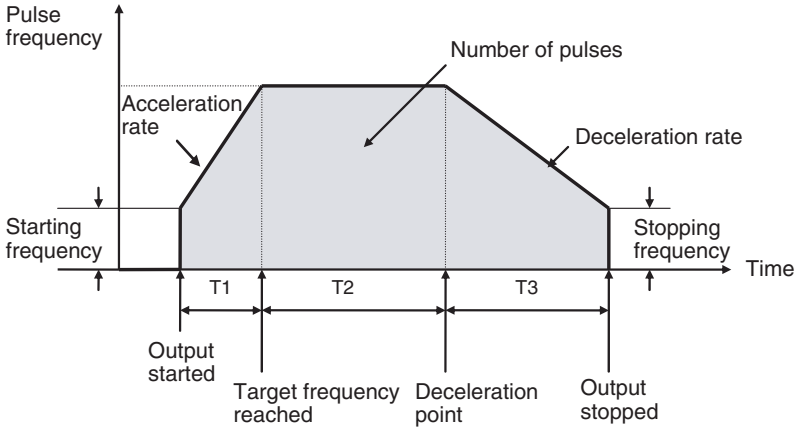
Note PLS2(887) calculates the number of pulses required for deceleration based on the specified target position, starting speed, and deceleration rate. This value is known as the number of deceleration pulses and deceleration starts when the number of remaining pulses reaches the number of deceleration pulses. This point is known as the deceleration point.
If the total number of output pulses is less than number of deceleration pulses, the total number of output pulses will be output resulting in triangular control.
If the total number of output pulses is greater than number of deceleration pulses but less than the number required for acceleration and deceleration, the deceleration point will be reached during acceleration and deceleration will start at that point.

Operation

When PLS2(887) is executed just once, the pulse output is controlled until it stops with the specified settings, so use either the up-differentiated variation of the instruction (@PLS2(887)) or an execution condition that is ON for just one cycle.

The number of pulses actually output depends on the pulse type, as shown below.

- Relative pulse output:
  Actual number of pulses = Target position
- Absolute pulse output (linear mode):
  Actual number of pulses = |Target position − Present position|



$T1 \approx 0.002 \times (\text{Target frequency} - \text{Starting frequency}) \div (\text{Acceleration rate})$

$T3 \approx 0.002 \times (\text{Target frequency} - \text{Starting frequency}) \div (\text{Deceleration rate})$

$T2 \approx (\text{Number of pulses} - ((\text{Target frequency} + \text{Starting frequency}) \times (T1+T3) \div 2)) \div \text{Target frequency}$

These equations are approximations. The actual values of T1, T2, and T3 will vary depending on the pulse output operating conditions. (The number of output pulses is output accurately.)

With version 3.3 or higher, if the starting frequency is set to 0 Hz when the pulse output clock is set to 20 MHz (full range), the starting frequency will be the acceleration rate and the stopping frequency will be the smaller of the acceleration rate or the deceleration rate.

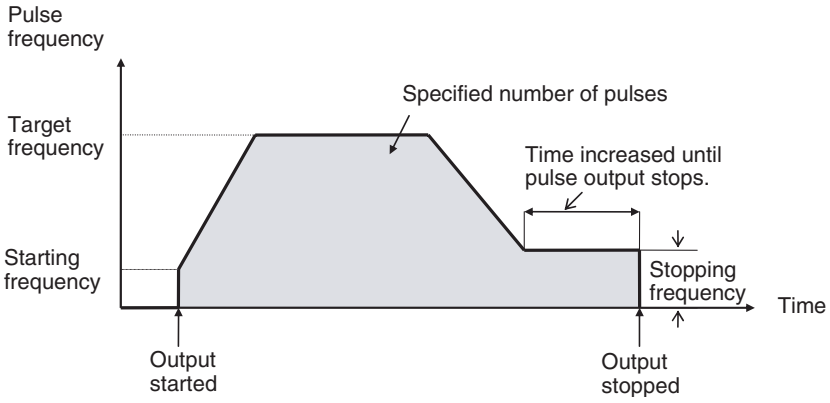
Note

1. The stopping frequency is the same as the starting frequency. If the stopping frequency is below the allowed output range (determined by the selected clock frequency), the lower limit frequency will be output.
2. If the number of pulses is lower than the number required for acceleration and deceleration, trapezoidal acceleration/deceleration will not be possible and deceleration will begin before the target frequency is reached. When the starting frequency has been set to 0, pulses may be output at the lower limit output frequency for the selected clock frequency.
3. When using PLS2(887) for an absolute pulse output (linear mode), check the present position before specifying the CW or CCW direction. Pulses will not be output if the direction setting is incorrect for the relationship between the present position and target position. An error will occur if PLS2(887) is executed with settings that prevent pulse output.
4. PLS2(887) will not be executed and an error will occur if a pulse output is already being output from the specified port.
5. Any of the following methods can be used to stop pulses being output by PLS2(887).
   a) Execute INI(880) with C set to #0003.
   b) The pulse output will stop when the target position is reached.
   c) Switch the Motion Control Module to PROGRAM mode.
   d) Execute ACC(888) with M set to #0008 or #0009 and the target frequency set to 0 Hz. Pulse output will decelerated to a stop. (If number

of output pulses set for PULS(886) in independent mode is reached, an immediate stop will be performed.)

e) Execute ACC(888) with M set to #0008 or #0009 and the target frequency set to 0 Hz. Pulse output will decelerated to a stop. (If number of output pulses set for PULS(886) in independent mode is reached, an immediate stop will be performed.)

⚠ Caution With PLS2(887), the output may continue for some time at the stopping frequency, depending on factors such as the acceleration/deceleration rate and the target speed. (Even when this happens, the correct number of pulses will be output.)



If this problem occurs, correct the system by adjusting the acceleration rate, deceleration rate, target speed, or starting frequency. If the starting frequency is low, it is easy to adjust the system by increasing the starting frequency to 500 Hz or higher.

Note PLS2(887) cannot be used if the pulse output operation mode is set to *Absolute pulse (Circular mode)* or *1 shot* mode in the System Setup.

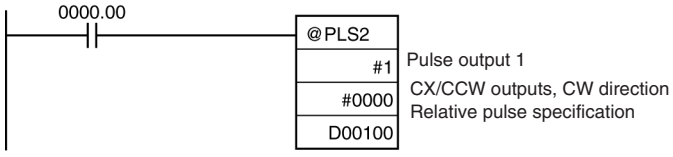
Flags

Name	Label	Operation
Error Flag	ER	ON if P is set to a value other than #1 or #2. ON if M is set to a value other than #0 or #1. ON if the pulse output operation mode is set to *Absolute pulse (Circular mode)* or *One-Short pulse output,* or *Time measurement using pulse counter* in the System Setup. ON if the target frequency, acceleration rate, or deceleration rate setting is incorrect. (For example, if the target frequency is less than the starting frequency.) ON if the CW/CCW direction setting is incorrect for the relationship between the present position and target position. ON if pulses are already being output from the specified output port. ON if an instruction controlling pulse output is being executed in the main program, an interrupt occurs, and PLS2(887) is executed in the interrupt task. OFF in all other cases.

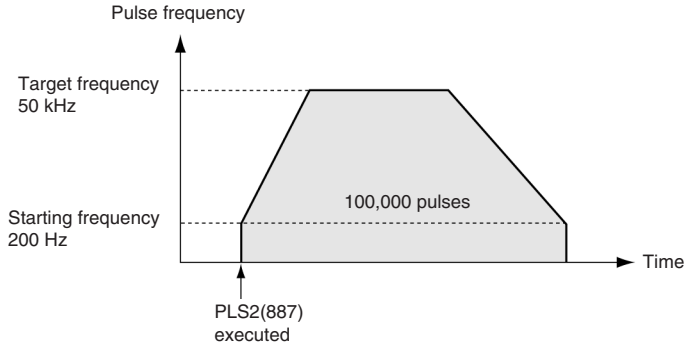
Example

When CIO 0000.00 goes from OFF to ON, PLS2(887) starts a relative pulse output of 100,000 pulses from pulse output 1. The pulse output begins at a starting frequency of 200 Hz and accelerates to a target frequency of 50 kHz at an acceleration rate of 500 Hz/2 ms. When the output reaches the deceler-

ation point, it decelerates back to the starting frequency of 200 Hz at a deceleration rate of 250 Hz/2 ms. The pulse output stops at 200 Hz.



D00100	86A0	Number of output pulses: 100,000 pulses
D00101	0001	
D00102	C350	Target frequency: 50 kHz
D00103	0000	
D00104	00C8	Starting frequency: 200 Hz
D00105	0000	
D00106	01F4	Acceleration rate: 500 Hz/2 ms
D00107	00FA	Deceleration rate: 250 Hz/2 ms



3-20-7 ACCELERATION CONTROL: ACC(888)

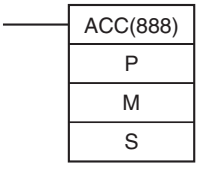
Purpose

ACC(888) outputs pulses to the specified output port at the specified frequency using the specified acceleration and deceleration rate. (The acceleration rate is the same as the deceleration rate.)

ACC(888) can perform positioning (independent mode) or speed control (continuous mode). For positioning, ACC(888) is used in combination with PULS(886). ACC(888) can also be executed during pulse output to change the target frequency or acceleration/deceleration rate, enabling smooth (sloped) speed changes.

This instruction is supported by the FQM1-MMP22 and FQM1-MMA22 Motion Control Modules only.

Ladder Symbol



P: Port specifier
M: Output mode
S: First word of settings table

Variations

Variations		
	Executed Each Cycle for ON Condition	ACC(888)
	Executed Once for Upward Differentiation	@ACC(888)
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

P: Port Specifier

The port specifier specifies the port where the pulses will be output.

P	Port
#1	FQM1-MMP22: Pulse output 1 FQM1-MMA22: Analog output 1
#2	FQM1-MMP22: Pulse output 2 FQM1-MMA22: Analog output 2

M: Output Mode

The value of M determines the pulse output mode in the FQM1-MMP22.

Module	Value of M	Mode
FQM1-MMP22 (Pulse output)	#0000	CW, acceleration, continuous mode
	#0001	CCW, acceleration, continuous mode
	#0002	CW, deceleration, continuous mode
	#0003	CCW, deceleration, continuous mode
	#0004	CW, acceleration, independent mode
	#0005	CCW, acceleration, independent mode
	#0006	CW, deceleration, independent mode
	#0007	CCW, deceleration, independent mode
	#0008	Continuous mode speed change
	#0009	Independent mode speed change
FQM1-MMA22	#0000 (fixed)	---

S: First Word of Settings Table

- FQM1-MMP22 Settings

Word	Content
S	Specifies the acceleration and deceleration rates • 2 ms cycle (speed-change cycle) Set a frequency change of 0001 to 270F hex (1 to 9,999 Hz, in 1-Hz units). The frequency is changed by this amount every 2 ms. • 1 ms cycle (speed-change cycle) Set a frequency change of 0001 to 270F hex (1 to 9,999 Hz, in 1-Hz units). The frequency is changed by this amount every 1 ms.
S+1, S+2	Specifies the target frequency in 8-digit hexadecimal. (S+1 contains the rightmost 4 digits and S+2 contains the leftmost 4 digits.) Set a frequency of 0000 0000 to 000F 4240 hex (0 Hz to 1 MHz in 1-Hz units). This is the target frequency after acceleration.

The target frequency can be set between 0 Hz and 1 MHz, but the frequency that can actually be output depends on the clock frequency. Refer to *7-6-4 Pulse Output Specifications* in the *FQM1 Series Flexible Motion Controller Operation Manual* (Cat. No. O012) to verify the allowed output range. An error will occur and the instruction will not be executed if the specified frequency exceeds the allowed output range. If the specified frequency is below the allowed output range, the lower limit frequency will be output.

The output frequencies are obtained by dividing the Motion Control Module's clock pulse with an integer dividing ratio, meaning the actual output frequency can be different from the set frequency.

Also, the pulse frequencies actually output during acceleration/deceleration are frequencies that can actually be output with the integer dividing ratio. If the acceleration or deceleration rate is low, there may not be an change in the frequency in every 2 ms interval.

Refer to *Precautions when Using Pulse Outputs* in the *FQM1 Series Flexible Motion Controller Operation Manual* (Cat. No. O012) for details.

The output frequency will not be changed unless a minimum of one pulse is output. For example, if 1 Hz is output when 20 MHz (1 Hz to 1 MHz) is being used, execution will not be enabled for 1 s while the 1-pulse output is being completed. The instruction can be executed, but a 1-pulse output wait time will be required until the frequency is actually changed. For instructions with automatic acceleration/deceleration, such as PLS2(887) or ACC(888), the frequency will be changed automatically according to the acceleration/deceleration rate, but for either the start frequency or the acceleration/deceleration rate, a 1-pulse output wait time will be required. When using low frequencies, therefore, allow for delays in speed changes.

If the target frequency is set to 0 (Hz) and the mode specifier is set to #0008 or #0009, the instruction can be executed at any time in the same way as when stopping pulse outputs using INI(880).

- FQM1-MMA22 Settings

Word	Content
S	Specifies the analog output's rate-of-change in 4-digit hexadecimal. The analog output value will be changed by this amount every 2 ms. –10 to 10 V 0000 to 2AF8 hex (0 to 11,000 decimal) = 0 to 110% (0 to +22 V) 0 to 10 V, 0 to 5 V, or 1 to 5 V 0000 to 1130 hex (0 to 4,400 decimal) = 0 to 110% (0 to +11 V, 0 to +5.5 V, or 0 to +4.4 V)
S+1	Specifies the target analog output value in 4-digit hexadecimal. –10 to 10 V EC78 to 1388 hex (–5,000 to 5,000 decimal) (resolution: 10,000) corresponding to 0% to 100% voltage (–10 to 10 V) (Actually, the setting range is EA84 to 157C (–5,500 to 5,500 decimal) corresponding to –5% to 105% voltage (–11 to 11 V).) 0 to 10 V, 0 to 5 V, or 1 to 5 V: 0000 to 0FA0 hex (0000 to 4,000 decimal) (resolution: 4,000) corresponding to 0% to 100% of the FS range. (Actually, the setting range is FF38 to 1068 (–200 to 4,200 decimal) corresponding to –5% to 105% voltage (–0.5 to 10.5 V, –0.25 to 5.25 V, or 0.8 to 5.2 V).)

Note An error will occur and the ER Flag will be turned ON if the settings exceed the ranges listed above.

Operand Specifications

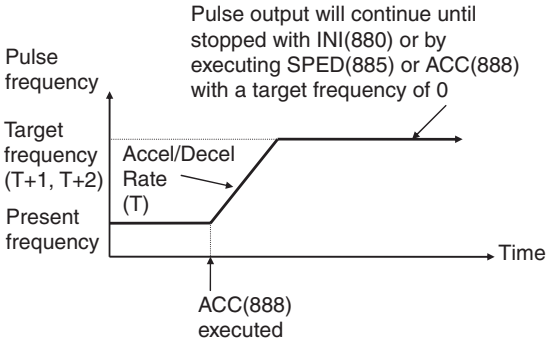
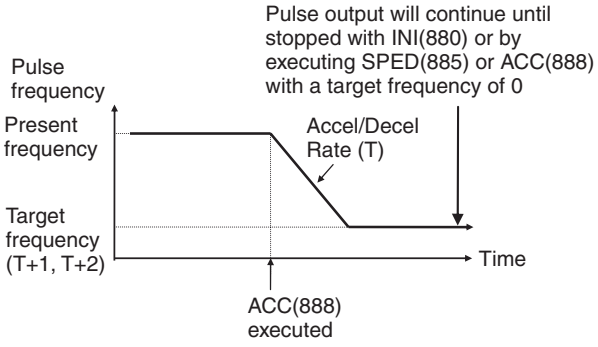
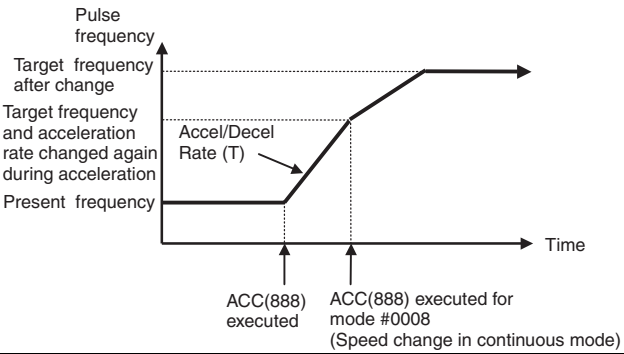
Area	P	M	S
CIO Area	---	---	CIO 0000 to CIO 6141
Work Area	---	---	W000 to W253
Auxiliary Bit Area	---	---	A448 to A957
Timer Area	---	---	T0000 to T0253
Counter Area	---	---	C0000 to C0253
DM Area	---	---	D00000 to D32765
Indirect DM addresses in binary	---	---	@ D00000 to @ D32767
Indirect DM addresses in BCD	---	---	*D00000 to *D32767
Constants	See operand description.	See operand description.	---
Data Resisters	---	---	---

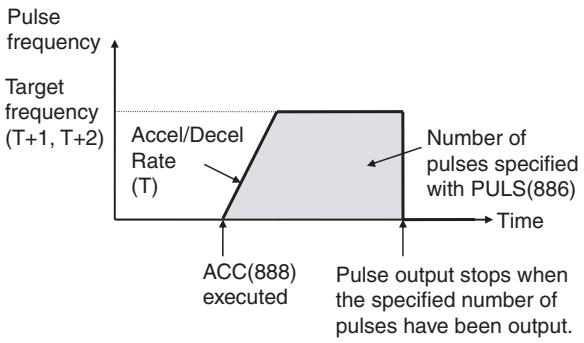
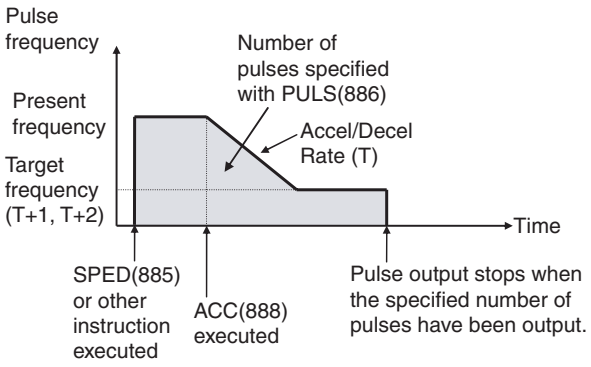
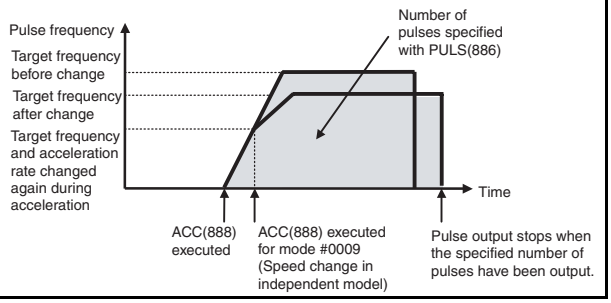
Area	P	M	S
Index Registers	---	---	---
Indirect addressing using Index Registers	---	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

Pulse Output (FQM1-MMP22)

Pulses can be output in independent mode (positioning) or continuous mode (speed control).

Mode	Description	Frequency changes
Speed control	Mode #0000 or #0001 (Acceleration + Continuous mode) The frequency is increased from the present frequency to the target frequency at the specified acceleration/deceleration rate. The pulse output continues at the target frequency. The output will continue until stopped with INI(880) or by executing SPED(885) or ACC(888) with a target frequency of 0.	
	Mode #0002 or #0003 (Deceleration + Continuous mode) The frequency is decreased from the present frequency to the target frequency at the specified acceleration/deceleration rate. The pulse output continues at the target frequency. The output will continue until stopped with INI(880) or by executing SPED(885) or ACC(888) with a target frequency of 0.	
	Mode #0008 (Speed changes in Continuous mode) Pulse output is continued while automatically determining the pulse output direction and the need for acceleration or deceleration. If this mode is used when pulses are not being output, the instruction will not be executed and the ER Flag will turn ON.	

Mode	Description	Frequency changes
Positioning	Mode #0004 or #0005 (Acceleration + Independent mode) The frequency is increased to the target frequency at the specified acceleration/deceleration rate. The pulse output continues at the target frequency until the number of pulses specified with PULS(886) have been output. The pulse output stops automatically at that point. (With independent mode, the number of pulses must be set in advance with PULS(886) and the output operates according to that setting.)	
	Mode #0006 or #0007 (Deceleration + Independent mode) The frequency is decreased from the present frequency to the target frequency at the specified acceleration/deceleration rate. The pulse output continues at the target frequency until the number of pulses specified with PULS(886) have been output. The pulse output stops automatically at that point. (With independent mode, the number of pulses must be set in advance with PULS(886) and the output operates according to that setting.)	
	Mode #0009 (Speed changes in Independent mode) Pulse output is continued while automatically determining the pulse output direction and the need for acceleration or deceleration. If this mode is used when pulses are not being output, the instruction will not be executed and the ER Flag will turn ON.	

- Independent Mode (Positioning) Operation

  In independent mode, only a preset number of pulses are output. Set the number of output pulses in advance with PULS(886).

  In independent mode (positioning), the number of output pulses must be specified in advance with PULS(886). (No pulses will be output if the number of output pulses is not specified before executing ACC(888).) If the pulse output has been stopped, it is necessary to set the number of output pulses again with PULS(886).

  If the number of output pulses set with PULS(886) is less than the number of pulses required for acceleration (Pulses ≈ Time to reach target frequency × (Target frequency − Starting frequency) ÷ 2), the pulse output will stop before the target frequency is reached.

  Likewise, if the number of output pulses set with PULS(886) is less than the number of pulses required for deceleration (Pulses ≈ Time to reach target frequency × (Target frequency − Starting frequency) ÷ 2), the pulse output will stop before the target frequency is reached.

  If the target frequency is set to 0 and the number of output pulses set with PULS(886) is greater than the number of pulses required for deceleration (Pulses ≈ Time to reach target frequency × (Target frequency − Starting

frequency) ÷ 2), the pulse output will stop before the specified number of pulses have been output.

If a high acceleration/deceleration rate and low number of output pulses are set, the effective operation will have almost no acceleration/deceleration and the system will operate at nearly a steady speed.

- Continuous Mode (Speed Control) Operation
  In continuous mode, pulses are output continuously until stopped by executing SPED(885) with a target frequency of 0, executing or ACC(888) with M set to #0008 or #0009 and a target frequency of 0, executing INI(880) with C = #0003, or switching the Motion Control Module to PROGRAM mode.
- Continuous Mode and Independent Mode Operation
  The Rotation Direction Flags (A874.08 and A875.08) will reflect the direction of rotation (CW or CCW) during pulse output.

Analog Output (FQM1-MMA22)

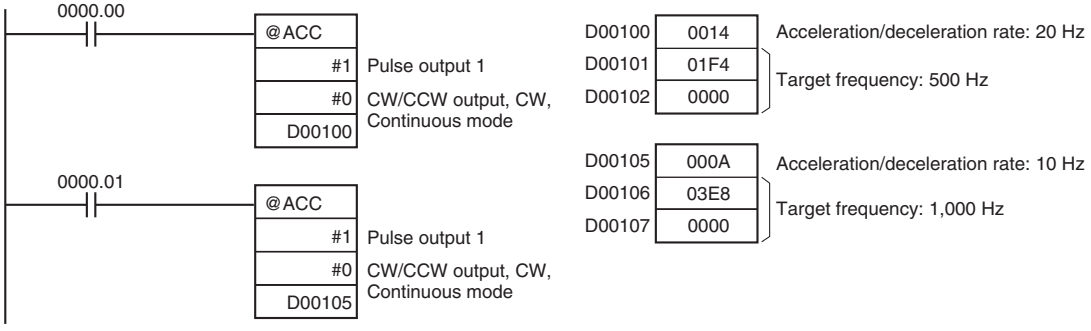
1. To generate an analog output, execute ACC(888) with P set to #0001 or #0002, M set to #0000, and S set to the first of two words containing the analog output rate-of-change (slope) and target output value. ACC(888) needs to be executed just once, so use either the up-differentiated variation of the instruction (@ACC(888)) or an execution condition that is ON for just one cycle.
2. When ACC(888) is executed, the analog output is increased every 2 ms by the rate-of-change amount specified in S. When the target analog output value is reached, the value will stop increasing and remain at the target value.
3. Analog outputs can be output independently and simultaneously from two output ports.
4. Analog outputs are not distinguished as continuous or independent outputs. The analog output generated by ACC(888) will maintain its output value until one of the following events occurs:
   a) SPED(885) or ACC(888) is executed with a different target value.
   b) The *Output stop function* is set to a setting other than *Hold* and the Motion Control Module is switched to PROGRAM mode or the Analog Output Conversion Enable Bit (A814.00 or A815.00) is reset to 0.
5. If an analog output is being generated by SPED(885), that output value can be changed by executing ACC(888) with a different analog output value.
6. The present analog output value will not be changed and an error will occur if ACC(888) is executed while another ACC(888) is generating an accelerating or decelerating analog output (i.e., the earlier ACC(888) analog output has not reached its target value).

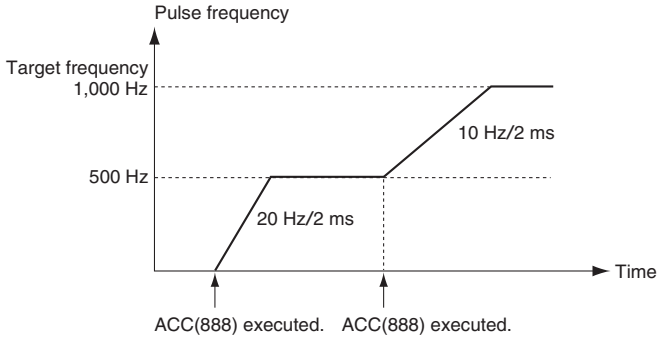
Flags

Name	Label	Operation
Error Flag	ER	ON if P is set to a value other than #0001 or #0002. For the FQM1-MMP22, ON if M is not set between #0 and #7. For the FQM1-MMA22, ON if M is not set to #0. For the FQM1-MMP22, ON if the specified frequency is not supported at the selected Motion Control Module clock frequency. For the FQM1-MMP22, ON if ACC(888) is executed when the present pulse output cannot be changed. (For example, the pulse output cannot be changed when pulses are already being output by an PLS2(887) instruction but the target value has not been reached.) ON if an instruction controlling a pulse output or analog output is being executed in the main program, an interrupt occurs, and ACC(888) is executed in the interrupt task. OFF in all other cases.

Example

When CIO 0000.00 goes from OFF to ON, ACC(888) starts pulse output from pulse output 1 in continuous mode in the clockwise direction using the CW/CCW method. Pulse output is accelerated at a rate of 20 Hz every 2 ms until the target frequency of 500 Hz is reached. When CIO 0000.01 goes from OFF to ON, ACC(888) changes to an acceleration rate of 10 Hz every 2 ms until the target frequency of 1,000 Hz is reached.



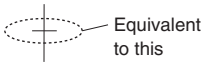
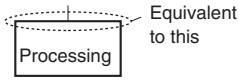


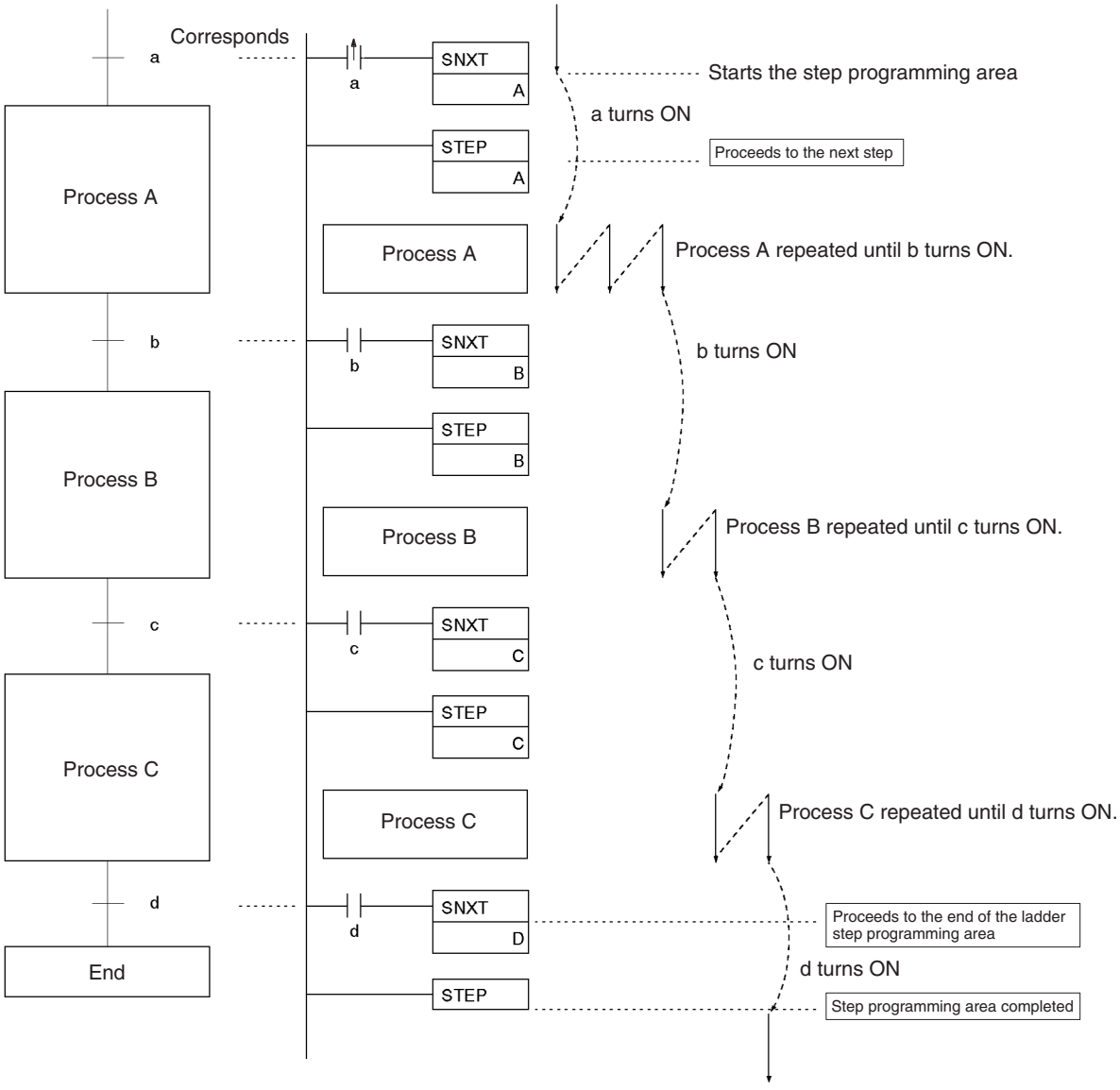
3-21 Step Instructions

This section describes Step Instructions, which are used to set up break points between sections in a large program so that the sections can be executed as units and reset upon completion.

Instruction	Mnemonic	Function code	Page
STEP DEFINE	STEP	008	563
STEP START	SNXT	009	563

In the FQM1, STEP(008)/SNXT(009) can be used together to create step programs.

Instruction	Operation	Diagram
SNXT(009): STEP START	Controls progression to the next step of the program.	Corresponds 
STEP(008): STEP DEFINE	Indicates the start of a step. Repeats the same step program until the conditions for progression to the next step are established.	Corresponds 



Note Work bits are used as the control bits for A, B, C and D.

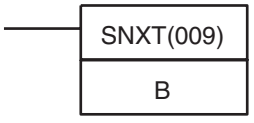
3-21-1 STEP DEFINE and STEP START: STEP(008)/SNXT(009)

Purpose

SNXT(009) is placed immediately before the STEP(008) instruction and controls step execution by turning the specified control bit ON. If there is another step immediately before SNXT(009), it also turns OFF the control bit of that process.

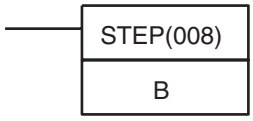
STEP(008) is placed immediately after the SNXT(009) instruction and before each process. It defines the start of each process and specifies the control bit for it. It is also placed at the end of the step programming area after the last SNXT(009) to indicate the end of the step programming area. When it appears at the end of the step programming area, STEP(008) does not take a control bit.

Ladder Symbols



B: Bit

When defining the beginning of a step, a control bit is specified as follows:.



B: Bit

When defining the end of a step, a control bit is not specified as follows:

Variations

Variations		
	Executed Each Cycle for ON Condition	STEP(008)/ SNXT(009)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
Not allowed	OK	Not allowed	Not allowed

Operand Specifications

Area	B
CIO Area	---
Work Area	W000.00 to W255.15
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

SNXT(009)

SNXT(009) is used in the following three ways:

1,2,3...

1. To start step programming execution.
2. To proceed to the next step control bit.
3. To end step programming execution.

The step programming area is from the first STEP(008) instruction (which always takes a control bit) to the last STEP(008) instruction (which never takes a control bit).

Starting Step Execution

SNXT(009) is placed at the beginning of the step programming area to start step execution. It turns ON the control bit specified for B for the next STEP(008) and proceeds to step B (all instructions after STEP(008) B). A up-differentiated execution condition must be used for the SNXT(009) instruction that starts step programming area execution, or step execution will last for only one cycle. (If SNXT(009) is incorrectly placed at the beginning of the step programming area, it will operate in the same manner as SET (i.e., once it is turned ON, it will maintain its status until it is turned OFF by RESET.) If SNXT(009) is already ON when starting the next cycle, the step programming area will not be executed in the following cycles.)

Proceeding to the Next Step

When SNXT(009) occurs in the middle of the step programming area, it is used to proceed to the next step. It turns OFF the previous control bit and turns ON the next control bit B, for the next step, thereby starting step B (all instructions after STEP(008) B).

Ending the Step Programming Area

When SNXT(009) is placed at the very end of the step programming area, it ends step execution and turns OFF the previous control bit. The control bit specified for B is a dummy bit. This bit will however be turned ON, so be sure to select a bit that will not cause problems.

STEP(008)

STEP(008) functions in following 2 ways, depending on its position and whether or not a control bit has been specified.

1,2,3... 1. Starts a specific step.

2. Ends the step programming area (i.e., step execution).

Starting a Step

STEP(008) is placed at the beginning of each step with an operand, B, that serves as the control bit for the step.

The control bit B will be turned ON by SNXT(009) and the instruction in the step will be executed from the one immediately following STEP(008). A200.12 (Step Flag) will also turn ON when execution of a step begins.

After the first cycle, step execution will continue until the conditions for changing the step are established, i.e., until the SNXT(009) instruction turns ON the control bit in the next STEP(008).

When SNXT (009) turns ON the control bit for a step, the control bit B of the current instruction will be reset (turned OFF) and the step controlled by bit B will become interlocked.

Handling of outputs and instructions in a step will change according to the ON/OFF status of the control bit B. (The status of the control bit is controlled by SNXT(009)). When control bit B is turned OFF, the instructions in the step are reset and are interlocked. Refer to the following tables.

Control bit status	Handling
ON	Instructions in the step are executed normally.
ON→OFF	Bits and instructions in the step are interlocked as shown in the next table.
OFF	All instructions in the step are processed as NOPs.

Interlock Status (IL)

Instruction output		Status
Bits specified for OUT, OUT NOT		All OFF
The following timer instructions: TIM, TIMH(015), and TMHH(540)	PV	0000 hex (reset)
	Completion Flag	OFF (reset)
Bits or words specified for other instructions (see note)		Holds the previous status (but the instructions are not executed)

Note Indicates all other instructions, such as SET, REST, CNT, CNTR(012), SFT(010), and KEEP(011).

The STEP(008) instruction must be placed at the beginning of each step. STEP(008) is placed at the beginning of a step area to define the start of the step.

Ending the Step Programming Area

STEP(008) is placed at the end of the step programming area without an operand to define the end of step programming When the control bit preceding a SNXT(009) instruction is turned OFF, step execute is stopped by SNXT(009).

Flags:STEP(008)

Name	Label	Operation
Error Flag	ER	ON when the specified bit B is not in the WR area. ON when STEP(008) is used in an interrupt program. OFF in all other cases.

Flags:SNXT(009)

Name	Label	Operation
Error Flag	ER	ON when the specified bit B is not in the WR area. ON when SNXT(009) is used in an interrupt program. OFF in all other cases.

Precautions

The control bit, B, must be in the Work Area for STEP(008)/SNXT(009).

A control bit for STEP(008)/SNXT(009) cannot be use anywhere else in the ladder diagram. If the same bit is used twice, a duplication bit error will occur.

If SBS(091) is used to call a subroutine from within a step, the subroutine outputs and instructions will not be interlocked when the control bit turns OFF.

Control bits within one section of step programming must be sequential and from the same word.

SNXT(009) will be executed only once, i.e., on the rising edge of the execution condition.

Input SNXT(009) at the end of the step programming area and make sure that the control bit is a dummy bit in the Work Area. If a control bit for a step is used in the last SNXT(009) in the step programming area, the corresponding step will be started when SNXT(009) is executed.

An error will occur and the Error Flag will turn ON if the operand B specified for SNXT(009) or STEP(008) is not in the Work Area or if the step program has been placed anywhere but in a cyclic task.

A200.12 (Step Flag) is turned ON for one cycle when STEP(008) is executed. This flag can be used to conduct initialization once the step execution has started.

Placement Conditions for Step Programming Areas (STEP B to STEP)

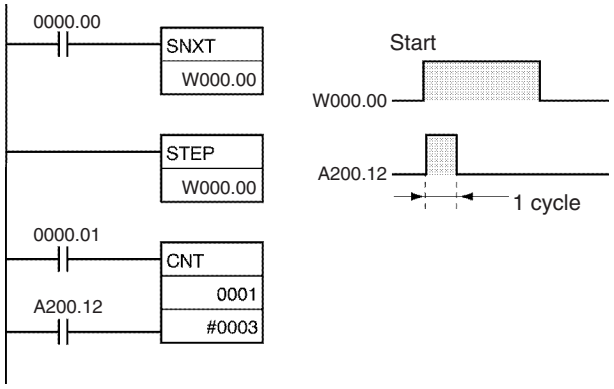
STEP(008) and SNXT(009) cannot be used inside of subroutines, interrupt tasks, or block programs.

Be sure that two steps are not executed during the same cycle.

Instructions that Cannot be Used Within Step Programs

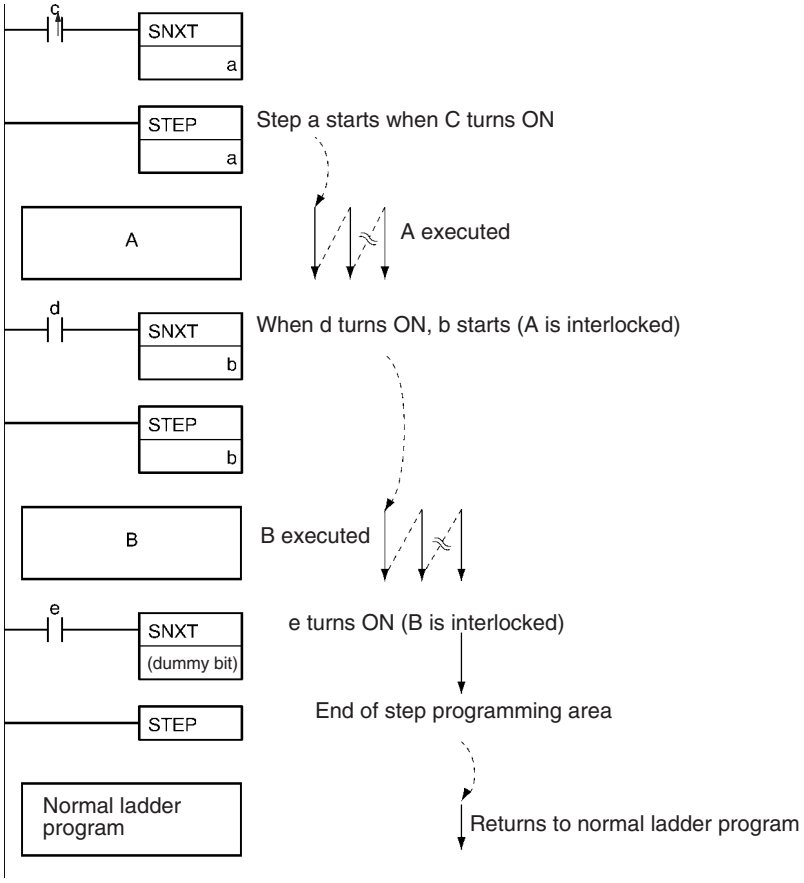
The instructions that cannot be used within step programs are listed in the following table.

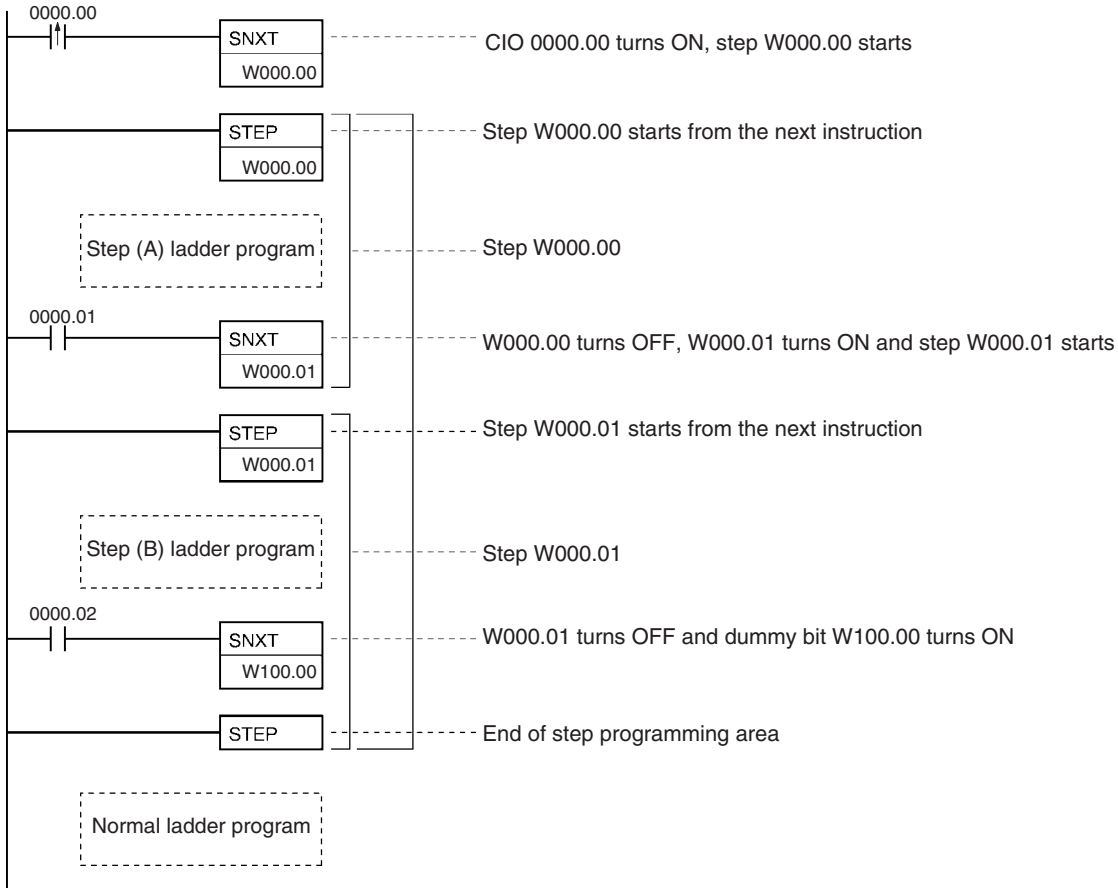
Function	Mnemonic	Name
Sequence Control Instructions	END(001)	END
	IL(002)	INTERLOCK
	ILC(003)	INTERLOCK CLEAR
	JMP(004)	JUMP
	JME(005)	JUMP END
Subroutine Instructions	SBN(092)	SUBROUTINE ENTRY
	RET(093)	SUBROUTINE RETURN



Related Bits

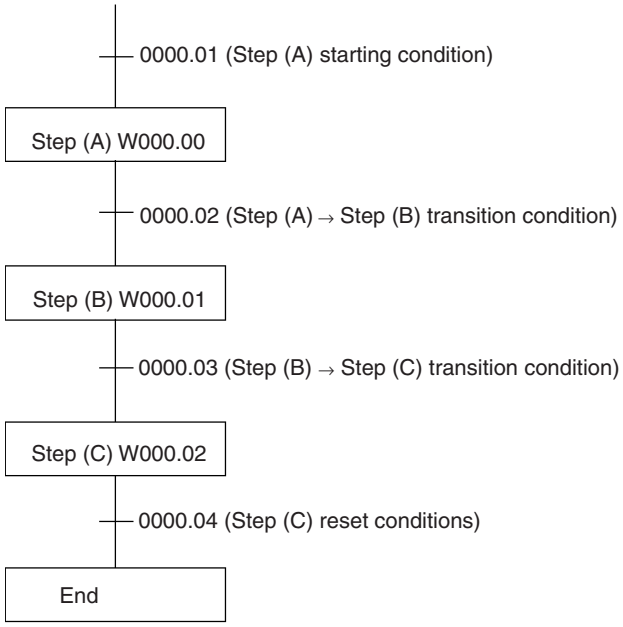
Name	Address	Details
Step Flag	A200.12	ON for one cycle when a step program is started using STEP(008). Can be used to reset timers and perform other processing when starting a new step.

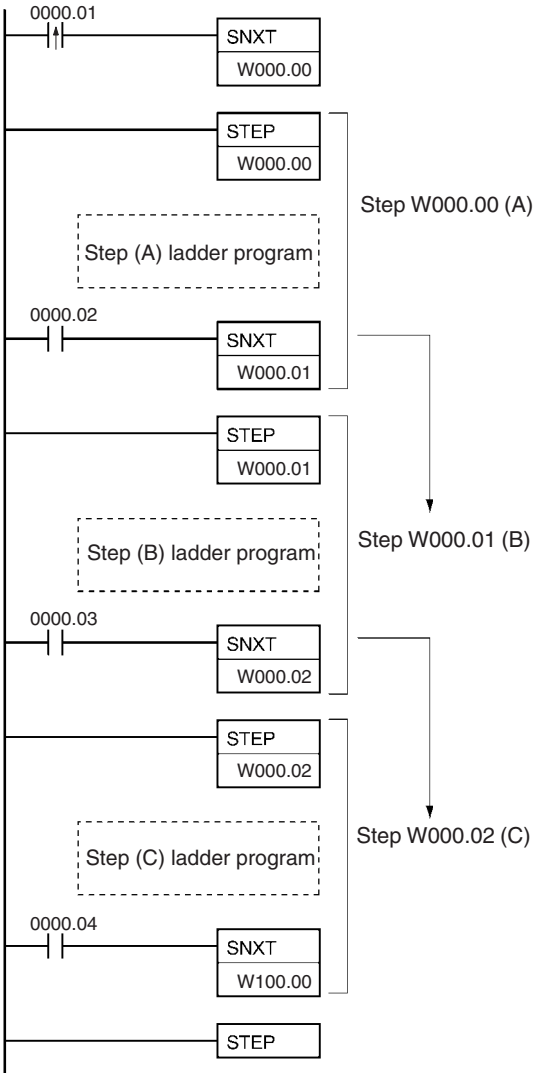

c
SNXT
a
STEP
a
Step a starts when C turns ON
A
A executed
d
SNXT
b
When d turns ON, b starts (A is interlocked)
STEP
b
B
B executed
e
SNXT
(dummy bit)
e turns ON (B is interlocked)
STEP
End of step programming area
Normal ladder program
Returns to normal ladder program



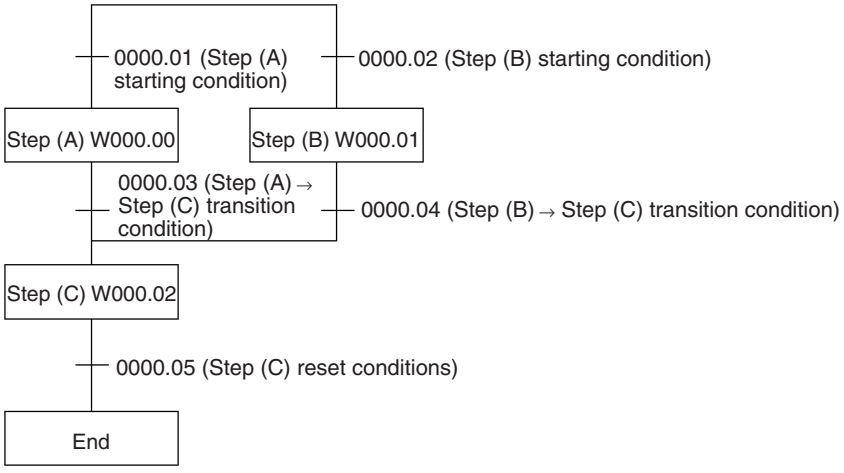
Examples

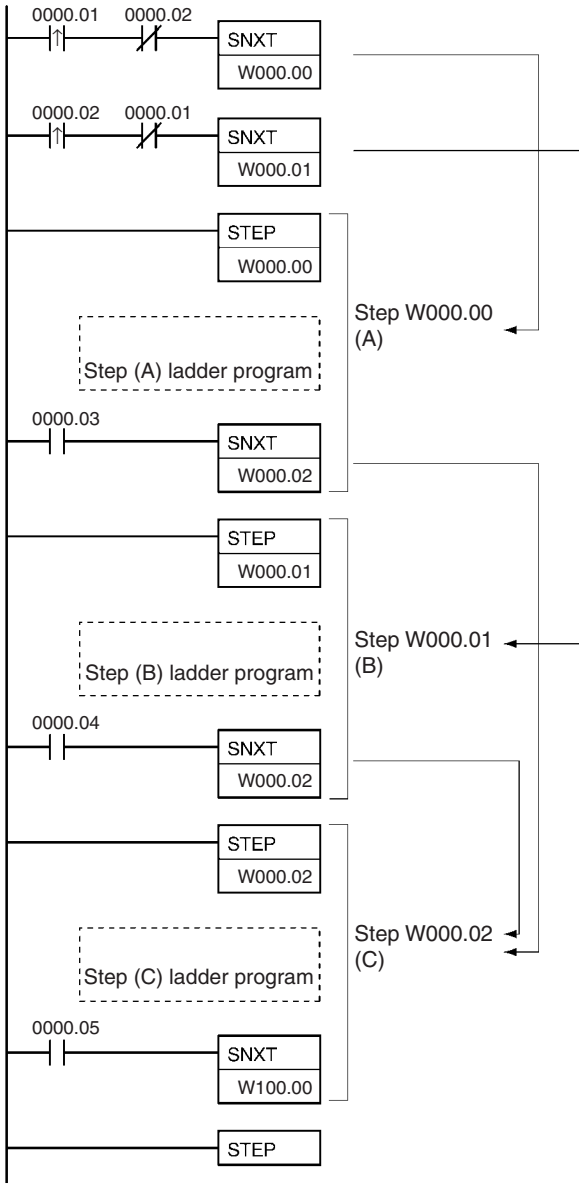
Sequential Control



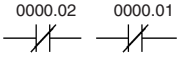


Branching Control



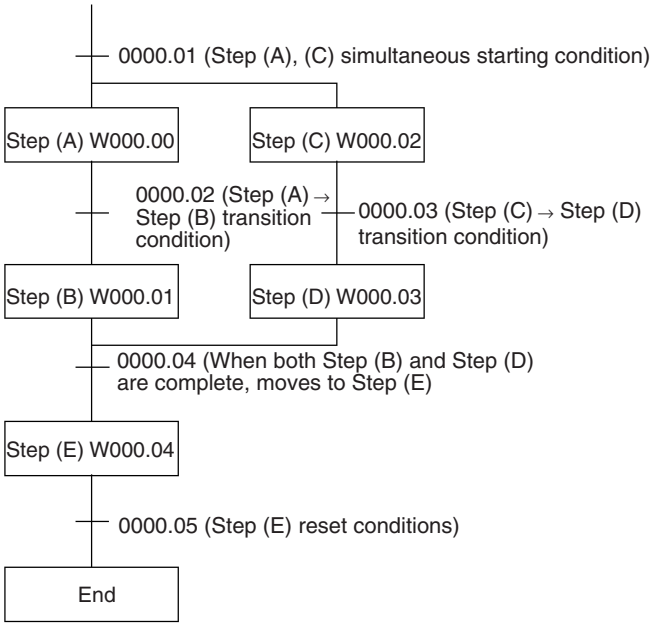


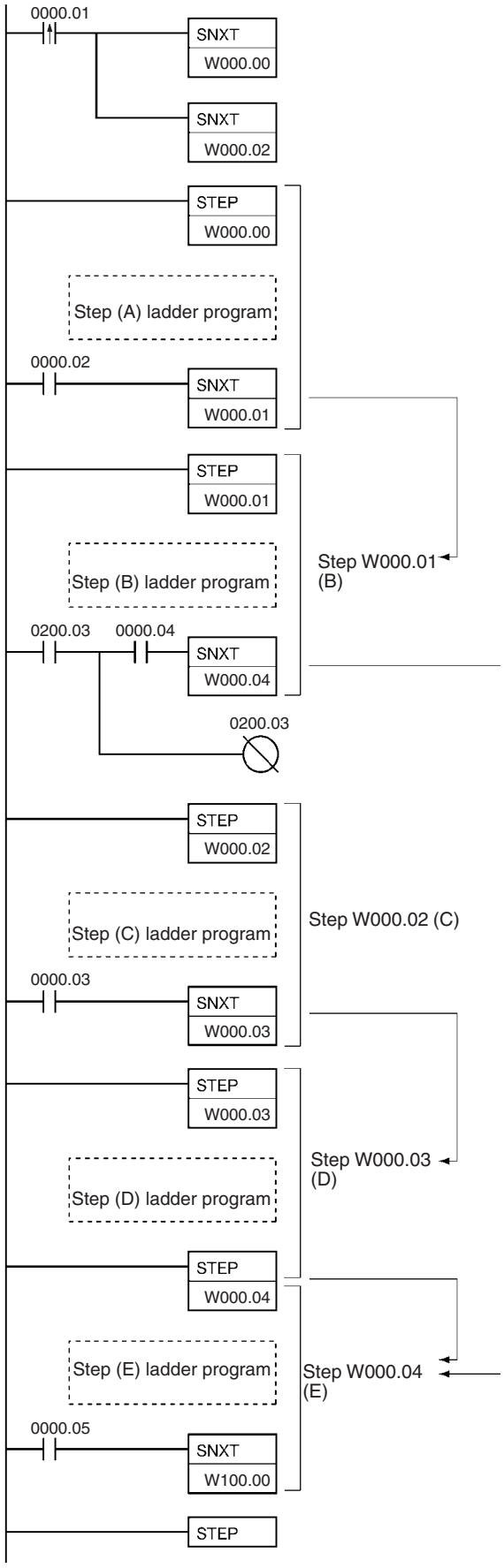
The above programming is used when steps A and B cannot be executed simultaneously. For simultaneous execution of A and B, delete the execution conditions illustrated below.



Note In the above example, where SNXT(009) is executed for W000.02, the branching moves onto the next steps even though the same control bit is used twice. This is not picked up as an error in the program check using the CX-Programmer. A duplicate bit error will only occur in a step ladder program only when a control bit in a step instructions is also used in the normal ladder diagram.

Parallel Control



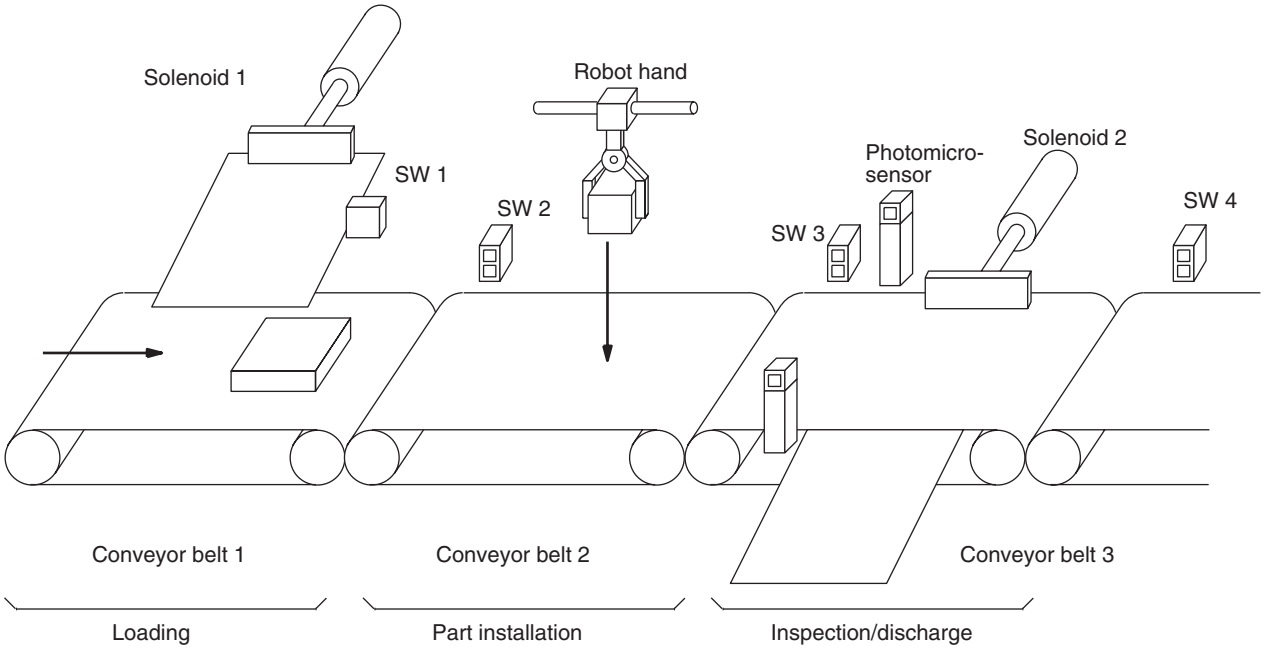

0000.01
SNXT
W000.00
SNXT
W000.02
STEP
W000.00
Step (A) ladder program
0000.02
SNXT
W000.01
STEP
W000.01
Step (B) ladder program
Step W000.01 (B)
0200.03
0000.04
SNXT
W000.04
0200.03
STEP
W000.02
Step (C) ladder program
Step W000.02 (C)
0000.03
SNXT
W000.03
STEP
W000.03
Step (D) ladder program
Step W000.03 (D)
STEP
W000.04
Step (E) ladder program
Step W000.04 (E)
0000.05
SNXT
W100.00
STEP

Application Examples

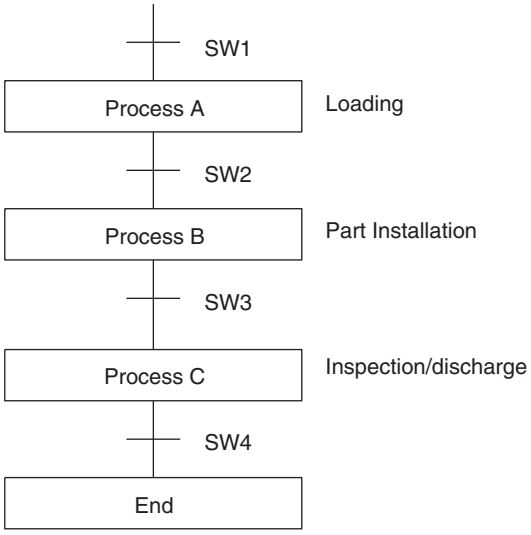
The following three examples demonstrate the three types of execution control possible with step programming. *Example 1* demonstrates sequential execution; *Example 2*, branching execution; and *Example 3*, parallel execution.

Example 1: Sequential Execution

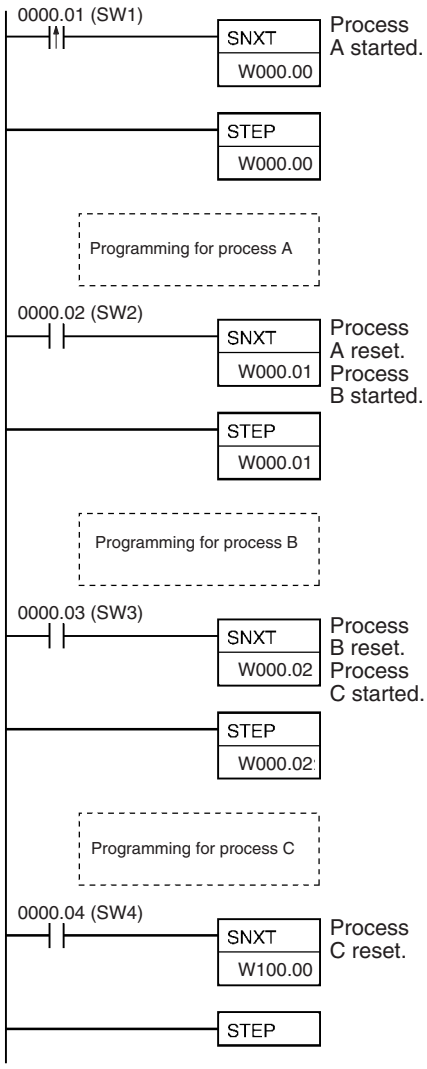
The following process requires that three processes, loading, part installation, and inspection/discharge, be executed in sequence with each process being reset before continuing on the next process. Various sensors (SW1, SW2, SW3, and SW4) are positioned to signal when processes are to start and end.



The following diagram demonstrates the flow of processing and the switches that are used for execution control.



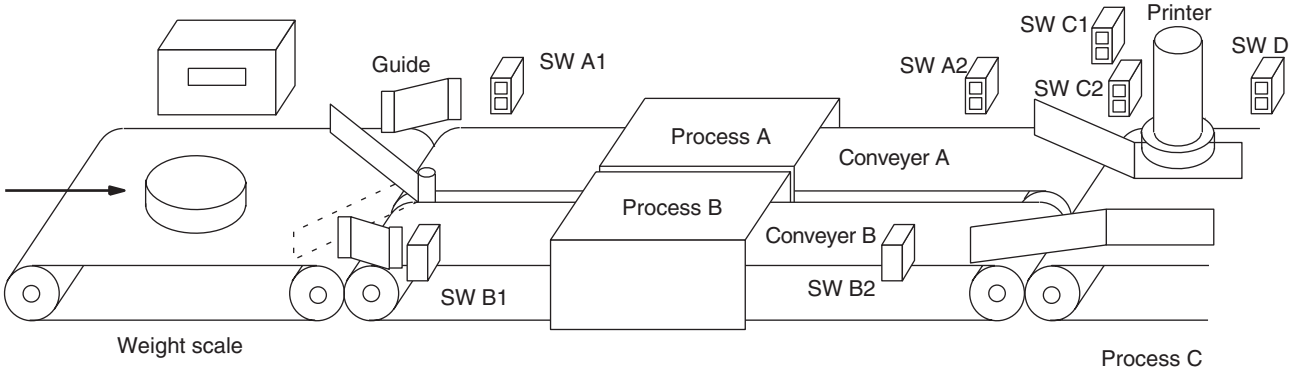
The program for this process, shown below, utilizes the most basic type of step programming: each step is completed by a unique SNXT(009) that starts the next step. Each step starts when the switch that indicates the previous step has been completed turns ON.



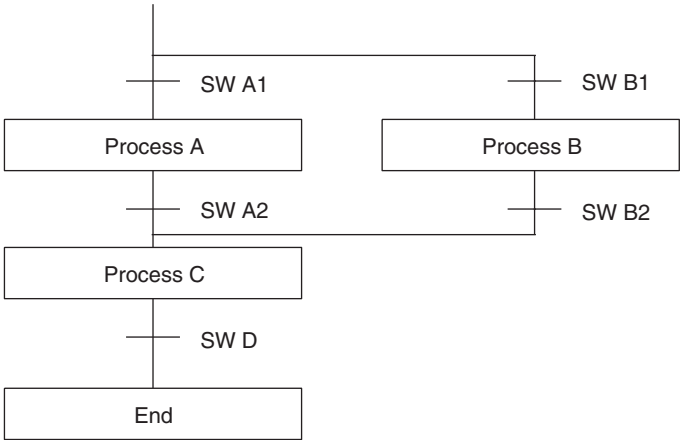
Address	Instruction	Operands
000000	@LD	0000.01
000001	SNXT(009)	W000.00
000002	STEP(008)	W000.00
	Process A	
000100	LD	0000.02
000101	SNXT(009)	W000.01
000102	STEP(008)	W000.01
	Process B	
000200	LD	0000.03
000201	SNXT(009)	W000.02
000202	STEP(008)	W000.02
	Process C	
000300	LD	0000.04
000301	SNXT(009)	W100.00
000302	STEP(008)	---

Example 2: Branching Execution

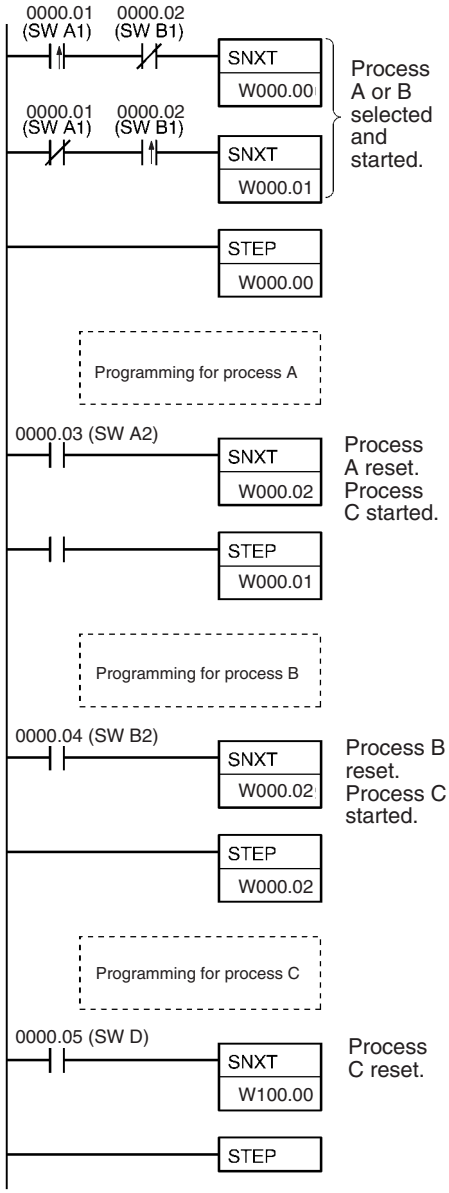
The following process requires that a product is processed in one of two ways, depending on its weight, before it is printed. The printing process is the same regardless of which of the first processes is used. Various sensors are positioned to signal when processes are to start and end.



The following diagram demonstrates the flow of processing and the switches that are used for execution control. Here, either process A or process B is used depending on the weight of the product.



The program for this process, shown below, starts with two SNXT(009) instructions that start processes A and B. Because of the way CIO 0000.01 (SW A1) and CIO 0000.02 (SW B1) are programmed, only one of these will be executed with an ON execution condition to start either process A or process B. Both of the steps for these processes end with a SNXT(009) that starts the step (process C).

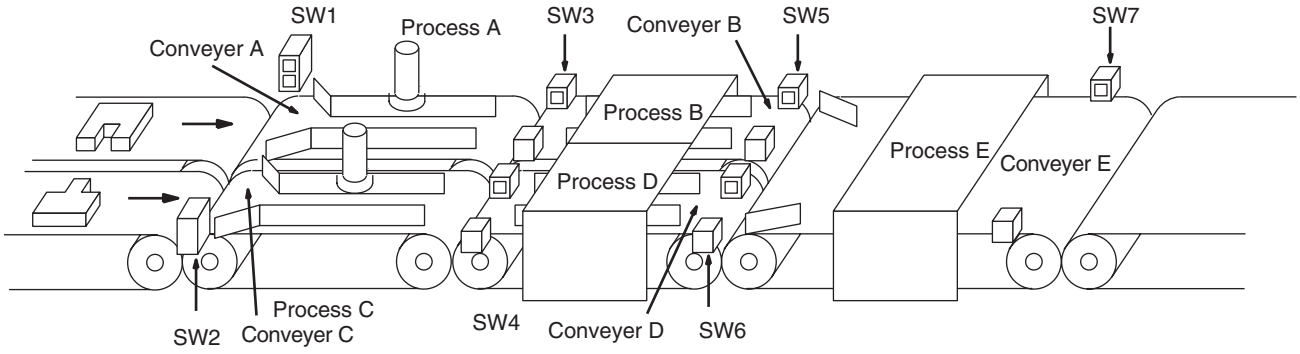


Address	Instruction	Operands
000000	@LD	0000.01
000001	AND NOT	0000.02
000002	SNXT(009)	W000.00
000003	LD NOT	0000.01
000004	@AND	0000.02
000005	SNXT(009)	W000.01
000006	STEP(008)	W000.00
Process A		
000100	LD	0000.03
000101	SNXT(009)	W000.02
000102	STEP(008)	W000.01
Process B		
000200	LD	0000.04
000201	SNXT(009)	W000.02
000202	STEP(008)	W000.02
Process C		
000300	LD	0000.05
000301	SNXT(009)	W100.00
000302	STEP(008)	---

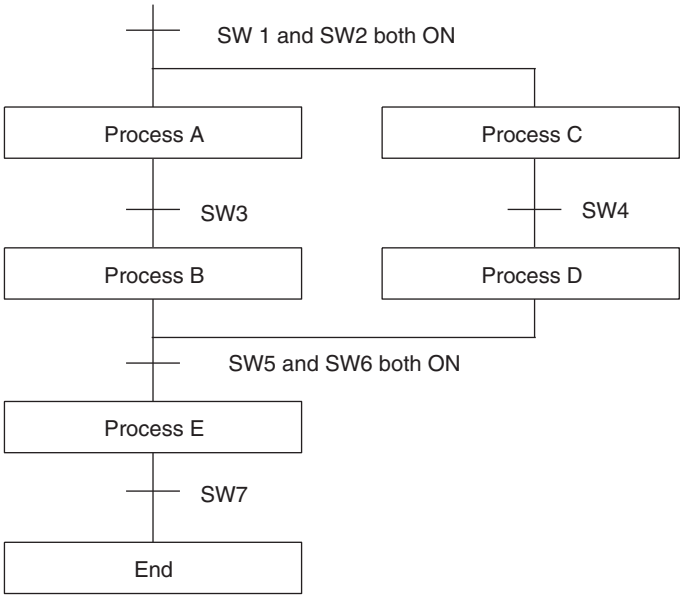
Note In the above programming, W000.02 is used in two SNXT(009) instructions. This will not produce a duplication error during the program check.

Example 3: Parallel Execution

The following process requires that two parts of a product pass simultaneously through two processes each before they are joined together in a fifth process. Various sensors are positioned to signal when processes are to start and end.

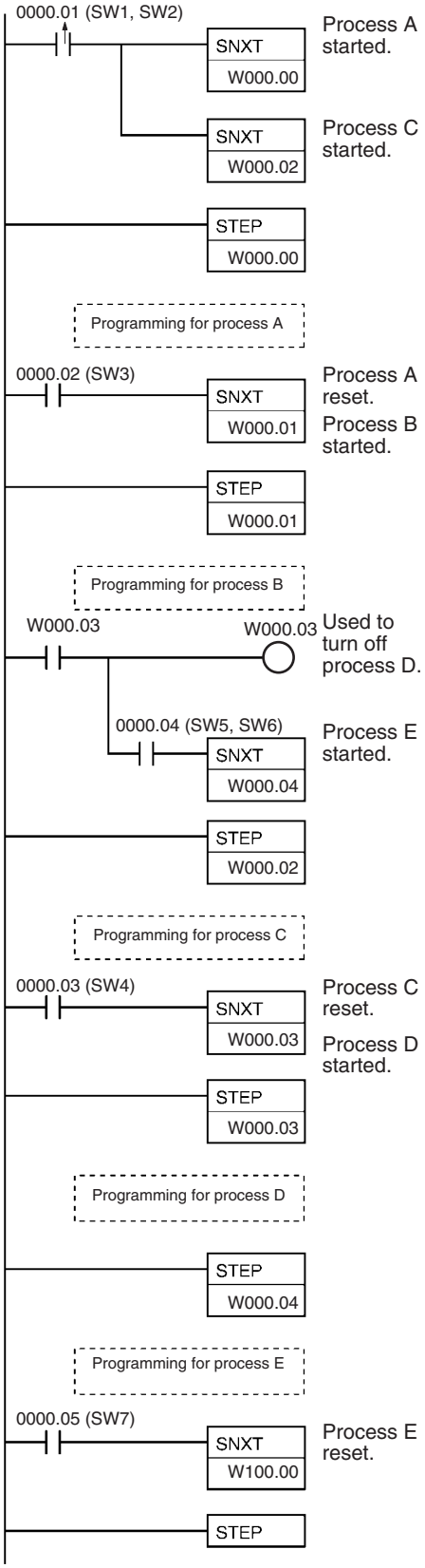


The following diagram demonstrates the flow of processing and the switches that are used for execution control. Here, process A and process C are started together. When process A finishes, process B starts; when process C finishes, process D starts. When both processes B and D have finished, process E starts.



The program for this operation, shown below, starts with two SNXT(009) instructions that start processes A and C. These instructions branch from the same instruction line and are always executed together, starting steps for both A and C. When the steps for both A and C have finished, the steps for process B and D begin immediately.

When both process B and process D have finished (i.e., when SW5 and SW6 turn ON), processes B and D are reset together by the SNXT(009) at the end of the programming for process B. Although there is no SNXT(009) at the end of process D, the control bit for it is turned OFF by executing SNXT(009) W000.04. This is because the OUT for bit W000.03 is in the step reset by SNXT(009) W000.04, i.e., W000.03 is turned OFF when SNXT(009) W000.04 is executed. Process B is thus reset directly and process D is reset indirectly before executing the step for process E.



Address	Instruction	Operands
000000	@LD	0000.01
000001	SNXT(009)	W000.00
000002	SNXT(009)	W000.02
000003	STEP(008)	W000.00
	Process A	
000100	LD	0000.02
000101	SNXT(009)	W000.01
000102	STEP(008)	W000.01
	Process B	
000200	LD	W000.03
000201	OUT	W000.03
000202	AND	0000.04
000203	SNXT(009)	W000.04
000204	STEP(008)	W000.02
	Process C	
000300	LD	0000.03
000301	SNXT(009)	W000.03
000302	STEP(008)	W000.03
	Process D	
000400	STEP(008)	W000.04
	Process E	
000500	LD	0000.05
000501	SNXT(009)	W100.00
000502	STEP(008)	---

3-22 I/O Refresh Instruction

This section describes the instruction used to refresh I/O.

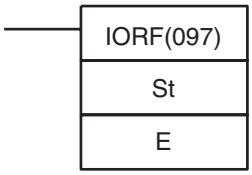
Instruction	Mnemonic	Function code	Page
I/O REFRESH	IORF	097	580

3-22-1 I/O REFRESH: IORF(097)

Purpose

Refreshes the specified I/O words.

Ladder Symbol



St: Starting word

E: End word

Variations

Variations		
	Executed Each Cycle for ON Condition	IORF(097)
	Executed Once for Upward Differentiation	@IORF(097)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

St: Starting Word
I/O Bit Area: CIO 2960 and CIO 2961
Basic I/O Unit I/O Bit Area: CIO 0000 to CIO 0019

E: End Word
I/O Bit Area: CIO 2960 and CIO 2961
Basic I/O Unit I/O Bit Area: CIO 0000 to CIO 0019

Note St and E must be in the same memory area.

Operand Specifications

Area	St	E
CIO Area	CIO 0000 to CIO 0019 CIO 2960 to CIO 2961	
Auxiliary Area	---	
Special Bit Area	---	
Timer Area	---	
Counter Area	---	
DM Area	---	
Indirect DM addresses in binary	---	
Indirect DM addresses in BCD	---	
Constants	---	
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

IORF(097) refreshes the I/O words between St and E.

The following words can be refreshed: I/O Bit Area (CIO 2960 and CIO 2961) and Basic I/O Unit I/O Bit Area (CIO 0000 to CIO 0019).

Words allocated to Special I/O Units cannot be refreshed with IORF(097).

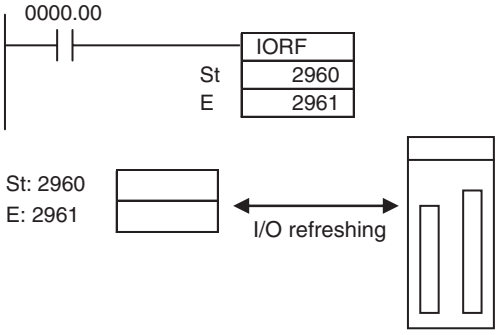
IORF(097) can be used in interrupt tasks to enable high-speed processing of the specified I/O words.

Flags

Name	Label	Operation
Error Flag	ER	ON if St is greater than E. ON if St and E are in different memory areas. OFF in all other cases.

Examples

The following example shows how to refresh the two words CIO 2960 and CIO 2961 when CIO 0000.00 turns ON.



3-23 Serial Communications Instructions

This section describes instructions used for serial communications.

Instruction	Mnemonic	Function code	Page
TRANSMIT	TXD	236	582
RECEIVE	RXD	235	587
CHANGE SERIAL PORT SETUP	STUP	237	592

3-23-1 Serial Communications

The TXD(236) and RXD(235) instructions send and receive data in no-protocol (custom) communications with an external device through a serial port on the Coordinator Module.

Instructions	Communications frames	Function
TXD(236) and RXD(235)	Any of the following can be used. No Start or End Code: Data Start and End Code: ST | Data | ED Only Start Code: ST | Data CR+LF End Code: Data | CR | LF Only End Code: Data | ED Start and CR+LF End Code: ST | Data | CR | LF	Sends or receives data in one direction only. A send delay can be set.

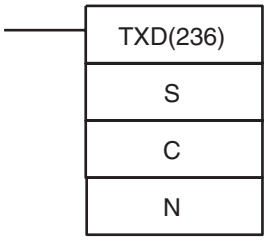
Instructions	Mode	Communications ports
TXD(236) and RXD(235)	No-protocol (custom)	Serial port in FQM1-CM002 FQM1-CM002 TXD(236)/ RXD(235) RXD(235) TXD(236)

3-23-2 TRANSMIT: TXD(236)

Purpose

Outputs the specified number of bytes of data from the RS-232C port or RS-422A port on the Coordinator Module.

Ladder Symbol



S: First source word

C: Control word

N: Number of bytes
0000 to 0100 hex (0 to 256)

Variations

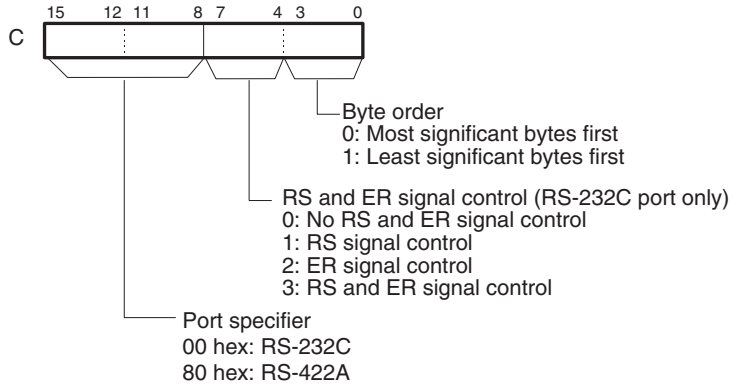
Variations		
	Executed Each Cycle for ON Condition	TXD(236)
	Executed Once for Upward Differentiation	@TXD(236)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

The contents of the control word, C, is as shown below.



Operand Specifications

Area	S	C	N
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A000 to A959		
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---	Specified values only	#0000 to #0100 (binary)
Data Resisters	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

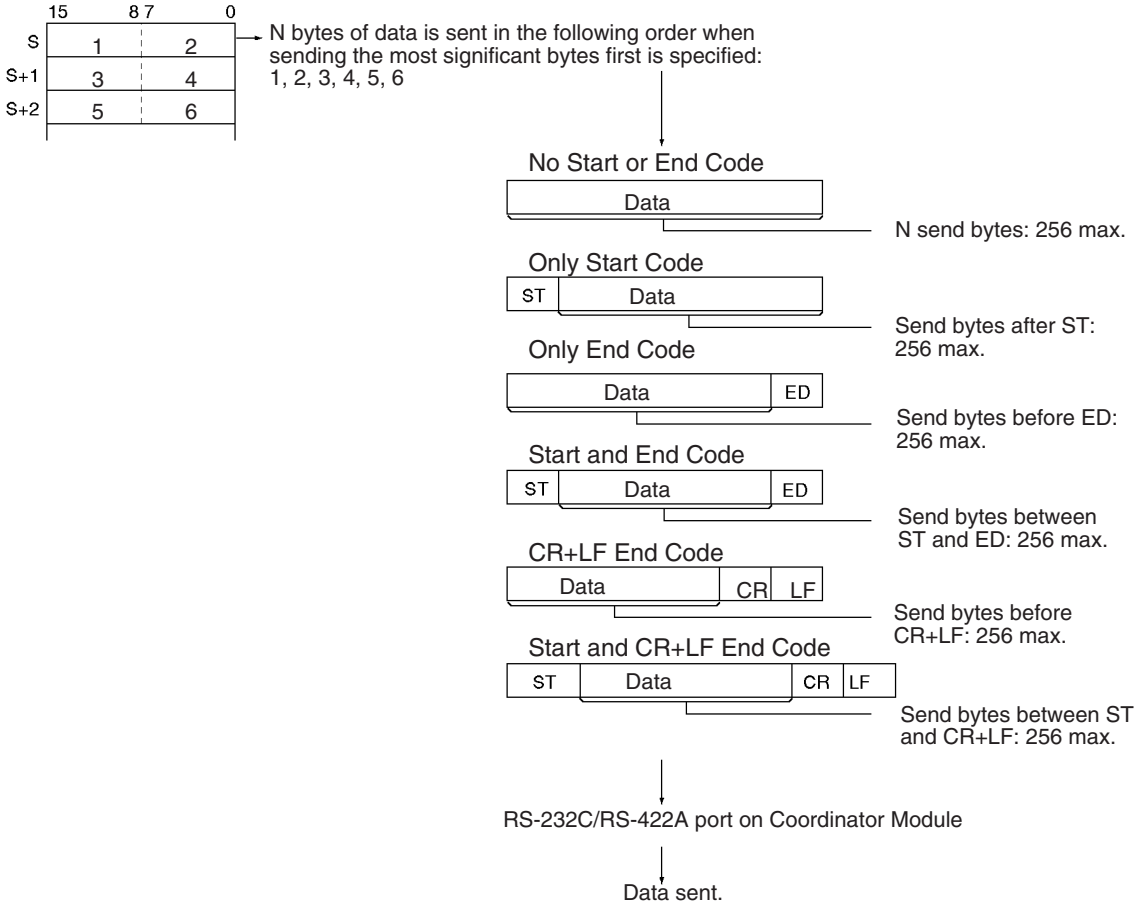
Description

TXD(236) reads N bytes of data from words beginning with S and outputs the raw data in no-protocol mode from the RS-232C port or RS-422A port. (The output port is specified with bits 8 to 15 of C.)

The start and end codes specified for no-protocol mode are added to the data before the data is output. The start and end codes are specified in the System Setup.

Data can be sent only when the port's Send Ready Flag is ON. The Send Ready Flags are A392.05 and A318.09.

The following diagram shows the order in which data is sent and the contents of the send frame for various start and end code settings.



Flags

Name	Label	Operation
Error Flag	ER	ON if the no-protocol mode is not set in the System Setup. ON if the value of C is not within range. ON if the value for N is not between 0000 and 0100 hex. ON if a send is attempted when the Send Ready Flag (A392.05/A318.09) is OFF. OFF in all other cases.

Precautions

TXD(236) can be used only for the Coordinator Modules's RS-232C port or RS-422A port. In addition, the port must be set to no-protocol mode in the System Setup.

The following send-message frame format can be set in the System Setup.

- Start code: None or 00 to FF hex.
- End code: None, CR+LF, or 00 to FF hex.

The data will be sent with any start and/or end codes specified in the System Setup. If start and end codes are specified, the codes will be added to the send data (N). Even in this case, the maximum number of bytes that can be specified for N is 256 bytes.

Data can be sent only when the port's Send Ready Flag (A392.05/A318.09) is ON.

Data is sent in the order specified in C.

Nothing will be sent if 0 is specified for N.

If RS signal control is specified in C, bit 15 of S will be used as the RS signal.

If ER signal control is specified in C, bit 15 of S will be used as the ER signal.

If RS and ER signal control is specified in C, bit 15 of S will be used as the RS signal and bit 14 of S will be used as the ER signal.

If 1, 2, or 3 hex is specified for RS and ER signal control in C, TXD(236) will be executed regardless of the status of the Send Ready Flag (A392.05).

An error will occur and the Error Flag will turn ON in the following cases.

- If no-protocol mode is not set for the port in the System Setup
- If the value of C is not within range
- If the value for N is not between 0000 and 0100 hex
- If a send is attempted when the Send Ready Flag (A392.05/A318.09) is OFF.

Note The timing of sending data can be coordinated with the receiving device by setting a send delay.

Related Flags and Words

The following System Setup settings and Auxiliary Area flag can be used as required when executing TXD(236).

System Setup Settings

Name	Description	Settings
No-protocol Mode Start Code	Specifies whether to use a start code in the frame format for no-protocol communications.	0: None (default) 1: Use start code (00 to FF hex)
No-protocol Mode End Code	Specifies whether to use an end code in the frame format for no-protocol communications.	0: None (default) 1: Use end code (00 to FF hex or CF+LF)
No-protocol Mode Send Delay	Specifies whether to delay sending data after execution of the instruction until sending the data from the port.	0 to 99,990 ms decimal (in 10-ms units) Default: 0 ms

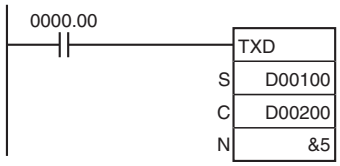
Auxiliary Area

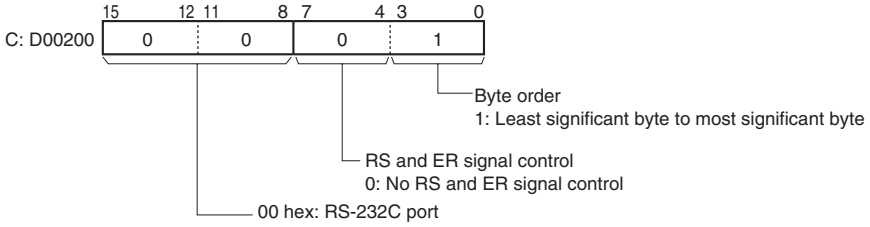
Send Ready Flags

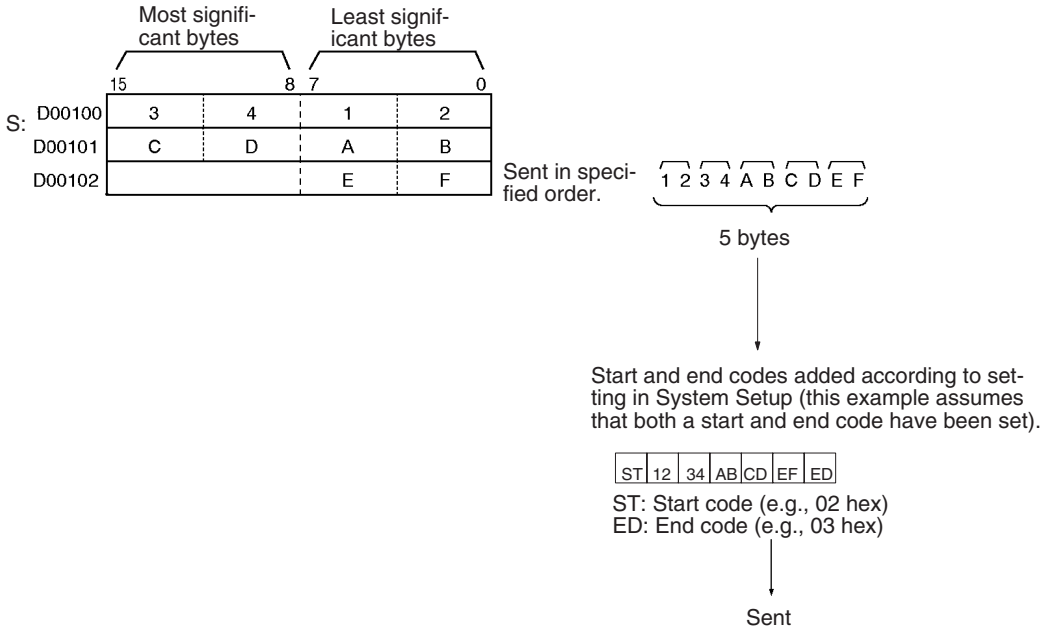
Name	Address	Contents
RS-232C Port Send Ready Flag	A392.05	ON when data can be sent in the no-protocol mode.
RS-422A Port Send Ready Flag	A318.09	

Example: Sending Data

When CIO 0000.00 and the RS-232C port Send Ready Flag (A410.09) are ON in the following example, 5 bytes of data are sent from the RS-232C port on the Coordinator Module.

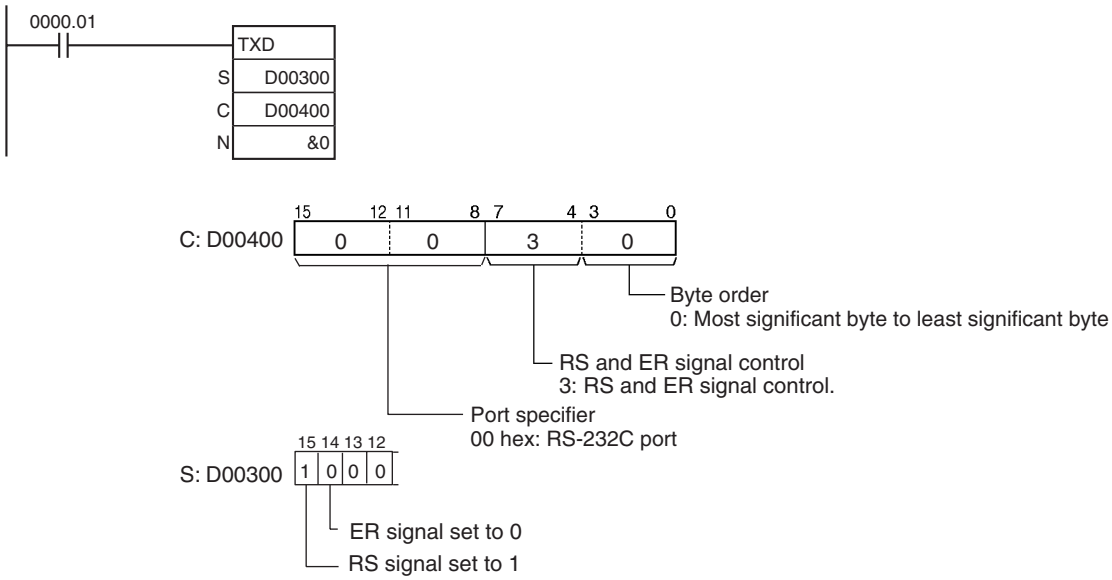






Example: Performing Signal Control

When CIO 0000.01 and the RS-232C port Send Ready Flag (A410.09) are ON in the following example, the RS signal is set according to the status of D00300 bit 15 and the ER signal is set according to the status of D00300 bit 14.

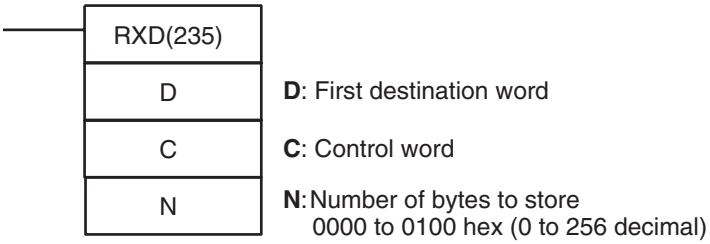


3-23-3 RECEIVE: RXD(235)

Purpose

Reads the specified number of bytes of data from the RS-232C port or RS-422A port on the Coordinator Module.

Ladder Symbol



Variations

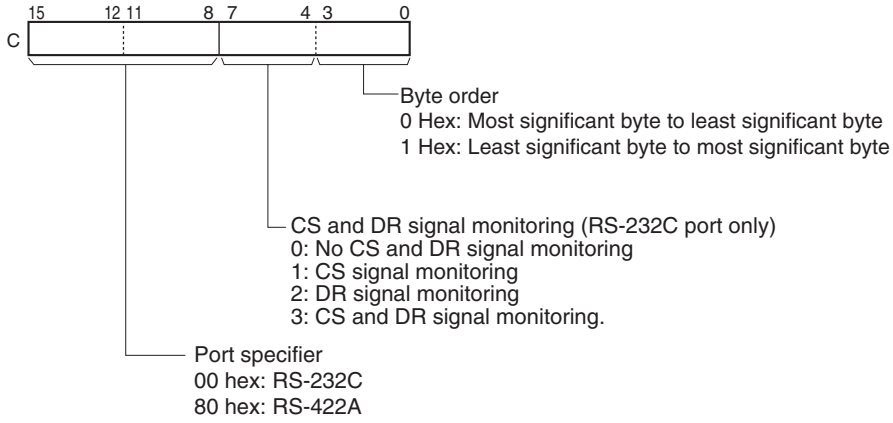
Variations		
	Executed Each Cycle for ON Condition	RXD(235)
	Executed Once for Upward Differentiation	@RXD(235)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

The contents of the control word, C, is as shown below.



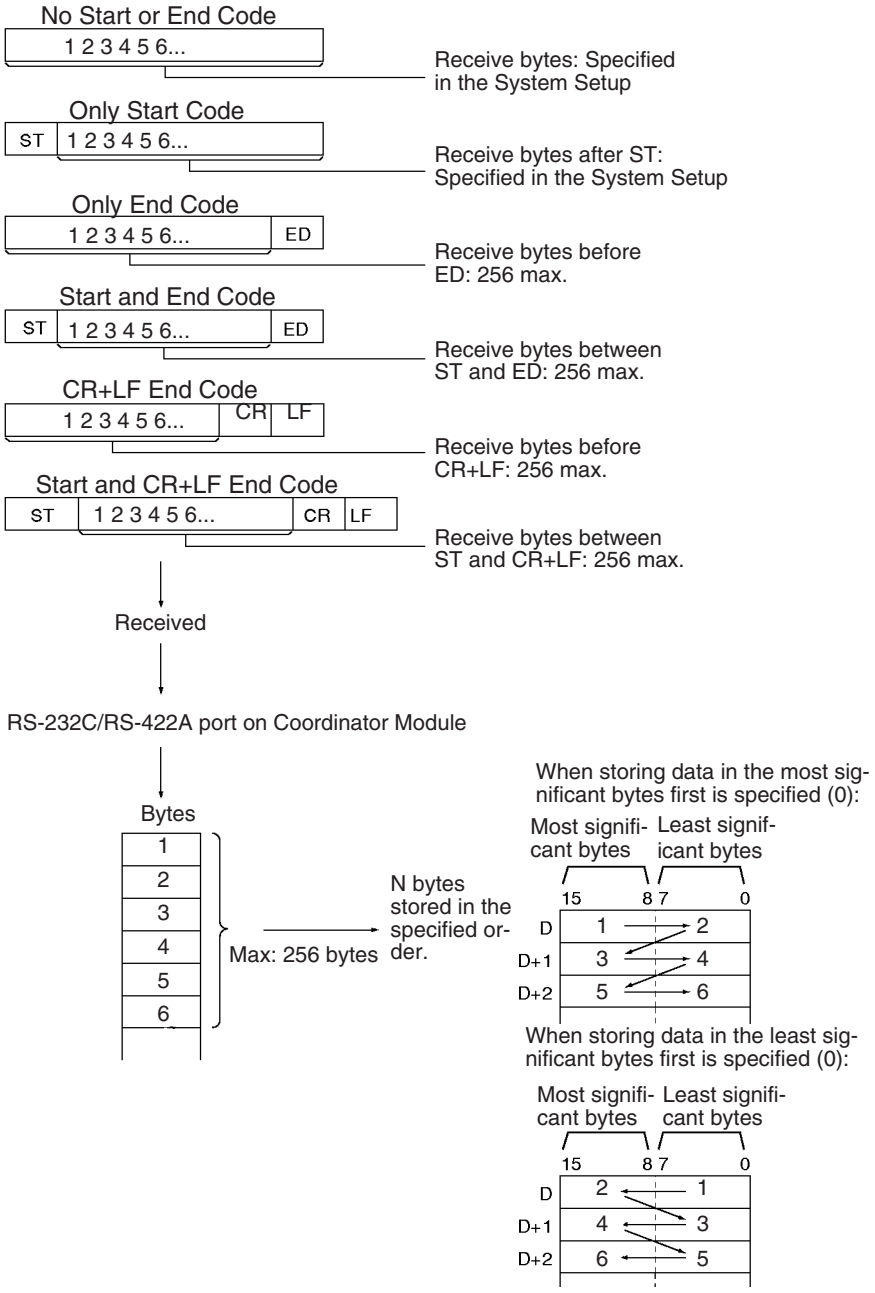
Operand Specifications

Area	D	C	N
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	A448 to A959	A000 to A959	
Timer Area	T0000 to T0255		
Counter Area	C0000 to C0255		
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---	Specified values only	#0000 to #0100 (binary)
Data Resisters	---	DR0 to DR15	
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

RXD(235) reads data that has been received in no-protocol mode at the RS-232C port or RS-422A port on the Coordinator Module (the port is specified with bits 8 to 15 of C) and stores N bytes of data in words beginning from D. If N bytes of data has not been received at the port, then only the data that has been received will be stored.

The following diagram shows the order in which data is received and the contents of the receive frame for various settings.



Flags

Name	Label	Operation
Error Flag	ER	ON if the no-protocol mode is not set in the System Setup. ON if the value of C is not within range. ON if the value for N is not between 0000 and 0100 hex. OFF in all other cases.

Precautions

RXD(235) can be used only for the RS-232C port or RS-422A port on the Coordinator Module. In addition, the port must be set to no-protocol mode.

The following receive message frame format can be set in the System Setup.

- Start code: None or 00 to FF hex
- End code: None, CR+LF, or 00 to FF hex. If no end code is specified, the number of bytes to receive is set from 00 to FF hex (1 to 256 decimal; 00 specifies 256 bytes).

If the number of bytes is specified in the System Setup, the Reception Completed Flag (A392.06/A318.10) will turn ON when the specified number of bytes has been received. When the Reception Completed Flag turns ON, the number of bytes in the Reception Counter (A393/A319) will have the same value as the number of receive bytes specified in the System Setup. If more bytes are received than specified, the Reception Overflow Flag (A392.07/A318.11) will turn ON.

If an end code is specified in the System Setup, the Reception Completed Flag (A392.06/A318.10) will turn ON when the end code is received or when 256 bytes of data have been received. If more data is received after the Reception Completed Flag (A392.06/A318.10) turns ON, the Reception Overflow Flag (A392.07/A318.11) will turn ON.

When RXD(235) is executed, data is stored in memory starting at D. Once the data has been stored, the Reception Completed Flag (A392.06/A318.10) will turn ON (OFF if the Reception Overflow Flag (A392.07/A318.11) is ON) and the Reception Counter (A393/A319) will be cleared to 0.

If the RS-232C Port Restart Bit (A526.00) or the RS-422A Port Restart Bit (A526.07) is turned ON, the Reception Completed Flag (A392.06/A318.10) will be turned OFF (even if the Reception Overflow Flag is ON), and the Reception Counter (A393/A319) will be cleared to 0.

Data will be stored in memory in the order specified in C.

If 0 is specified for N, the Reception Completed Flag (A392.06/A318.10) and Reception Overflow Flag (A392.07/A318.11) will be turned OFF, the Reception Counter (A393/A319) will be cleared to 0, and nothing will be stored in memory.

If CS signal monitoring is specified in C, the status of the CS signal will be stored in bit 15 of D.

If DR signal monitoring is specified in C, the status of the DR signal will be stored in bit 15 of D.

If CS and DR signal monitoring is specified in C, the status of the CS signal will be stored in bit 15 of D and the status of the DR signal will be stored in bit 14 of D.

Receive data will not be stored if CS or DR signal monitoring is specified.

If 1, 2, or 3 hex is specified for CS and DR signal control in C, RXD(235) will be executed regardless of the status of the Reception Completed Flag (A392.06/A318.10).

An error will occur and the Error Flag will turn ON in the following cases.

- If no-protocol mode is not set for the port in the System Setup
- If the value of C is not within range
- If the value for N is not between 0000 and 0100 hex

Note Further data cannot be received if RXD(235) has not completed reading the current data. Always execute RXD(235) after receiving data and before data is received again.

Related Flags and Words

The following System Setup settings and Auxiliary Area flag can be used as required when executing RXD(235).

System Setup Settings

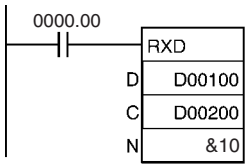
Name	Description	Settings
No-protocol Mode Start Code	Specifies whether to use a start code in the frame format for no-protocol communications.	0: None (default) 1: Use start code (00 to FF hex)
No-protocol Mode End Code	Specifies whether to use an end code in the frame format for no-protocol communications.	0: None (default) 1: Use end code (00 to FF hex or CF+LF)
No-protocol Mode Number of bytes of Data	Specifies the number of bytes to receive when an end code is not used.	00 hex: 256 bytes 01 to FF hex: 1 to 255 bytes

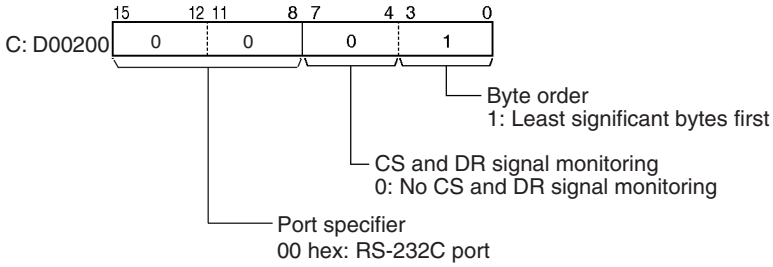
Auxiliary Area Flags

Name	Address	Contents
Reception Completed Flag	A392.06	ON when no-protocol reception is completed. Number of Receive Bytes Specified: The flag will turn ON when the specified number of bytes has been received. End Code Specified: The flag will turn ON when the end code is received or when 256 bytes have been received.
Reception Overflow Flag	A392.07	ON when more than the expected number of receive bytes has been received in no-protocol mode. Number of Receive Bytes Specified: The flag will turn ON when more data is received after reception was completed but before the received data is read from the buffer with RXD(235). End Code Specified: The flag will turn ON when more data is received after receiving an end code but before the received data is read from the buffer with RXD(235), or when 257 or more bytes of data are received without an end code.
Reception Counter	A393	Counts in hexadecimal the number of bytes received in no-protocol mode (0 to 256 decimal).
Reception Completed Flag	A318.10	ON when no-protocol reception is completed. Number of Receive Bytes Specified: The flag will turn ON when the specified number of bytes has been received. End Code Specified: The flag will turn ON when the end code is received or when 256 bytes have been received.
Reception Overflow Flag	A318.11	ON when more that the expected number of receive bytes has been received in no-protocol mode. Number of Receive Bytes Specified: The flag will turn ON when more data is received after reception was completed but before the received data was not read from the buffer with RXD(235). End Code Specified: The flag will turn ON when more data is received after receiving an end code but before the received data is read from the buffer with RXD(235), or when 257 or more bytes of data are received without an end code.
Reception Counter	A319	Counts in hexadecimal the number of bytes received in no-protocol mode (0 to 256 decimal).

Examples

When CIO 0000.00 is ON in the following example, data is received from the RS-232C port and 10 bytes of data are stored starting in D00100.





This example assumes that both a start and end code have been specified in the System Setup.

ST	31	32	33	34	41	42	43	44	45	46	ED

ST: Start code (e.g., 02 hex)
ED: End code (e.g., 03 hex)

Stored

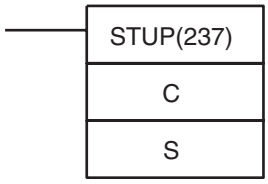
	Most significant bytes		Least significant bytes	
	15	8	7	0
D: D00100	3	2	3	1
D00101	3	4	3	3
D00102	4	2	4	1
D00103	4	4	4	3
D00104	4	6	4	5

3-23-4 CHANGE SERIAL PORT SETUP: STUP(237)

Purpose

Changes the communications parameters of a serial port on the FQM1-CM002 Coordinator Module. STUP(237) thus enables the protocol mode to be changed during FQM1 operation.

Ladder Symbol



C: Control word (port)

S: First source word

Variations

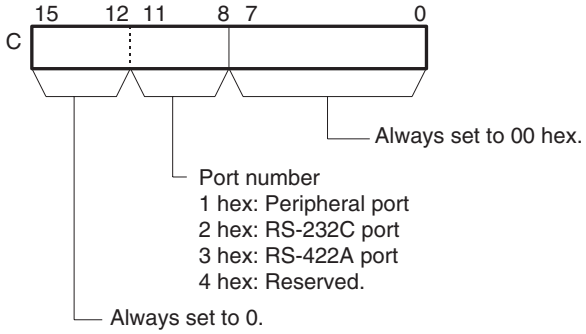
Variations		
	Executed Each Cycle for ON Condition	STUP(237)
	Executed Once for Upward Differentiation	@STUP(237)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	Not allowed

Operands

The contents of the control word, C, are as shown below.



Operand Specifications

Area	C	S
CIO Area	CIO 0000 to CIO 6143	CIO 0000 to CIO 6134
Work Area	W000 to W255	W000 to W246
Auxiliary Bit Area	A000 to A959	A000 to A950
Timer Area	T0000 to T0255	T0000 to T0246
Counter Area	C0000 to C0255	C0000 to C0246
DM Area	D00000 to D32767	D00000 to D32758
Indirect DM addresses in binary	@ D00000 to @ D32767	
Indirect DM addresses in BCD	*D00000 to *D32767	
Constants	Specified values only	#0000
Data Resisters	DR0 to DR15	---
Index Registers	---	
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15	

Description

STUP(237) writes 10 words of data from S to S+9 to the communications setup area of the specified port, as shown in the following table. When the constant #0000 is designated for S, the communications settings of the corresponding port will be set to the default settings.

Unit address	Module	Port No.	Serial port	Serial port communications setup area
00 hex	Coordinator Module	1 hex	Peripheral port	Communications parameters for the peripheral port in the System Setup
		2 hex	RS-232C port	Communications parameters for the RS-232C port in the System Setup
		3 hex	RS-422A port	Communications parameters for the RS-422A port in the System Setup

When STUP(237) is executed, the corresponding Port Settings Changing Flag (A619.01, A619.02, or A318.15) will turn ON. The flag will remain ON until changing the parameters has been completed.

Use STUP(237) to change communications parameter for a port during operation based on specified conditions. For example, STUP(237) can be used as follows:

The RS-422A port is set to Serial Gateway mode normally to send commands from a PT directly to a Servo Driver. When controlling the Servomotor by changing the parameters of the Servo Driver from the ladder program, the RS-422A port is switched to No-protocol mode.

The following communications parameters can be set:
Protocol mode, baud rate, data format, protocol macro transfer method, maximum length of protocol macro communications data, etc.

Note If the power to the FQM1 is cycled after the communications settings are changed with STUP(237), the original settings (i.e., the ones before STUP(237) was executed) will be used when the power turns ON.

Flags

Name	Label	Operation
Error Flag	ER	ON if the values in C are not within range. ON if STUP(237) is executed for a port whose Communications Settings Changing Flag is already ON. ON if STUP(237) is executed in an interrupt task. OFF in all other cases.

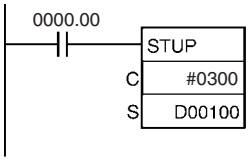
Related Flags and Words

The following flags can be used as required when executing STUP(237). These flags are in the Auxiliary Area.

Name	Address	Contents
Peripheral Port Settings Changing Flag	A619.02	ON when the communications parameters are being changed for the peripheral port.
RS-232C Port Settings Changing Flag	A619.01	ON when the communications parameters are being changed for the RS-232C port.
RS-422A Port Settings Changing Flag	A318.15	ON when the communications parameters are being changed for the RS-422A port.

Examples

When CIO 0000.00 turns ON in the following example, the communications parameters for the RS-422A port are changed to the settings contained in the 10 words from D00100 to D00109. In this example, the setting are changed from the Serial Gateway Mode to the No-protocol Mode.



S: D00100	8	3	0	0	Port setting: Defaults, Protocol mode: 3 hex (no-protocol).
S+1: D00101	0	0	0	0	Baud rate: Default (9,600 bps)
S+2: D00102					
to	to				
S+9: D00109					

Transferred

RS-422A port parameter area

+360	8	3	0	0
+361	0	0	0	0
+362				
to	to			
+367				

3-24 Debugging Instructions

This section describes the instruction used to debug programs.

Instruction	Mnemonic	Function code	Page
TRACE MEMORY SAMPLING	TRSM	045	596

3-24-1 Trace Memory Sampling: TRSM(045)

Purpose

When TRSM(045) is executed, the status of a preselected bit or word is sampled and stored in Trace Memory. TRSM(045) can be used anywhere in the program, any number of times.

Ladder Symbol

Variations

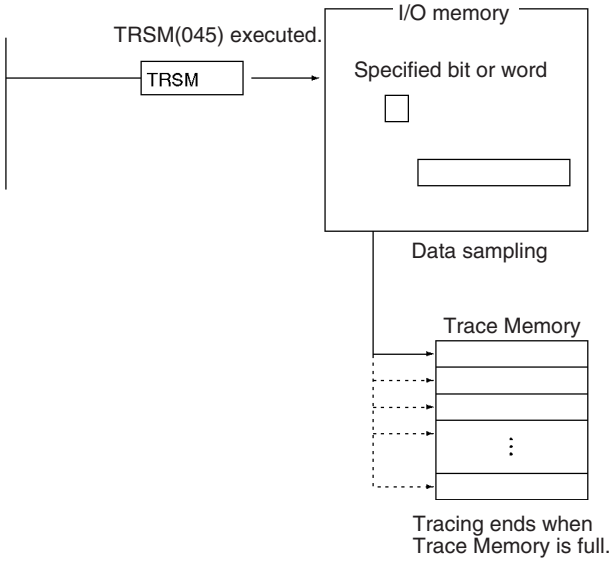
Variations		
	Executed Each Cycle	TRSM(045)
	Executed Once for Upward Differentiation	Not supported
	Executed Once for Downward Differentiation	Not supported

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Description

Before TRSM(045) is executed, the bit or word to be traced must be specified with the CX-Programmer. Each time that TRSM(045) is executed, the current value of the specified bit or word is sampled and recorded in order in Trace Memory. The trace ends when the Trace Memory is full. The contents of Trace Memory can be monitored from the CX-Programmer when necessary.

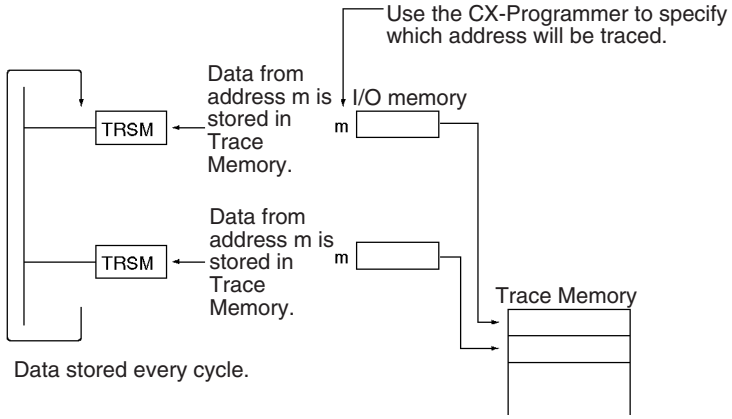


This instruction only indicates when the specified data will be sampled. All other settings and data trace operations are set with the CX-Programmer. The other two ways to control data sampling are sampling at the end of each cycle and sampling at a specified interval (independent of the cycle time).

TRSM(045) does not require an execution condition and is always executed as if it had an ON execution condition. Connect TRSM(045) directly to the left bus bar.

Use TRSM(045) to sample the value of the specified bit or word at the point in the program when the instruction's execution condition is ON. If the instruction's execution condition is ON every cycle, the specified bit or word's value will be stored in Trace Memory every cycle.

It is possible to incorporate two or more TRSM(045) instructions in a program. In this case, the value of the same specified bit or word will be stored in Trace Memory each time that one of the TRSM(045) instructions is executed.



Note Refer to the CX-Programmer Operation Manual (Cat. No. W437) for details on data tracing.

The data-tracing operations performed with the CX-Programmer are summarized in the following list.

1,2,3... 1. Set the following parameters with the CX-Programmer.
   a) Set the address of the bit or word to be traced.
   b) Set the trigger condition. One of the three following conditions can control when data stored into Trace Memory is valid.
      i) The Trace Start Bit goes from OFF to ON.
      ii) A specified bit goes from OFF to ON.
      iii) The value of a specified word matches the set value.
   c) Set the sampling interval to "TRSM" for sampling at the execution of TRSM(045) in the program.
   d) Set the delay.
2. When the Sampling Start Bit (A508.15) is turned from OFF to ON with the CX-Programmer, the specified data will begin being sampled each time that TRSM(045) is executed and the sampled data will be stored in Trace Memory. The Trace Busy Flag (A508.13) will be turned ON at the same time.
3. When the trigger condition (Trace Start Bit ON, specified bit ON, or value of specified word matching set value) is met, the sampled data will be valid beginning with the next sample plus or minus the number of samples set with the delay setting. The Trace Trigger Monitor Flag (A508.11) will be turned ON at the same time.
4. The trace will end when TRSM(045) has been executed enough times to fill the Trace Memory. When the trace ends, the Trace Completed Flag (A508.12) will be turned ON and the Trace Busy Flag (A508.13) will be turned OFF.
5. Read the contents of Trace Memory with the CX-Programmer.

The following table shows relevant bits and flags in the Auxiliary Area. Only A508.14 and A508.15 are meant to be controlled by the user, and A508.15

must not be turned ON from the program, i.e., it must be turned ON only from the CX-Programmer.

Name	Address	Operation
Trace Trigger Monitor Flag	A508.11	This flag is turned ON when the trigger condition has been established with the Trace Start Bit. It is turned OFF when sampling is started for the next trace (by the Sampling Start Bit).
Trace Completed Flag	A508.12	This flag is turned ON when trace samples have filled the Trace Memory. It is turned OFF the next time that the Sampling Start Bit goes from OFF to ON.
Trace Busy Flag	A508.13	This flag is turned ON when the Sampling Start Bit goes from OFF to ON. It is turned OFF when the trace is completed.
Trace Start Bit	A508.14	The trace trigger conditions are established when this bit is turned from OFF to ON. The offset indicated by the delay value (positive or negative) determines which data samples are valid.
Sampling Start Bit	A508.15	When this bit is turned from OFF to ON from the CX-Programmer, data samples will start being stored in Trace Memory with one of the following three methods: 1) Periodic sampling (10 to 2,550 ms intervals) 2) Sampling at TRSM(045) execution 3) Sampling at the end of each cycle This bit must be turned ON and OFF from the CX-Programmer.

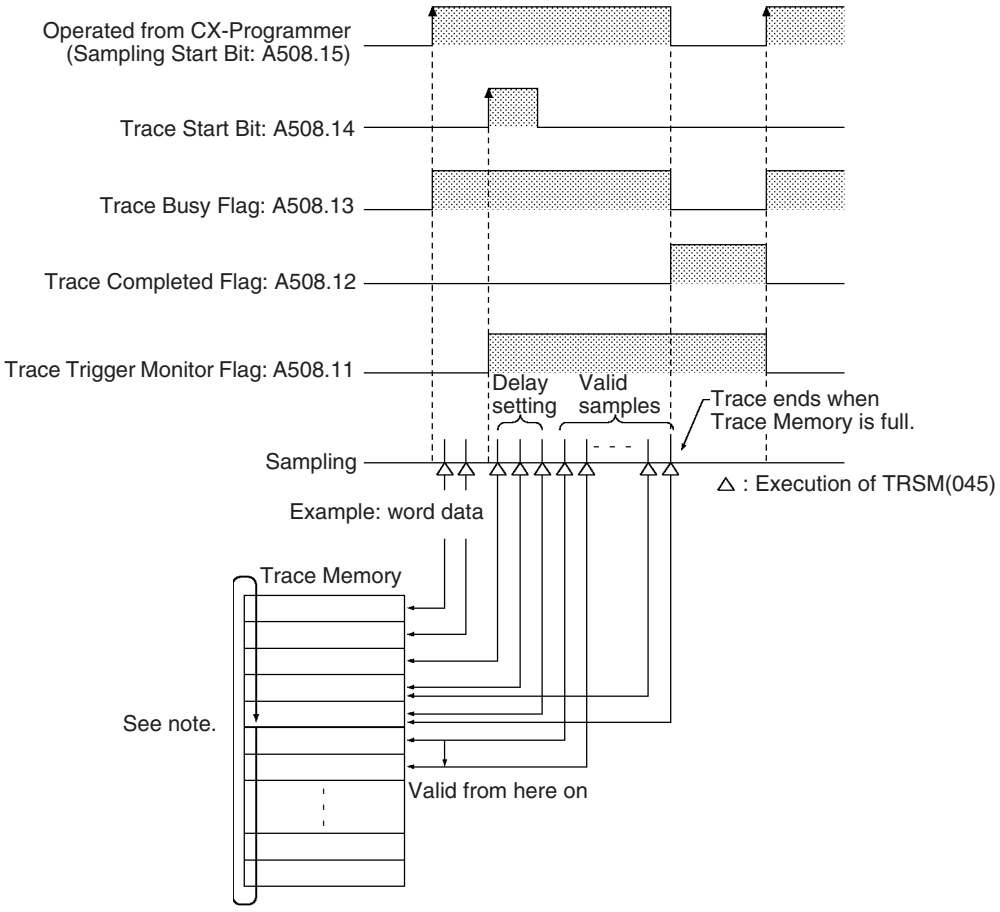
Precautions

TRSM(045) is processed as NOP(000) when data tracing is not being performed or when the sampling interval set in the parameters with the CX-Programmer is not set to sample on TRSM(045) instruction execution.

Do not turn the Sampling Start Bit (A508.15) ON or OFF from the program. This bit must be turned ON and OFF from the CX-Programmer.

Example

The following example shows the overall data trace operation.



Note Trace Memory has a ring structure. Data is stored to the end of the Trace Memory area and then wraps to the beginning of the area, ending just before the first valid data sample.

3-25 Failure Diagnosis Instructions

This section describes instructions used to define and handle errors.

Instruction	Mnemonic	Function code	Page
FAILURE ALARM	FAL	006	600
SEVERE FAILURE ALARM	FALS	007	603

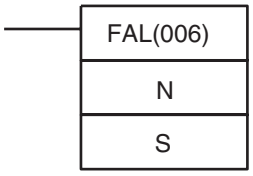
3-25-1 FAILURE ALARM: FAL(006)

Purpose

Generates or clears user-defined non-fatal errors. Non-fatal errors do not stop FQM1 operation.

Ladder Symbol

- Generating or Clearing User-defined Non-fatal Errors



N: FAL number

S: #0000 or #0000 to #FFFF

Variations

Variations		
	Executed Each Cycle for ON Condition	FAL(006)
	Executed Once for Upward Differentiation	@FAL(006)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

Generating or Clearing User-defined Non-fatal Errors

The following table shows the function of the operands.

N	S	Function
0	#0001 to #01FF	Clears the non-fatal error with the corresponding FAL number.
	#FFFF	Clears all non-fatal errors.
	Other (See note.)	Clears the most serious non-fatal error.
1 to 511 (These FAL numbers are shared with FALS numbers.)	Creating errors: #0000 Clearing errors: #0000 to #FFFF	Generates a non-fatal error with the corresponding FAL number.

Note Other settings would be constants #0200 through #FFFE or a word address.

Operand Specifications

Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W255
Auxiliary Bit Area	---	A000 to A959
Timer Area	---	T0000 to T0255
Counter Area	---	C0000 to C0255
DM Area	---	D00000 to D32767
Indirect DM addresses in binary	---	@ D00000 to @ D32767

Area	N	S
Indirect DM addresses in BCD	---	*D00000 to *D32767
Constants	0 to 511	#0000 to #FFFF (binary)
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

The operation of FAL(006) depends on the value of N. Set N to 0000 to clear an error and set N to 0001 to 01FF to generate an error.

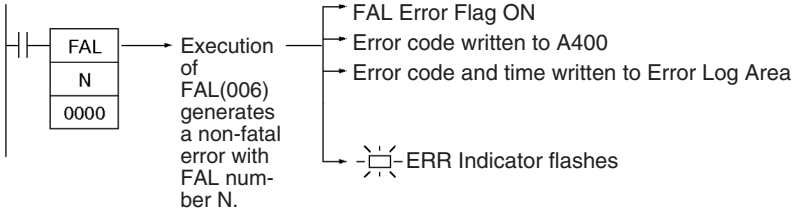
Generating Non-fatal User-defined Errors

When FAL(006) is executed with N set to an FAL number (&1 to &511) that is not equal to the content of A400 (the system-generated FAL/FALS number), a non-fatal error will be generated with that FAL number and the following processing will be performed:

1,2,3...

1. The FAL Error Flag (A402.15) will be turned ON. (FQM1 operation will continue.)
2. The error code will be written to A400. Error codes 4101 to 42FF correspond to FAL numbers 0001 to 01FF (1 to 511).

   Note If a fatal error or a more serious non-fatal error occurs at the same time as the FAL(006) instruction, the more serious error's error code will be written to A400.
3. The error code will be written to the Error Log Area (A100 through A199).
4. The ERR Indicator on the Modules will flash.



Clearing Non-fatal Errors without the CX-Programmer

When FAL(006) is executed with N set to 0, non-fatal errors can be cleared. The value of S will determine the processing, as shown in the following table.

S	Process
&1 to &511 (0001 to 01FF hex)	The FAL error of the specified number will be cleared.
FFFF hex	All non-fatal errors (including system errors) will be cleared.
0200 to FFFE hex or word specification	The most serious non-fatal error (even if it is a non-fatal system error) that has occurred will be cleared. When more than one FAL error has occurred, the FAL error with the smallest FAL number will be cleared.

Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range of 0 to 511 decimal. OFF in all other cases.

The following tables show relevant words and flags in the Auxiliary Area.

- Auxiliary Area Words/Flags for User-defined Errors

Name	Address	Operation
FAL Error Flag	A402.15	ON when an error is generated with FAL(006).

- Auxiliary Area Words/Flags for both User-defined and System Errors

Name	Address	Operation
Error Log Area	A100 to A199	The Error Log Area contains the error codes for the most recent 20 errors, including errors generated by FAL(006).
Error code	A400	When an error occurs, its error code is stored in A400. The error codes for FAL numbers 0001 to 01FF are 4101 to 42FF, respectively. If two or more errors occur simultaneously, the error code of the most serious error will be stored in A400.

Precautions

N must be between 0 and 511. An error will occur and the Error Flag will be turned ON if N is outside of the specified range.

Examples

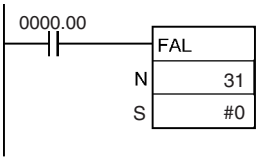
Creating a User-defined Error

When CIO 0000.00 turns ON in the following example, a non-fatal error will be generated with FAL number 31 and the following processing will be performed:

1,2,3...

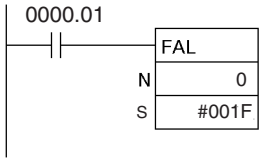
1. The FAL Error Flag (A402.15) will be turned ON. (FQM1 operation will continue.)
2. The error code 411F will be written to A400. Error codes 4101 to 42FF correspond to FAL numbers 0001 to 01FF (1 to 511).

   Note If a fatal error or a more serious non-fatal error occurs at the same time as or before the FAL(006) instruction, the more serious error's error code will be written to A400.
3. The error code will be written to the Error Log Area (A100 through A199).
4. The ERR Indicator on the Modules will flash.



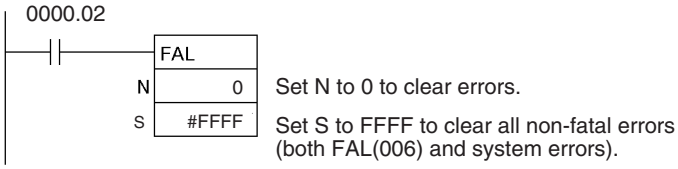
Clearing a Particular Non-fatal Error

When CIO 0000.01 is ON in the following example, FAL(006) will clear the non-fatal error with FAL number 31, and turn OFF the FAL Error Flag (A402.15).



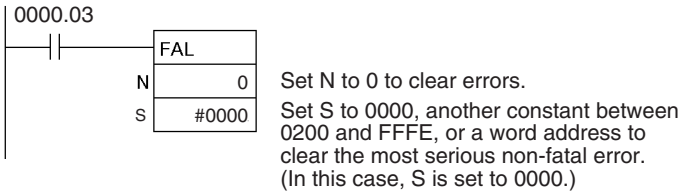
Clearing All Non-fatal Errors

When CIO 0000.02 is ON in the following example, FAL(006) will clear all of the non-fatal errors, and turn OFF the FAL Error Flag (A402.15).



Clearing the Most Serious Non-fatal Error

When CIO 0000.03 is ON in the following example, FAL(006) will clear the most serious non-fatal error that has occurred and reset the error code in A400. If the cleared error was originally generated by FAL(006), the FAL Error Flag (A402.15) will be turned OFF.

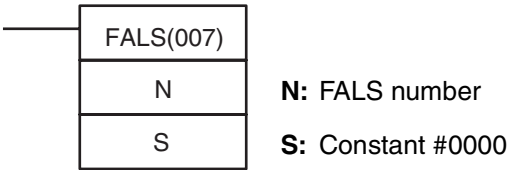


3-25-2 SEVERE FAILURE ALARM: FALS(007)

Purpose

Generates user-defined fatal errors. Fatal errors stop FQM1 operation.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	FALS(007)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

The following table shows the function of the operands.

Operand	Function
N	1 to 511 (These FALS numbers are shared with FAL numbers.)
S	Specify #0000.

Operand Specifications

Area	N	S
CIO Area	---	CIO 0000 to CIO 6143
Work Area	---	W000 to W255
Auxiliary Bit Area	---	A000 to A959
Timer Area	---	T0000 to T0255
Counter Area	---	C0000 to C0255
DM Area	---	D00000 to D32767
Indirect DM addresses in binary	---	@ D00000 to @ D32767
Indirect DM addresses in BCD	---	*D00000 to *D32767
Constants	1 to 511	#0000 to #FFFF (binary)
Index Registers	---	
Data Resisters	---	
Index Registers	---	
Indirect addressing using Index Registers	---	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

Description

FALS(007) generates user-defined error.

1,2,3...

1. The FALS Error Flag (A401.06) will be turned ON. (FQM1 operation will stop.)
2. The error code will be written to A400. Error codes C101 to C2FF correspond to FALS numbers 0001 to 01FF (1 to 511).

   Note If an error more serious than the FALS(007) instruction (one with a higher error code) has occurred, A400 will contain the more serious error's error code.
3. The error code will be written to the Error Log Area (A100 through A199).
4. The ERR Indicators on the FQM1 Modules will be lit.

Note Input #1 to #511 for the FALS number on the CX-Programmer.

Clearing FALS(007) User-defined Fatal Errors

To clear errors generated by FALS(007), first eliminate the cause of the error, and then either clear the error from the CX-Programmer or turn the FQM1 OFF and then ON again.

Flags

Name	Label	Operation
Error Flag	ER	ON if N is not within the specified range 1 to 511. OFF in all other cases.

The following tables show relevant words and flags in the Auxiliary Area.

- Auxiliary Area Words/Flags for User-defined Errors Only

Name	Address	Operation
FALS Error Flag	A401.06	ON when an error is generated with FALS(007).

- Auxiliary Area Words/Flags for both User-defined and System Errors

Name	Address	Operation
Error Log Area	A100 to A199	The Error Log Area contains the error codes for the most recent 20 errors, including errors generated by FALS(007).
Error code	A400	When an error occurs, its error code is stored in A400. The error codes for FALS numbers 0001 to 01FF (1 to 511 decimal) are C101 to C2FF, respectively. If two or more errors occur simultaneously, the error code of the most serious error will be stored in A400.

Precautions

N must be between 0001 and 01FF. An error will occur and the Error Flag will be turned ON if N is outside of the specified range.

Examples

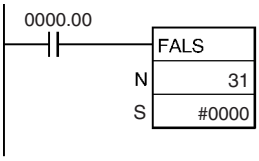
Generating a User-defined Error

When CIO 0000.00 is ON in the following example, FALS(007) will generate a fatal error with FALS number 31 and execute the following processes.

1,2,3...

1. The FALS Error Flag (A401.06) will be turned ON.
2. The corresponding error code (C11F) will be written to A400.

   Note A400 will contain the error code of the most serious of all of the errors that have occurred, including non-fatal and fatal system errors, as well as errors generated by FAL(006) and FAL(007).
3. The error code will be written to the Error Log Area (A100 through A199).
4. The ERR Indicators on the Modules will be lit.



3-26 Other Instructions

This section describes instructions for manipulating the Carry Flag, selecting the EM bank, and extending the maximum cycle time.

Instruction	Mnemonic	Function code	Page
SET CARRY	STC	040	606
CLEAR CARRY	CLC	041	606

3-26-1 SET CARRY: STC(040)

Sets the Carry Flag (CY).

Ladder Symbol

Variations

Variations		
	Executed Each Cycle for ON Condition	STC(040)
	Executed Once for Upward Differentiation	@STC(040)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Description

When the execution condition is ON, STC(040) turns ON the Carry Flag (CY). Although STC(040) turns the Carry Flag ON, the flag will be turned ON/OFF by the execution of subsequent instructions that affect the Carry Flag.

Flags

Name	Label	Operation
Error Flag	ER	Unchanged
Equals Flag	=	Unchanged
Carry Flag	CY	ON
Negative Flag	N	Unchanged

Precautions

ROL(027), ROLL(572), ROR(028), and RORL(573) make use of the Carry Flag in their rotation shift operations. When using any of these instructions, use STC(040) and CLC(041) to set and clear the Carry Flag.

3-26-2 CLEAR CARRY: CLC(041)

Purpose

Turns OFF the Carry Flag (CY).

Ladder Symbol

Variations

Variations		
	Executed Each Cycle for ON Condition	CLC(041)
	Executed Once for Upward Differentiation	@CLC(041)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Description

When the execution condition is ON, CLC(041) turns OFF the Carry Flag (CY). Although CLC(041) turns the Carry Flag OFF, the flag will be turned ON/OFF by the execution of subsequent instructions which affect the Carry Flag.

Flags

Name	Label	Operation
Error Flag	ER	Unchanged
Equals Flag	=	Unchanged
Carry Flag	CY	OFF
Negative Flag	N	Unchanged

Precautions

+C(402), +CL(403), +BC(406), and +BCL(407) make use of the Carry Flag in their addition operations. Use CLC(041) just before any of these instructions to prevent any influence from other preceding instructions.

–C(412), –CL(413), –BC(416), and –BCL(417) make use of the Carry Flag in their subtraction operations. Use CLC(041) just before any of these instructions to prevent any influence from other preceding instructions.

ROL(027), ROLL(572), ROR(028), and RORL(573) make use of the Carry Flag in their rotation shift operations. When using any of these instructions, use STC(040) and CLC(041) to set and clear the Carry Flag.

Note The +(400), +L(401), +B(404), +BL(405), –(410), –L(411), –B(414), and –BL(415) instructions do not include the Carry Flag in their addition and subtraction operations. In general, use these instructions when performing addition or subtraction.

3-27 Block Programming Instructions

This section describes block programs and the block programming instructions.

Instruction	Mnemonic	Function code	Page
BLOCK PROGRAM BEGIN	BPRG	096	611
BLOCK PROGRAM END	BEND	801	611
IF (NOT)	IF (NOT)	802	613
ELSE	ELSE	803	613
IF END	IEND	804	613

3-27-1 Introduction

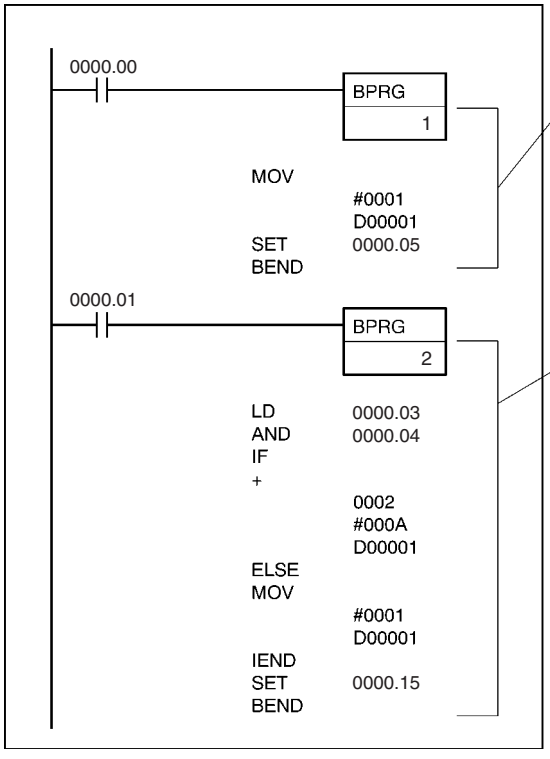
Block Programs

Up to 128 block programs can be created within the overall user program (all tasks). The execution of each block program is controlled by a single execution condition. All instructions between BPRG(096) and BEND(801) are executed unconditionally when the execution condition for BPRG(096) is turned ON. The execution of all the block programming instructions except for BPRG(096) is not affected by the execution condition. This allows programming that is to be executed under a single execution condition to be grouped together in one block program.

Each block is started by one execution condition in the ladder diagram and all instructions within the block are written in mnemonic form. The block program is thus a combination of ladder and mnemonic instructions.

Block programs enable programming operations that can be difficult to program with ladder diagrams, such as conditional branches and step progressions.

The following example shows two block programs.



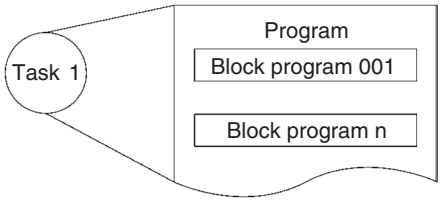
Tasks and Block Programs

Block programs can be located within tasks. While tasks are used to divide large programming units, block programs can be used within tasks to further divide programming into smaller units controlled with a single ladder diagram execution condition.

Just like tasks, block programs that are not executed (i.e., which have an OFF input condition) do not require execution time and can thus be used to reduce the cycle time (somewhat the same as jumps). Also like tasks, other blocks can be paused or restarted from within a block program.

There are, however, differences between tasks and block programs. One difference is that input conditions are not used within block programs unless intentionally programmed with IF(802) instructions. Also, there are some instructions that cannot be used within block programs, such as those that detect upward and downward differentiation.

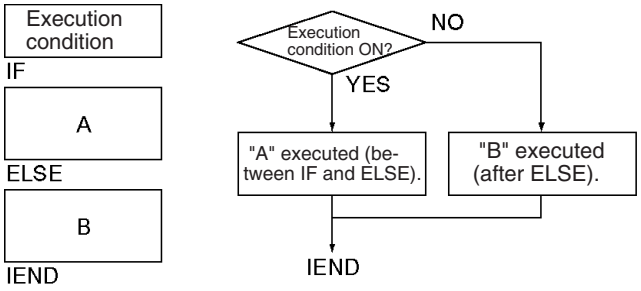
Block programs can be used either within cyclic tasks or interrupt tasks. Each block program number from 0 to 127 can be used only once and cannot be used again, even in a different task.



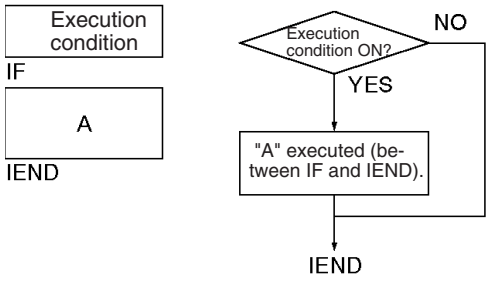
Using Block Programming Instructions

Basically speaking, IF(802), ELSE(803), and IEND(810) are used for execution conditions (along with bits) inside block programs.

If “A” or “B” is to be executed, then IF A ELSE B IEND are used as shown below.



If “A” or nothing is to be executed, IF A IEND are used as shown below.



Instructions Taking Execution Conditions within Block Programs

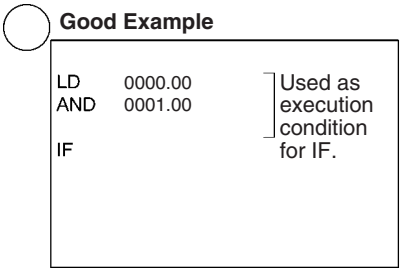
The following instruction can take execution conditions within a block program.

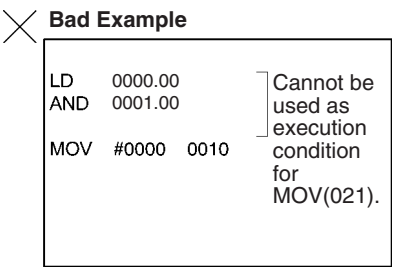
Instruction type	Instruction name	Mnemonic
Block programming instructions	IF (NOT)	IF (NOT)

Instructions with Application Restrictions within Block Programs

The instructions listed in the following table can be used only to create execution conditions for IF(802) and cannot be used by themselves. The execution of these instructions may be unpredictable if used by themselves or in combination with any other instructions.

Mnemonic	Name
LD/LD NOT	LOAD/LOAD NOT
AND/AND NOT	AND/AND NOT
OR/OR NOT	OR/OR NOT
>, <, =, >=, <=, <> (S) (L)	Symbol Comparison Instruction (not right-hand instructions)





Instructions Not Applicable in Block Programs

The instructions listed in the following table cannot be used within block programs.

Instruction group	Mnemonic	Name	Alternative
Sequence Output Instructions	OUT	OUTPUT	Use SET and RSET.
	OUT NOT	OUTPUT NOT	
	DIFU(013)	DIFFERENTIATE UP	None
	DIFD(014)	DIFFERENTIATE DOWN	None
	KEEP(011)	KEEP	None
Sequence Control Instructions	IL(002) and ILC(003)	INTERLOCK and INTERLOCK CLEAR	Divide the block program into smaller blocks.
	END(001)	END	Use BEND(801).
Timer and Counter Instructions	TIM	TIMER	None
	TIMH(015)	HIGH-SPEED TIMER	
	TMHH(540)	ONE-MS TIMER	
	CNT	COUNTER	
	CNTR(012)	REVERSIBLE COUNTER	
Subroutine Instructions	SBN(092) and RET(093)	SUBROUTINE ENTRY and SUBROUTINE RETURN	None
Shift Instructions	SFT(010)	SHIFT REGISTER	Use other Shift Instructions.

Instruction group	Mnemonic	Name	Alternative
Upward and Downward Differentiated Instructions	Mnemonics with @	Upward Differentiated Instructions	None
	Mnemonics with %	Downward Differentiated Instructions	None

Note JMP(004) and JME(005) can be used. JMP(004) does not take any execution condition and jumps to JME(005) unconditionally.

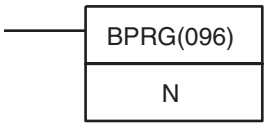
3-27-2 BLOCK PROGRAM BEGIN/END: BPRG(096)/BEND(801)

Purpose

Define a block programming area. For every BPRG(096), there must be a corresponding BEND(801).

Ladder Symbols

BLOCK PROGRAM BEGIN



N: Block program number

BLOCK PROGRAM END

BEND(801)

Variations

BPRG(096)

Variations		
Variations	**Executed Each Cycle for ON Condition**	BPRG(096)
	Executed Once for Upward Differentiation	Not supported.
	Executed Once for Downward Differentiation	Not supported.

BEND(801)

Variations	Always Executed in Block Program

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
(See note.)	OK	OK	OK

Note BPRG(096) is allowed only once at the beginning of each block program.

Operands

N: Block Program Number
The block program number must be between 0 and 127 decimal.

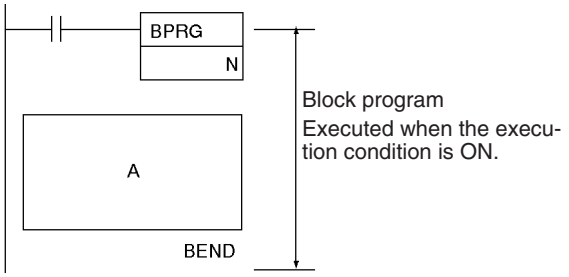
Operand Specifications (BPRG(096))

Area	N
CIO Area	---
Work Area	---
Auxiliary Bit Area	---
Timer Area	---
Counter Area	---
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	0 to 127 (decimal)

Area	N
Index Registers	---
Indirect addressing using Index Registers	---

Description

BPRG(096) executes the block program with the block number designated in N, i.e., the one immediately after it and ending with BEND(801). All instructions between BPRG(096) and BEND(801) are executed with ON execution conditions (i.e., unconditionally).



When the execution condition for BPRG(096) is OFF, the block program will not be executed and no execution time will be required for the instructions in the block program.

Flags

BPRG(096)

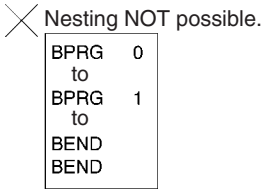
Name	Label	Operation
Error Flag	ER	ON if BPRG(096) is already being executed. ON if N is not between 0 and 127 (BCD). ON if the same block program number is used more than once. OFF in all other cases.

BEND(801)

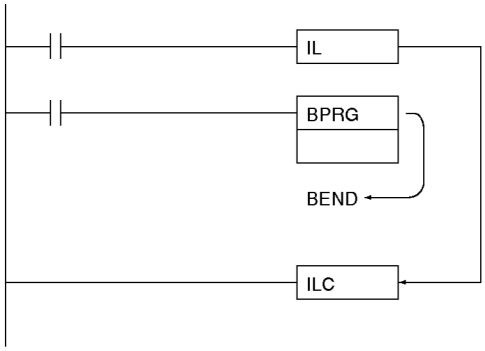
Name	Label	Operation
Error Flag	ER	ON if a block program is not being executed. OFF in all other cases.

Precautions

Each block program number can be used only once within the entire user program. Block programs cannot be nested.



If the block program is in an interlocked program section and the execution condition for IL(002) is OFF, the block program will not be executed.

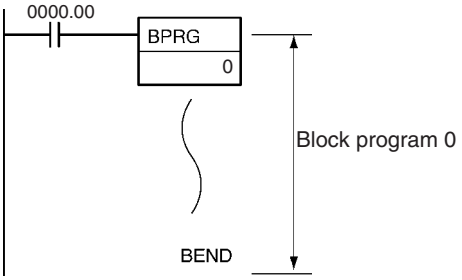


BPRG(096) and the corresponding BEND(801) must be in the same task.

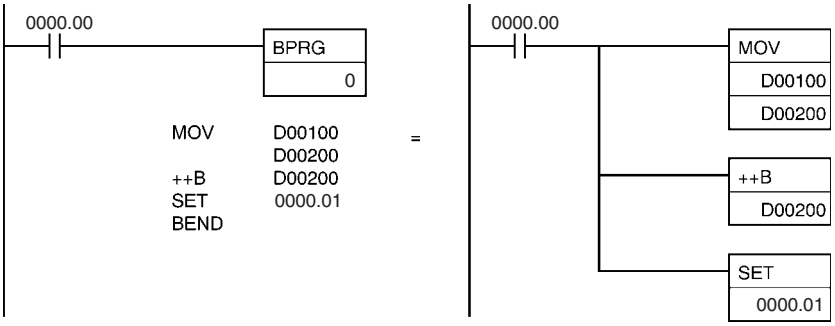
An error will occur and the Error Flag will turn ON if BPRG(096) is in the middle of a block program, BEND(801) is not in a block program, N is not between #0000 and #007F (binary), there is no block program, or if the same block program number is used more than once.

Examples

When CIO 0000.00 turns ON in the following example, block program 0 will be executed. When CIO 0000.00 is OFF, the block program will not be executed.



The two program sections shown below both execute MOV(021), ++B(594), and SET for the same execution condition (i.e., when CIO 0000.00 turns ON).



3-27-3 Branching: IF (NOT)(802), ELSE(803), and IEND(804)

Purpose

Branches the block program either based on an execution condition or on the status of an operand bit.

Ladder Symbol

IF(802) B

IF(802)

IF(802) NOT B

ELSE(803)

IEND(804)

B: Bit operand

Variations

Variations	Always Executed in Block Program

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Note IF(802), ELSE(803), and IEND(804) can be used in block programming regions within subroutines and interrupt tasks.

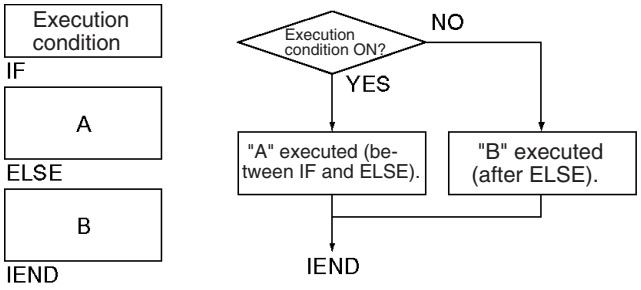
Operand Specifications

Area	B
CIO Area	CIO 0000.00 to CIO 6143.15
Work Area	W000.00 to W255.15
Auxiliary Bit Area	A00000 to A959.15
Timer Area	T0000 to T0255
Counter Area	C0000 to C0255
Task Flags	TK0000
Condition Flags	ER, CY, >, =, <, N, OF, UF, >=, <>, <=, ON, OFF, AER
Clock Pulses	0.02 s, 0.1 s, 0.2 s, 1 s, 1 min
DM Area	---
Indirect DM addresses in binary	---
Indirect DM addresses in BCD	---
Constants	---
Data Resisters	---
Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15

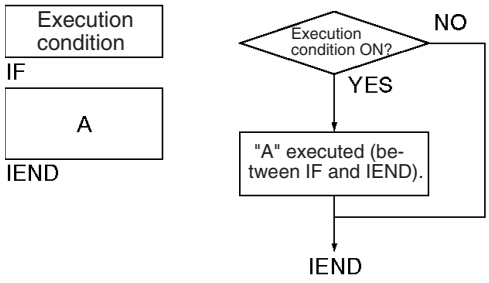
Description

Operation without an Operand for IF(802)

If an operand bit is not specified, an execution condition must be created before IF(802) starting with LD. If the execution condition is ON, the instructions between IF(802) and ELSE(803) will be executed and if the execution condition is OFF, the instructions between ELSE(803) and IEND(804) will be executed.

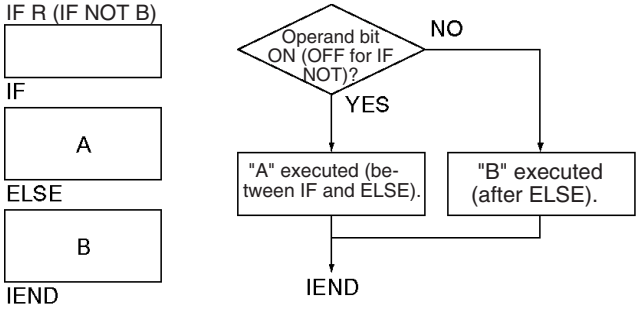


If the ELSE(803) instruction is omitted and the execution condition is ON, the instructions between IF(802) and IEND(804) will be executed and if the execution condition is OFF, only the instructions after IEND(804) will be executed.

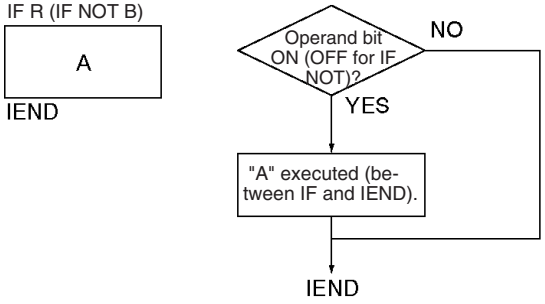


Operation with an Operand for IF(802) or IF NOT(802)

An operand bit, B, can be specified for IF(802) or IF NOT(802). If the operand bit is ON, the instructions between IF(802) and ELSE(803) will be executed. If the operand bit is OFF, the instructions between ELSE(803) and IEND(804) will be executed. For IF NOT(802), the instructions between IF(802) and ELSE(803) will be executed if the operand bit is OFF, and the instructions between ELSE(803) and IEND(804) will be executed if the operand bit is ON.



If the ELSE(803) instruction is omitted and the operand bit is ON, the instructions between IF(802) and IEND(804) will be executed and if the operand bit is OFF, only the instructions after IEND(804) will be executed. The same will happen for the opposite status of the operand bit if IF NOT(802) is used.



Flags

Name	Label	Operation
Error Flag	ER	ON if the branch instructions are not in a block program. ON if more than 253 branches are nested. OFF in all other cases.

Precautions

Instructions in block programs are generally executed unconditionally. Branching, however, can be used to create conditional execution based on execution conditions or operand bits.

Use IF A ELSE B IEND to branch between A and B.

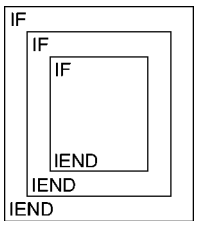
Use IF A IEND to branch between A and doing nothing.

Branches can be nested to up to 253 levels.

A error will occur and the Error Flag will turn ON if the branch instructions are not in a block program or if more than 253 branches are nested.

Nesting Branches

Up to 253 branches can be nested within the top level branch.

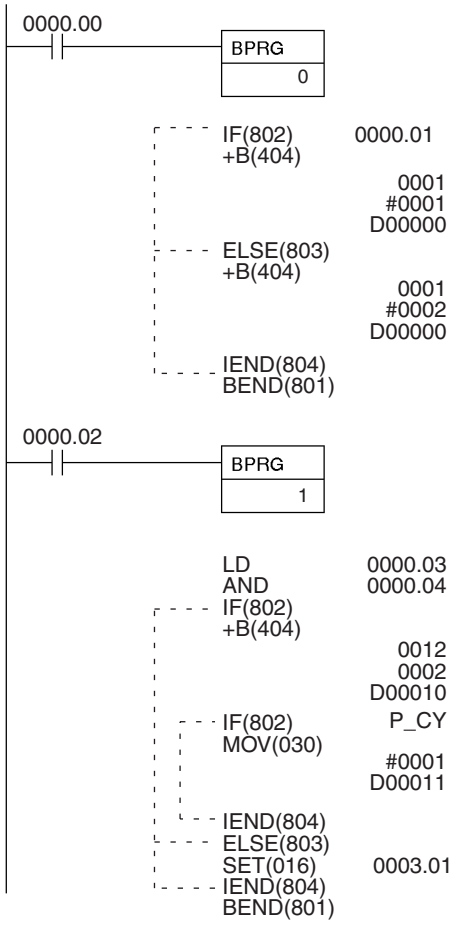


Examples

The following example shows two different block programs controlled by CIO 0000.00 and CIO 0000.02.

The first block executes one of two additions depending on the status of CIO 0000.01. This block is executed when CIO 0000.00 is ON. If CIO 0000.01 is ON, 0001 is added to the contents of CIO 0001. If CIO 0000.01 is OFF, 0002 is added to the contents of CIO 0001. In either case, the result is placed in D00000.

The second block is executed when CIO 0000.02 is ON and shows nesting two levels. If CIO 0000.03 and CIO 0000.04 are both ON, the contents of CIO 0012 and CIO 0002 are added and the result is placed in D00010 and then 0001 is moved into D00011 based on the status of CY. If either CIO 0000.03 or CIO 0000.04 is OFF, then the entire addition operation is skipped and CIO 0003.01 is turned ON.



Address	Instruction	Operands
000000	LD	0000.00
000001	BPRG(096)	0
000002	IF(802)	0000.01
000003	+B(404)	
		0001
		#0001
		D00000
000004	ELSE(803)	
000005	+B(404)	
		0001
		#0002
		D00000
000006	IEND(804)	
000007	BEND(801)	
000008	LD	0000.02
000009	BPRG(096)	1
000010	LD	0000.03
000011	AND	0000.04
000012	IF(802)	
000013	+B(404)	
		0012
		0002
		D00010
000014	IF(802)	CY
000015	MOV(030)	
		#0001
		D00011
000016	IEND(804)	
000017	ELSE(803)	
000018	SET(016)	0003.01
000019	IEND(804)	
000020	BEND(801)	

3-28 Function Block Instructions

This section describes the instructions used with function blocks.

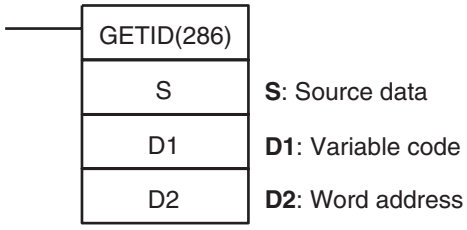
Instruction	Mnemonic	Function code	Page
GET VARIABLE ID	GETID	286	618

3-28-1 GET VARIABLE ID: GETID(286)

Purpose

Outputs the FINS command variable type (data area) code and word address for the specified variable or address. This instruction is generally used to get the assigned address of a variable in a function block.

Ladder Symbol



Variations

Variations		
	Executed Each Cycle for ON Condition	GETID(286)
	Executed Once for Upward Differentiation	@GETID(286)
	Executed Once for Downward Differentiation	Not supported.

Applicable Program Areas

Block program areas	Step program areas	Subroutines	Interrupt tasks
OK	OK	OK	OK

Operands

S: Source data
Specifies the variable or address for which the variable type and word address will be retrieved.

D1: Variable code
Contains the FINS variable type code (data area code) of the source data.

D2: Word address
Contains the word address of the source data in 4-digit hexadecimal.

Operand Specifications

Area	S	D1	D2
CIO Area	CIO 0000 to CIO 6143		
Work Area	W000 to W255		
Auxiliary Bit Area	---	A000 to A959	
Timer Area	---	T0000 to T0255	
Counter Area	---	C0000 to C0255	
DM Area	D00000 to D32767		
Indirect DM addresses in binary	@ D00000 to @ D32767		
Indirect DM addresses in BCD	*D00000 to *D32767		
Constants	---		
Data Registers	---	DR0 to DR15	

Area	S	D1	D2
Index Registers	---		
Indirect addressing using Index Registers	,IR0 to ,IR15 –2048 to +2047 ,IR0 to –2048 to +2047 ,IR15 DR0 to DR15, IR0 to DR0 to DR15, IR15 ,IR0+(++) to ,IR15+(++) ,–(– –)IR0 to , –(– –)IR15		

Description

GETID(286) retrieves the data area address of the specified source variable or address, outputs the data area code to D1 in 4-digit hexadecimal, and outputs the word address number to D2 in 4-digit hexadecimal.

The following table shows the variable type (data area) codes and corresponding address ranges for the PLC's data areas.

Data area		Data size	Data area code (Output to D1.)	Address (Output to D2.)
CIO Area	CIO	Word	00B0 hex	0000 to 17FF hex (0000 to 6,143 decimal)
Work Area	W		00B1 hex	0000 to 00FF hex (000 to 255 decimal)
DM Area	D		0082 hex	0000 to 7FFF hex (0000 to 32,767 decimal)

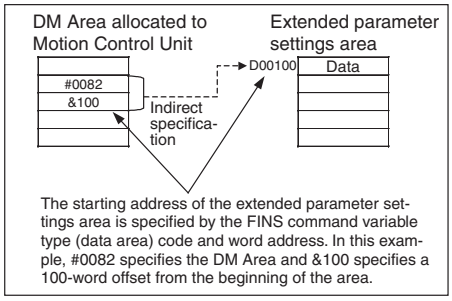
Variables in function blocks are automatically allocated addresses by the CX-Programmer unless the AT specification is used. For example, if it is necessary to indirectly specify the extended parameter settings of a Special Unit such as a Motion Control Unit and a variable is used at the beginning of the extended parameter settings area, that variable's address must be set. In this case, GETID(286) can be used to retrieve the variable's data area address.

Flags

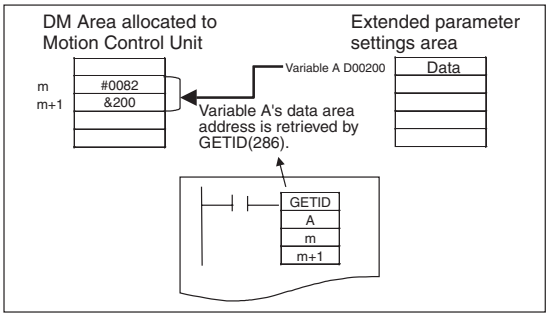
Name	Label	Operation
Error Flag	ER	ON if S is not within the allowed range.

Example

Normal Operation



Using Function Blocks



SECTION 4
Instruction Execution Times and Number of Steps

This section provides instruction execution times and the number of steps for each FQM1 instruction.

4-1 FQM1 Instruction Execution Times and Number of Steps 622
- 4-1-1 Sequence Input Instructions 622
- 4-1-2 Sequence Output Instructions 623
- 4-1-3 Sequence Control Instructions 623
- 4-1-4 Timer and Counter Instructions 624
- 4-1-5 Comparison Instructions 624
- 4-1-6 Data Movement Instructions 625
- 4-1-7 Data Shift Instructions 626
- 4-1-8 Increment/Decrement Instructions 627
- 4-1-9 Symbol Math Instructions 628
- 4-1-10 Conversion Instructions 629
- 4-1-11 Logic Instructions 629
- 4-1-12 Special Math Instructions 630
- 4-1-13 Floating-point Math Instructions 630
- 4-1-14 Double-precision Floating-point Instructions 631
- 4-1-15 Table Data Processing Instructions 632
- 4-1-16 Data Control Instructions 633
- 4-1-17 Subroutine Instructions 633
- 4-1-18 Interrupt Control Instructions 633
- 4-1-19 High-speed Counter and Pulse Output Instructions (Only for Motion Control Modules) 634
- 4-1-20 Step Instructions 635
- 4-1-21 I/O Refresh Instructions 635
- 4-1-22 Serial Communications Instructions 636
- 4-1-23 Debugging Instructions 636
- 4-1-24 Failure Diagnosis Instructions 636
- 4-1-25 Other Instructions 636
- 4-1-26 Block Programming Instructions 637
- 4-1-27 Special Function Block Instructions 637

4-1 FQM1 Instruction Execution Times and Number of Steps

The following table lists the execution times for all instructions that are available for the FQM1.

The total execution time of instructions within one whole user program is the process time for program execution when calculating the cycle time. (See note.)

Note User programs are allocated tasks that can be executed within cyclic tasks and interrupt tasks that satisfy interrupt conditions.

Execution times for most instructions differ depending on the conditions when the instruction is executed. The top line for each instruction in the following table shows the minimum time required to process the instruction and the necessary execution conditions, and the bottom line shows the maximum time and execution conditions required to process the instruction.

The execution time can also vary when the execution condition is OFF.

The following table also lists the length of each instruction in the *Length (steps)* column. The number of steps required in the user program area for each of the instructions varies from 1 to 7 steps, depending upon the instruction and the operands used with it. The number of steps in a program is not the same as the number of instructions.

Note

1. Program capacity for the FQM1 is measured in steps. Basically speaking, 1 step is equivalent to 1 word.

   Most instructions are supported in differentiated form (indicated with ↑, ↓, @, and %). Specifying differentiation will increase the execution times by the following amounts.

Symbol	μs
↑ or ↓	+0.5
@ or %	+0.5

2. Use the following time as a guideline when instructions are not executed. Approx. 0.2 to 0.5 μs

4-1-1 Sequence Input Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
LOAD	LD	---	1	0.10	Yes	---
LOAD NOT	LD NOT	---	1	0.10	Yes	---
AND	AND	---	1	0.10	Yes	---
AND NOT	AND NOT	---	1	0.10	Yes	---
OR	OR	---	1	0.10	Yes	---
OR NOT	OR NOT	---	1	0.10	Yes	---
AND LOAD	AND LD	---	1	0.05	Yes	---
OR LOAD	OR LD	---	1	0.05	Yes	---
NOT	NOT	520	1	0.05	Yes	---
CONDITION ON	UP	521	3	0.50	Yes	---
CONDITION OFF	DOWN	522	4	0.50	Yes	---
LOAD BIT TEST	LD TST	350	4	0.35	Yes	---
LOAD BIT TEST	LD TSTN	351	4	0.35	Yes	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
AND BIT TEST	AND TST	350	4	0.35	Yes	---
AND BIT TEST	AND TSTN	351	4	0.35	Yes	---
OR BIT TEST	OR TST	350	4	0.35	Yes	---
OR BIT TEST	OR TSTN	351	4	0.35	Yes	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-2 Sequence Output Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
OUTPUT	OUT	---	1	0.35	Yes	---
OUTPUT NOT	OUT NOT	---	1	0.35	Yes	---
KEEP	KEEP	011	1	0.40	Yes	---
DIFFERENTIATE UP	DIFU	013	2	0.50	Yes	---
DIFFERENTIATE DOWN	DIFD	014	2	0.50	Yes	---
SET	SET	---	1	0.30	Yes	---
RESET	RSET	---	1	0.30	Yes	---
MULTIPLE BIT SET	SETA	530	4	11.8	---	Setting 1 bit
				64.1	---	Setting 1,000 bits
MULTIPLE BIT RESET	RSTA	531	4	11.8	---	Resetting 1 bit
				64.0	---	Resetting 1,000 bits
SINGLE BIT RESET	RSTB	533	2	0.50	Yes	---
SINGLE BIT OUTPUT	OUTB	534	2	0.50	Yes	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-3 Sequence Control Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
END	END	001	1	7.0	Yes	---
NO OPERATION	NOP	000	1	0.05	Yes	---
INTERLOCK	IL	002	1	0.15	Yes	---
INTERLOCK CLEAR	ILC	003	1	0.15	Yes	---
JUMP	JMP	004	2	0.95	Yes	---
JUMP END	JME	005	2	---	---	---
CONDITIONAL JUMP	CJP	510	2	0.95	Yes	When jump condition is met

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
CONDITIONAL JUMP	CJPN	511	2	0.95	Yes	When jump condition is met
MULTIPLE JUMP	JMP0	515	1	0.15	Yes	---
MULTIPLE JUMP END	JME0	516	1	0.15	Yes	---
FOR-NEXT LOOPS	FOR	512	2	1.00	Yes	Specification with a constant
FOR-NEXT LOOPS	NEXT	513	1	0.45	Yes	When repeating a loop
				0.55	Yes	When ending loop repetition
BREAK LOOP	BREAK	514	1	0.15	Yes	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-4 Timer and Counter Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
TIMER	TIM	---	3	1.30	Yes	---
COUNTER	CNT	---	3	1.30	Yes	---
HIGH-SPEED TIMER	TIMH	015	3	1.80	Yes	---
ONE-MS TIMER	TMHH	540	3	1.75	Yes	---
REVERSIBLE COUNTER	CNTR	012	3	24.8	---	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-5 Comparison Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
Input Comparison Instructions (unsigned)	LD, AND, OR +=	300	4	0.35	Yes	---
	LD, AND, OR + <>	305				
	LD, AND, OR + <	310				
	LD, AND, OR +<=	315				
	LD, AND, OR +>	320				
	LD, AND, OR +>=	325				
Input Comparison Instructions (double, unsigned)	LD, AND, OR +=+L	301	4	0.35	Yes	---
	LD, AND, OR +<>+L	306				
	LD, AND, OR +<+L	311				
	LD, AND, OR +<=+L	316				
	LD, AND, OR +>+L	321				
	LD, AND, OR +>=+L	326				

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
Input Comparison Instructions (signed)	LD, AND, OR +=+S	302	4	0.35	Yes	---
	LD, AND, OR +<>+S	307				
	LD, AND, OR +<+S	312				
	LD, AND, OR +<=+S	317				
	LD, AND, OR +>+S	322				
	LD, AND, OR +>=+S	327				
Input Comparison Instructions (double, signed)	LD, AND, OR +=+SL	303	4	0.35	Yes	---
	LD, AND, OR +<>+SL	308				
	LD, AND, OR +<+SL	313				
	LD, AND, OR +<=+SL	318				
	LD, AND, OR +>+SL	323				
	LD, AND, OR +>=+SL	328				
COMPARE	CMP	020	3	0.10	Yes	---
DOUBLE COMPARE	CMPL	060	3	0.50	Yes	---
SIGNED BINARY COMPARE	CPS	114	3	0.30	Yes	---
DOUBLE SIGNED BINARY COMPARE	CPSL	115	3	0.50	Yes	---
TABLE COMPARE	TCMP	085	4	30.3	---	---
MULTIPLE COMPARE	MCMP	019	4	47.5	---	---
UNSIGNED BLOCK COMPARE	BCMP	068	4	50.3	---	---
EXPANDED BLOCK COMPARE	BCMP2	502	4	15.3	---	Number of data words: 1
				689.1	---	Number of data words: 255
AREA RANGE COMPARE	ZCP	088	3	11.6	---	---
DOUBLE AREA RANGE COMPARE	ZCPL	116	3	11.4	---	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-6 Data Movement Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
MOVE	MOV	021	3	0.30	Yes	---
DOUBLE MOVE	MOVL	498	3	0.60	Yes	---
MOVE NOT	MVN	022	3	0.35	Yes	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
DOUBLE MOVE NOT	MVNL	499	3	0.60	Yes	---
MOVE BIT	MOVB	082	4	0.50	Yes	---
MOVE DIGIT	MOVD	083	4	0.50	Yes	---
BLOCK TRANSFER	XFER	070	4	0.8	Yes	Transferring 1 word
				650.2	Yes	Transferring 1,000 words
BLOCK SET	BSET	071	4	0.55	Yes	Setting 1 word
				400.2	Yes	Setting 1,000 words
DATA EXCHANGE	XCHG	073	3	0.80	Yes	---
SINGLE WORD DISTRIBUTE	DIST	080	4	10.5	---	---
DATA COLLECT	COLL	081	4	10.5	---	---
MOVE TO REGISTER	MOVR	560	3	0.60	Yes	---
MOVE TIMER/ COUNTER PV TO REGISTER	MOVRW	561	3	0.60	---	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-7 Data Shift Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
SHIFT REGISTER	SFT	010	3	12.4	---	Shifting 1 word
				368.1	---	Shifting 1,000 words
REVERSIBLE SHIFT REGISTER	SFTR	084	4	14.0	---	Shifting 1 word
				1.44 ms	---	Shifting 1,000 words
ASYNCHRONOUS SHIFT REGISTER	ASFT	017	4	13.9	---	Shifting 1 word
				3.915 ms	---	Shifting 1,000 words
WORD SHIFT	WSFT	016	4	9.7	---	Shifting 1 word
				728.1	---	Shifting 1,000 words
ARITHMETIC SHIFT LEFT	ASL	025	2	0.45	Yes	---
DOUBLE SHIFT LEFT	ASLL	570	2	0.80	Yes	---
ARITHMETIC SHIFT RIGHT	ASR	026	2	0.45	Yes	---
DOUBLE SHIFT RIGHT	ASRL	571	2	0.80	Yes	---
ROTATE LEFT	ROL	027	2	0.45	Yes	---
DOUBLE ROTATE LEFT	ROLL	572	2	0.80	Yes	---
ROTATE LEFT WITHOUT CARRY	RLNC	574	2	0.45	Yes	---
DOUBLE ROTATE LEFT WITHOUT CARRY	RLNL	576	2	0.80	Yes	---
ROTATE RIGHT	ROR	028	2	0.45	Yes	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
DOUBLE ROTATE RIGHT	RORL	573	2	0.80	Yes	---
ROTATE RIGHT WITHOUT CARRY	RRNC	575	2	0.45	Yes	---
DOUBLE ROTATE RIGHT WITHOUT CARRY	RRNL	577	2	0.80	Yes	---
ONE DIGIT SHIFT LEFT	SLD	074	3	10.1	---	Shifting 1 word
				1.208 ms	---	Shifting 1,000 words
ONE DIGIT SHIFT RIGHT	SRD	075	3	11.7	---	Shifting 1 word
				1.775 ms	---	Shifting 1,000 words
SHIFT N-BITS LEFT	NASL	580	3	0.45	Yes	---
DOUBLE SHIFT N-BITS LEFT	NSLL	582	3	0.80	Yes	---
SHIFT N-BITS RIGHT	NASR	581	3	0.45	Yes	---
DOUBLE SHIFT N-BITS RIGHT	NSRL	583	3	0.80	Yes	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-8 Increment/Decrement Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
INCREMENT BINARY	++	590	2	0.45	Yes	---
DOUBLE INCREMENT BINARY	++L	591	2	0.80	Yes	---
DECREMENT BINARY	– –	592	2	0.45	Yes	---
DOUBLE DECREMENT BINARY	– –L	593	2	0.80	Yes	---
INCREMENT BCD	++B	594	2	12.1	---	---
DOUBLE INCREMENT BCD	++BL	595	2	9.37	---	---
DECREMENT BCD	– –B	596	2	11.5	---	---
DOUBLE DECREMENT BCD	– –BL	597	2	9.3	---	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-9 Symbol Math Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
SIGNED BINARY ADD WITHOUT CARRY	+	400	4	0.30	Yes	---
DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+L	401	4	0.60	Yes	---
SIGNED BINARY ADD WITH CARRY	+C	402	4	0.40	Yes	---
DOUBLE SIGNED BINARY ADD WITH CARRY	+CL	403	4	0.60	Yes	---
BCD ADD WITHOUT CARRY	+B	404	4	16.3	---	---
DOUBLE BCD ADD WITHOUT CARRY	+BL	405	4	22.9	---	---
BCD ADD WITH CARRY	+BC	406	4	17.2	---	---
DOUBLE BCD ADD WITH CARRY	+BCL	407	4	24.1	---	---
SIGNED BINARY SUBTRACT WITHOUT CARRY	–	410	4	0.3	Yes	---
DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	–L	411	4	0.60	Yes	---
SIGNED BINARY SUBTRACT WITH CARRY	–C	412	4	0.40	Yes	---
DOUBLE SIGNED BINARY SUBTRACT WITH CARRY	–CL	413	4	0.60	Yes	---
BCD SUBTRACT WITHOUT CARRY	–B	414	4	16.3	---	---
DOUBLE BCD SUBTRACT WITHOUT CARRY	–BL	415	4	23.1	---	---
BCD SUBTRACT WITH CARRY	–BC	416	4	18.1	---	---
DOUBLE BCD SUBTRACT WITH CARRY	–BCL	417	4	24.2	---	---
SIGNED BINARY MULTIPLY	*	420	4	0.65	Yes	---
DOUBLE SIGNED BINARY MULTIPLY	*L	421	4	12.8	---	---
UNSIGNED BINARY MULTIPLY	*U	422	4	0.75	Yes	---
DOUBLE UNSIGNED BINARY MULTIPLY	*UL	423	4	12.4	---	---
BCD MULTIPLY	*B	424	4	16.9	---	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
DOUBLE BCD MULTIPLY	*BL	425	4	34.7	---	---
SIGNED BINARY DIVIDE	/	430	4	0.70	Yes	---
DOUBLE SIGNED BINARY DIVIDE	/L	431	4	11.9	---	---
UNSIGNED BINARY DIVIDE	/U	432	4	0.8	Yes	---
DOUBLE UNSIGNED BINARY DIVIDE	/UL	433	4	11.9	---	---
BCD DIVIDE	/B	434	4	18.3	---	---
DOUBLE BCD DIVIDE	/BL	435	4	26.7	---	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-10 Conversion Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
BCD-TO-BINARY	BIN	023	3	0.40	Yes	---
DOUBLE BCD-TO-DOUBLE BINARY	BINL	058	3	7.4	---	---
BINARY-TO-BCD	BCD	024	3	8.0	---	---
DOUBLE BINARY-TO-DOUBLE BCD	BCDL	059	3	8.0	---	---
2'S COMPLEMENT	NEG	160	3	0.35	Yes	---
DOUBLE 2'S COMPLEMENT	NEGL	161	3	0.60	Yes	---
ASCII CONVERT	ASC	086	4	11.8	---	Converting 1 digit into ASCII
				18.1	---	Converting 4 digits into ASCII
ASCII TO HEX	HEX	162	4	12.2	---	Converting 1 digit

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-11 Logic Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
LOGICAL AND	ANDW	034	4	0.30	Yes	---
DOUBLE LOGICAL AND	ANDL	610	4	0.60	Yes	---
LOGICAL OR	ORW	035	4	0.45	Yes	---
DOUBLE LOGICAL OR	ORWL	611	4	0.60	Yes	---
EXCLUSIVE OR	XORW	036	4	0.45	Yes	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
DOUBLE EXCLUSIVE OR	XORL	612	4	0.60	Yes	---
EXCLUSIVE NOR	XNRW	037	4	0.45	Yes	---
DOUBLE EXCLUSIVE NOR	XNRL	613	4	0.60	Yes	---
COMPLEMENT	COM	029	2	0.45	Yes	---
DOUBLE COMPLEMENT	COML	614	2	0.80	Yes	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-12 Special Math Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
ARITHMETIC PROCESS	APR	069	4	24.3	---	Linear approximation specification, normal
				12.1	---	Linear approximation table transfer, 1 word
				126.1	---	Linear approximation table transfer, 128 words
				241.3	---	Linear approximation table transfer, 256 words
				21.5	---	Linear approximation buffer specification, 256 words, beginning
				186.9	---	Linear approximation buffer specification, 256 words, end
				104.5	---	Linear approximation buffer specification, 128 words, end
BIT COUNTER	BCNT	067	4	0.65	Yes	Counting 1 word
VIRTUAL AXIS	AXIS	981	4	47.9	---	Relative mode
				48.1	---	Absolute mode
				8.3	---	Stopping processing

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-13 Floating-point Math Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
FLOATING TO 32-BIT	FIXL	451	3	7.4	---	---
32-BIT TO FLOATING	FLTL	453	3	7.0	---	---
FLOATING-POINT ADD	+F	454	4	11.4	---	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
FLOATING-POINT SUBTRACT	–F	455	4	11.0	---	---
FLOATING-POINT DIVIDE	/F	457	4	11.1	---	---
FLOATING-POINT MULTIPLY	*F	456	4	11.0	---	---
DEGREES TO RADIANS	RAD	458	3	9.7	---	---
RADIANS TO DEGREES	DEG	459	3	9.4	---	---
SINE	SIN	460	3	15.8	---	---
COSINE	COS	461	3	15.5	---	---
TANGENT	TAN	462	3	17.5	---	---
ARC SINE	ASIN	463	3	42.7	---	---
ARC COSINE	ACOS	464	3	42.5	---	---
ARC TANGENT	ATAN	465	3	21.3	---	---
SQUARE ROOT	SQRT	466	3	25.5	---	---
EXPONENT	EXP	467	3	18.1	---	---
LOGARITHM	LOG	468	3	16.1	---	---
EXPONENTIAL POWER	PWR	840	4	31.5	---	---
Floating Symbol Comparison	LD, AND, OR +=F	329	3	8.9	---	---
	LD, AND, OR +<>F	330				
	LD, AND, OR +<F	331				
	LD, AND, OR +<=F	332				
	LD, AND, OR +>F	333				
	LD, AND, OR +>=F	334				

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-14 Double-precision Floating-point Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
DOUBLE FLOATING TO 16-BIT BINARY	FIXD	841	3	15.0	---	---
DOUBLE FLOATING TO 32-BIT BINARY	FIXLD	842	3	15.2	---	---
16-BIT BINARY TO DOUBLE FLOATING	DBL	843	3	10.2	---	---
32-BIT BINARY TO DOUBLE FLOATING	DBLL	844	3	10.2	---	---
DOUBLE FLOATING-POINT ADD	+D	845	4	19.1	---	---
DOUBLE FLOATING-POINT SUBTRACT	–D	846	4	19.3	---	---

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
DOUBLE FLOATING-POINT MULTIPLY	*D	847	4	24.1	---	---
DOUBLE FLOATING-POINT DIVIDE	/D	848	4	34.7	---	---
DOUBLE DEGREES TO RADIANS	RADD	849	3	38.1	---	---
DOUBLE RADIANS TO DEGREES	DEGD	850	3	38.6	---	---
DOUBLE SINE	SIND	851	3	56.8	---	---
DOUBLE COSINE	COSD	852	3	53.5	---	---
DOUBLE TANGENT	TAND	853	3	125.4	---	---
DOUBLE ARC SINE	ASIND	854	3	27.0	---	---
DOUBLE ARC COSINE	ACOSD	855	3	29.6	---	---
DOUBLE ARC TANGENT	ATAND	856	3	19.5	---	---
DOUBLE SQUARE ROOT	SQRTD	857	3	62.3	---	---
DOUBLE EXPONENT	EXPD	858	3	158.1	---	---
DOUBLE LOGARITHM	LOGD	859	3	22.4	---	---
DOUBLE EXPONENTIAL POWER	PWRD	860	4	285.0	---	---
DOUBLE SYMBOL COMPARISON	LD, AND, OR +=D	335	3	13.1	---	---
	LD, AND, OR +<>D	336				
	LD, AND, OR +<D	337				
	LD, AND, OR +<=D	338				
	LD, AND, OR +>D	339				
	LD, AND, OR +>=D	340				

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-15 Table Data Processing Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
FIND MAXIMUM	MAX	182	4	13.0	---	Searching for 1 word
				1.41 ms	---	Searching for 1,000 words
FIND MINIMUM	MIN	183	4	12.8	---	Searching for 1 word
				1.412 ms	---	Searching for 1,000 words

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-16 Data Control Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
SCALING	SCL	194	4	22.7	---	---
SCALING 2	SCL2	486	4	21.8	---	---
SCALING 3	SCL3	487	4	26.1	---	---
AVERAGE	AVG	195	4	27.9	---	Average of an operation
				27.9	---	Average of 64 operations

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-17 Subroutine Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
SUBROUTINE CALL	SBS	091	2	25.5	Yes	---
SUBROUTINE ENTRY	SBN	092	2	---	---	---
SUBROUTINE RETURN	RET	093	1	21.9	Yes	---
MACRO	MCRO	099	4	47.4	---	---
JUMP TO SUBROUTINE	JSB	982	4	34.9	---	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-18 Interrupt Control Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
SET INTERRUPT MASK	MSKS	690	3	7.6	---	---
READ INTERRUPT MASK	MSKR	692	3	5.2	---	---
CLEAR INTERRUPT	CLI	691	3	7.2	---	---
DISABLE INTERRUPTS	DI	693	1	5.3	---	---
ENABLE INTERRUPTS	EI	694	1	5.6	---	---
INTERVAL TIMER	STIM	980	4	9.5	---	One-shot timer
				11.0	---	One-shot pulse output
				9.5	---	Scheduled interrupt
				10.8	---	Reading timer PV
				7.4	---	Stopping timer
				17.8	---	Starting pulse counting
				14.7	---	Stopping pulse counting

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-19 High-speed Counter and Pulse Output Instructions (Only for Motion Control Modules)

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
MODE CONTROL	INI	880	4	16.7	---	Starting high-speed counter comparison
				12.7	---	Stopping high-speed counter comparison
				13.3	---	Changing pulse output PV
				10.9	---	Changing high-speed counter circular value
				16.7	---	Starting pulse output comparison
				12.6	---	Stopping pulse output comparison
				14.9	---	Changing pulse output PV
				13.1	---	Changing pulse output circular value
				12.5	---	Stopping pulse output
				10.1	---	Stopping sampling counter comparison
				14.5	---	Changing sampling counter PV
				13.9	---	Changing sampling counter circular value
HIGH-SPEED COUNTER PV READ	PRV	881	4	13.5	---	Reading pulse output PV
				15.1	---	Reading high-speed counter PV
				50.8	---	Reading analog input PV
				14.3	---	Reading high-speed counter travel distance
				12.1	---	Reading high-speed counter latched value
COMPARISON TABLE LOAD	CTBL	882	4	36.5	---	Registering target value table and starting comparison for 1 target value
				259.6	---	Registering target value table and starting comparison for 48 target values
				22.1	---	Executing range comparison for 1 range
				113.7	---	Executing range comparison for 16 ranges
				22.1	---	Only registering target value table for 1 target value
				240.1	---	Only registering target value table for 48 target values
				20.9	---	Registering a sampling counter target value table and starting comparison
				42.8	---	Analog output

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
SPEED OUTPUT	SPED	885	4	23.7	---	Continuous mode
				32.7	---	Independent mode
				42.9	---	Analog output
SET PULSES	PULS	886	4	15.9	---	Setting pulse output in relative mode
				16.1	---	Setting pulse output in absolute mode
				31.5	---	Absolute output mode (electronic cam)
				35.7	---	Absolute output mode (electronic cam, 0 point can be passed)
				40.4	---	Absolute output mode (electronic cam, 0 point can be passed and the output frequency can be calculated automatically)
PULSE OUTPUT	PLS2	887	4	53.5	---	---
ACCELERATION CONTROL	ACC	888	4	42.5	---	Continuous mode
				44.1	---	Independent mode
				18.7	---	Analog output

4-1-20 Step Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
STEP DEFINE	STEP	008	2	24.3	---	Step control bit ON
				13.0	---	Step control bit OFF
STEP START	SNXT	009	2	9.1	---	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-21 I/O Refresh Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
I/O REFRESH	IORF	097	3	7.7	---	Refreshing 1 input word
				7.6	---	Refreshing 1 output word
				20.1	---	Refreshing 1 input word on Basic I/O Unit
				20.1	---	Refreshing 1 output word on Basic I/O Units
				57.6	---	Refreshing 10 input words on Basic I/O Unit
				59.9	---	Refreshing 10 output words on Basic I/O Units

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-22 Serial Communications Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
TRANSMIT	TXD	236	4	24.1	---	Sending 1 byte
				342.6	---	Sending 256 bytes
RECEIVE	RXD	235	4	36.2	---	Storing 1 byte
				348.9	---	Storing 256 bytes
CHANGE SERIAL PORT SETUP	STUP	237	3	441.1	---	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-23 Debugging Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
TRACE MEMORY SAMPLING	TRSM	045	1	34.6	---	Sampling 1 bit and 0 words
				148.3	---	Sampling 31 bits and 6 words

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-24 Failure Diagnosis Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
FAILURE ALARM	FAL	006	3	157.1	---	Recording errors
				56.0	---	Deleting errors (in order of priority)
				457.0	---	Deleting errors (all errors)
				53.6	---	Deleting errors (individually)
SEVERE FAILURE ALARM	FALS	007	3	---	---	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-25 Other Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)	Hardware implementation	Conditions
SET CARRY	STC	040	1	0.15	Yes	---
CLEAR CARRY	CLC	041	1	0.15	Yes	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-26 Block Programming Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)		Conditions
BLOCK PROGRAM BEGIN	BPRG	096	2	20.3	---	---
BLOCK PROGRAM END	BEND	801	1	17.2	---	---
Branching	IF (execution condition)	802	1	6.8	Yes	IF true
				12.2		IF false
Branching	IF (relay number)	802	2	11.0	Yes	IF true
				16.5		IF false
Branching (NOT)	IF NOT (relay number)	802	2	11.5	Yes	IF true
				16.8		IF false
Branching	ELSE	803	1	11.4	Yes	IF true
				13.4		IF false
Branching	IEND	804	1	13.5	Yes	IF true
				7.0		IF false

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

4-1-27 Special Function Block Instructions

Instruction	Mnemonic	Code	Length (steps) (See note.)	ON execution time (μs)		Conditions
GET VARIABLE ID	GETID	286	4	21.3	---	---

Note When a double-length operand is used, add 1 to the value shown in the length column in the above table.

Execution Times for Function Block Instances

The influence on the cycle time resulting from executing function block instances in FQM1-CM002/MMP22/MMA22 CPU Units is as follows:

Cycle time influence of function block instance execution =
Starting time A + I/O parameter transfer time B +
Function block execution time C

A, B, and C are as follows:

Description			Model
			FQM1-CM002 FQM1-MMP22 FQM1-MMA22
A	Starting time	Time required to start execution excluding the I/O parameter transfer time	15.0 μs
B	I/O parameter transfer time	Time for 1 BOOL I/O variable (1 bit)	1.0 μs
		Time for 1 INT, UINT, or WORD I/O variable (1 word)	0.8 μs
		Time for 1 DINT, UDINT, or DWORD I/O variable (2 words)	1.1 μs
		Time for 1 LINT, ULINT, or LWORD I/O variable (4 words)	2.2 μs
C	Function block execution time	Total of the execution times for the instructions inside the instance (same as for other ladder diagram programming)	

Example

The execution time for a function block instance with three INT input variables (one word each) and two INT output variables (one word each) would be as follows assuming that the total instruction execution time for the instructions in the function block is 10 μs.

6.8 μs + (3 + 2) × 0.3 μs + 10 μs = 18.3 μs

Note If there is more than one instance of the same function block in a program, the overall execution time would be increased by the above amount for each instance.

Number of Program Steps for Function Block Instances

The number of program steps required when function block instances are used in a user program is as follows:

Number of steps =
Number of instances × (Steps to call the function block m +
Steps to transfer I/O parameters n × Number of parameters) +
Instruction steps in function block definition p (See note.)

Note If there is more than one instance of the same function block in a program, the the number of instruction steps for the function block definition is required only for the first instance. The "Instruction steps in function block definition p" in the above formula is not multiplied by the number of instances.

Description			Time for FQM1 CPU Unit
m	Number of steps required to call the function block	---	57 steps
n	Number of steps required to transfer I/O parameters	Time for 1 BOOL I/O variable (1 bit)	6 steps
		Time for 1 INT, UINT, or WORD I/O variable (1 word)	6 steps
		Time for 1 DINT, UDINT, or DWORD I/O variable (2 words)	6 steps
		Time for 1 LINT, ULINT, or LWORD I/O variable (4 words)	12 steps
p	Number of instruction steps	Add 27 steps to the total of the instruction steps inside the function block definition (same as for other ladder diagram programming)	

Example

The number of instructions required for one instance of a function block with five INT input variables (one word each) and five INT output variables (one word each) would be as follows assuming that the function block definition contained 100 programming steps.

57 + (5 + 5) × 6 steps + 100 steps + 27 steps = 244 steps

Index

A

addressing
operands, 4
See also index registers
ASCII
converting ASCII to hexadecimal, 345
converting hexadecimal to ASCII, 341
table of characters, 7

B

Basic I/O Units
Basic I/O Unit instructions, 64, 580
BCD data, 7
bits
setting and resetting, 129
block programs
block programming instructions, 66, 608
branching, 613
description, 608–610
instruction execution times, 637

C

checksum instructions, 467
communications
description of serial communications, 582
instruction execution times, 636
receiving from RS-232C port, 587
serial communications instructions, 65, 582–595
transmitting from RS-232C port, 582
comparing tables, 530
comparison, 530
comparison instructions
execution times, 625
control bits
Sampling Start Bit, 598
Trace Start Bit, 598
conversion instructions
execution times, 629
converting
See also data, converting
counters, 152
execution times, 624
reversible counter, 163
cycle time
instruction execution times, 621

D

data
converting
radians and degrees, 400–401, 443–444
data control instructions
execution times, 633
data format
floating-point data, 425
data formats, 7
data movement instructions
execution times, 626
data shift instructions
execution times, 626
data tracing
See also tracing
debugging
debugging instructions, 65, 596–599
failure diagnosis instructions, 66, 600
debugging instructions
execution times, 636
decrement instructions
execution times, 627
degrees
converting degrees to radians, 400, 443
DM Area
using DM Area bits in execution conditions, 113
Double-precision Floating-point Input Comparison Instructions, 462
Double-precision Floating-point Instructions, 425

E

EC Directives, xxiii
errors
codes
programming, 600, 603
fatal
clearing, 603
generating, 603
instruction processing errors, 9
non-fatal
clearing, 600
generating, 600
program errors, 9
user-programmed errors, 600, 603
execution condition
outputting, 132
execution times, 621–622
exponents, 415, 457

F–G

failure diagnosis instructions
execution times, 636
fatal operating errors
generating and clearing, 603
flags

CY
clearing, 606
ER Flag, 9
Trace Busy Flag, 598
Trace Completed Flag, 598
Trace Trigger Monitor Flag, 598
floating-point data, 381, 426
comparing, 421
comparison, 421
double-precision floating-point instructions, 53
exponents, 415, 457
floating-point math instructions, 49, 380–462
format, 425
logarithms, 417, 459
square roots, 413, 456
floating-point decimal, 8
floating-point math instructions
execution times, 630
function codes
instructions listed by function codes, 78

H

high-speed counter and pulse output instructions, 521
high-speed counting
reading the PV, 527

I

I/O memory address
See also internal I/O memory address
increment instructions
execution times, 627
index registers
setting a timer/counter PV address in an index register, 222
setting a word/bit address in an index register, 221
input instructions
execution times, 622
instruction execution times, 622
instruction sets
-(410), 295
--(592), 269
*(420), 312
*B(424), 318
*BL(425), 320
*D(847), 439
*F(456), 396, 439
*L(421), 314
*U(422), 315
*UL(423), 317
+(400), 282
++(590), 265
++B(594), 273
++BL(595), 275
++L(591), 267
+B(404), 289
+BC(406), 292
+BCL(407), 293
+BL(405), 290
+C(402), 285
+CL(403), 287
+D(845), 436
+F(454), 392, 436
+L(401), 283
/(430), 321
/B(434), 328
/BL(435), 329
/D(848), 441
/F(457), 397
/L(431), 323
/U(432), 324
/UL(433), 326
ACC(888), 555
ACOS(464), 410, 452
ACOSD(855), 452
AND, 99
AND LD, 105
AND NOT, 101
ANDL(610), 353
ANDW(034), 351
APR(069), 368
ASC(086), 341
ASIN(463), 408, 450
ASIND(854), 450
ATAN(465), 412, 454
ATAND(856), 454
AVG(195), 486
-B(414), 304
--B(596), 277
-BC(416), 309
BCD(024), 334
BCDL(059), 336
-BCL(417), 310
BCMP(068), 187
BCNT(067), 375
BIN(023), 331
BINL(058), 333
-BL(415), 306
--BL(597), 279
BREAK(514), 150
BSET(071), 213
-C(412), 300
CJP(510), 141
CJPN(511), 141
-CL(413), 302
CLC(041), 606
CLI(691), 512
CMP(020), 172
CMPL(060), 175
CNT, 160

CNTR(012), 163
COLL(081), 219
COM(029), 365
COML(614), 366
COS(461), 404, 447
COSD(852), 447
CPS(114), 177
CPSL(115), 180
CTBL(882), 530
–D(846), 437
DBL(843), 433
DBLL(844), 434
DEG(459), 401, 444
DEGD(850), 444
DI(693), 513
DIFD(014), 122–124
  using in interlocks, 136
  using in jumps, 140, 144, 146
DIFU(013), 122–124
  using in interlocks, 136
  using in jumps, 140, 144, 146
DIST(080), 217
Double-precision Floating-point Input Comparison Instructions (335 to 340), 462
DOWN(522), 112
EI(694), 514
ELSE(803), 613
END(001), 134
EXP(467), 415, 457
EXPD(858), 457
-F(455), 394, 437
FAL(006), 600
FALS(007), 603
FIX(450), 386, 430
FIXD(841), 430
FIXL(451), 388, 432
FIXLD(842), 432
FLT(452), 389, 433
FLTL(453), 390, 434
FOR(512), 147
HEX(162), 345
IEND(804), 613
IF(802), 613
IL(002), 135–138
ILC(003), 135–138
INI(880), 521
IORF(097), 580
JME(005), 138
JME0(516), 145
JMP(004), 138
JMP0(515), 145
KEEP(011), 119
-L(411), 296
--L(593), 271
LD, 95
LD NOT, 97
LOG(468), 417, 459
LOGD(859), 459
MAX(182), 467
MCMP(019), 182, 196
MCRO(099), 496
MIN(183), 471
MOV(021), 199
MOVB(082), 204
MOVD(083), 206
MOVL(498), 201
MOVR(560), 221
MOVRW(561), 222
MSKR(692), 510
MSKS(690), 508
MVN(022), 200
MVNL(499), 203
NEG(160), 338
NEGL(161), 339
NEXT(513), 147
NOP(000), 134
NOT(520), 111
OR, 102
OR LD, 107
OR NOT, 104
ORW(035), 354
ORWL(611), 356
OUT, 117
OUT NOT, 118
OUTB(534), 132
PLS2(887), 550
PRV(881), 527
PULS(886), 543
PWRD(860), 461
RAD(458), 400, 443
RADD(849), 443
RET(093), 503
RSET, 125
RSTA(531), 126–129, 132
RSTB(533), 129
RXD(235), 587
SBN(092), 500
SBS(091), 491
SCL(194), 475
SCL2(486), 479
SCL3(487), 483
SET, 125
SETA(530), 126–129, 132
SETB(532), 129
SIGN(600), 349
SIN(460), 403, 446
SIND(851), 446
Single-precision Floating-point Input Comparison Instructions (329 to 334), 421
SNXT(009), 563
SPED(885), 537
SQRT(466), 413, 456

SQRTD(857), 456
STEP(008), 563
STUP(237), 592
TAN(462), 406
TAND(853), 449
TCMP(085), 185
testing bit status, 113
TIM, 153
TIMH(015), 156
TMHH(540), 158
TRSM(045), 596
TST(350), 113
TSTN(351), 113
TXD(236), 582
UP(521), 112
XCGL(562), 216
XCHG(073), 215
XFER(070), 211
XFRB(062), 208
XNRL(613), 363
XNRW(037), 362
XORL(612), 360
XORW(036), 358
ZCP(088), 193
ZCPL(116), 196
instructions, 87
Basic I/O Unit instructions, 64
block programming instructions, 66, 608
classified by function, 12
comparison instructions, 28, 167–193
controlling execution conditions
UP(521) and DOWN(522), 112
controlling high-speed counters and pulse outputs, 521
conversion instructions, 44, 331
counter instructions, 26, 152
data control instructions, 58, 475–490
data movement instructions, 31, 199
data shift instructions, 34, 225–264
debugging instructions, 65, 596–599
decrement instructions, 38, 265–280
differentiated instructions, 2
execution times, 622
failure diagnosis instructions, 66, 600
floating-point math instructions, 49, 380–462
high-speed counter instructions, 521
I/O Refresh instruction, 580
increment instructions, 38, 265–280
input comparison instructions, 167–172, 421, 462
instruction execution times, 621
instruction variations, 3
interrupt control instructions, 61, 508
listed alphabetically, 69
listed by function code, 78
logic instructions, 47, 351–367
number of steps, 621
pulse output instructions, 521
sequence control instructions, 23, 134–151
sequence input instructions, 18, 95–116
sequence output instructions, 20, 117–129
serial communications instructions, 65, 582–595
special math instructions, 49, 368–376
step instructions, 64, 562–579
steps per instruction, 622
subroutine instructions, 60, 491–503
symbol math instructions, 39, 281–330
table data processing instructions, 53, 57, 467
task control instructions, 68
timer instructions, 26, 152
interlocks, 135–138
internal I/O memory address
setting a timer/counter PV address in an index register, 222
setting a word/bit address in an index register, 221
interrupt control instructions
execution times, 634
interrupts
clearing, 512
disabling all, 513
enabling all, 514
masking, 508
reading mask status, 510
scheduled
reading interval, 510

J

jumps, 138, 145
CJP(510) and CJPN(511), 141

L

ladder diagrams
controlling bit status
using DIFU(013) and DIFD(014), 122–124
using KEEP(011), 119–122
using SET and RSET, 125–126
using SETA(530) and RSTA(531), 126–129, 132
latching relays
using KEEP(011), 119
logarithm, 417, 459
logic instructions
execution times, 629
loops
BREAK(514), 150
FOR(512) and NEXT(513), 147

M

mathematics
averaging, 486
exponents, 415, 457

finding the maximum in a range, 467
finding the minimum in a range, 471
floating-point addition, 392, 436
floating-point division, 397
floating-point math instructions, 49, 380–462
floating-point multiplication, 396, 439
floating-point subtraction, 394, 437
logarithm, 417, 459
See also trigonometric functions
special math instructions, 49, 368–376
square root, 413, 456
symbol math instructions, 39, 281–330
trigonometric functions, 368

N

non-fatal operating errors
generating and clearing, 600

O

operands, 4
inputting data, 4
output instructions
execution times, 623

P

PC memory address
See also internal I/O memory address
precautions
general, xviii
safety, xviii
program capacity, 2
programming
creating step programs, 562
instruction execution times, 622
preparing data in data areas, 213
program capacity, 2
program errors, 9
use of TR Bits, 110
pulse outputs, 521
controlling, 521, 555

R

radians
converting radians to degrees, 401, 444
range comparison, 193, 196
refreshing
differentiated refreshing instructions, 109
immediate refreshing instructions, 109
with IORF(097), 580
resetting bits, 129
RS-232C port
receiving from RS-232C port, 587
transmitting from RS-232C port, 582

S

safety precautions
See also precautions
searching instructions, 467
self-maintaining bits
using KEEP(011), 121
sequence control instructions
execution times, 623
serial communications
description, 582
serial communications instructions
execution times, 636
setting bits, 129
signed binary data, 7
removing sign, 349
Single-precision Floating-point Input Comparison Instructions, 421
special math instructions
execution times, 630
speed outputs, 537
square root
floating-point data, 413, 456
stack instructions, 467
execution times, 632
stack processing
execution times, 632
stacks
stack instructions, 467
step instructions
execution times, 634–635
step programs
creating, 562
subroutine instructions
execution times, 633
subroutines
execution times, 633
symbol math instructions
execution times, 628

T

tasks
block programs within tasks, 609
task control instructions, 68
timers, 152
execution times, 624
tracing
flags and control bits, 598
trigonometric functions

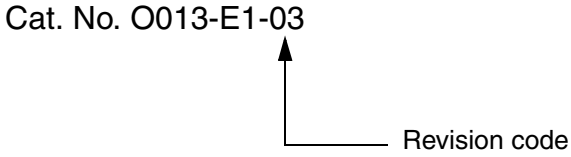
arc cosine, 410, 452
arc sine, 408, 450
arc tangent, 412, 454
converting degrees to radians, 400, 443
converting radians to degrees, 401, 444
cosine, 404, 447
sine, 403, 446
tangent, 406, 449

U–W

unsigned binary data, 7

Revision History

A manual revision code appears as a suffix to the catalog number on the front cover of the manual.



The following table outlines the changes made to the manual during each revision. Page numbers refer to the previous version.

Revision code	Date	Revised content
01	December 2005	Original production
02	November 2006	Information added and changed for the addition of functionality supported by unit version 3.2 of the Coordinator Module and Motion Control Modules.
03	April 2008	**Page vii:** Updated description for mounting CJ-series Units **Page vii:** Added version upgrade table for version 3.2 to version 3.3. **Page 538:** Added paragraph at bottom of page. **Page 545:** Added a paragraph above operand specifications table. **Page 549:** Changed T settings for Equal Flag in the table. **Page 551:** Added paragraph above operand specifications table. **Page 553:** Added information on version 3.3 or later above notes. **Page 556:** Added a paragraph at bottom of page.

OMRON Corporation
Industrial Automation Company
Control Devices Division H.Q.
Motion Control Department
Shiokoji Horikawa, Shimogyo-ku,
Kyoto, 600-8530 Japan
Tel: (81) 75-344-7173/Fax: (81) 75-344-7149

2-2-1 Nishikusatsu, Kusatsu-shi,
Shiga, 525-0035 Japan
Tel: (81) 77-565-5223/Fax: (81) 77-565-5568

Regional Headquarters
OMRON EUROPE B.V.
Wegalaan 67-69-2132 JD Hoofddorp
The Netherlands
Tel: (31)2356-81-300/Fax: (31)2356-81-388

OMRON ELECTRONICS LLC
One Commerce Drive Schaumburg,
IL 60173-5302 U.S.A.
Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

OMRON ASIA PACIFIC PTE. LTD.
No. 438A Alexandra Road # 05-05/08 (Lobby 2),
Alexandra Technopark, Singapore 119967
Tel: (65) 6835-3011/Fax: (65) 6835-2711

OMRON (CHINA) CO., LTD.
Room 2211, Bank of China Tower,
200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-5037-2222/Fax: (86) 21-5037-2200

OMRON Industrial Automation Global: **www.ia.omron.com**

Authorized Distributor:

Printed in Japan

Cat. No. O013-E1-03